SCA Service Component Architecture

EJB Session Bean Binding

SCA Version 1.00, February 22 2007

Technical Contacts:

Ron Barack	SAP AG	ron.barack@sap.com
Henning Blohm	SAP AG	henning.blohm@sap.com
Dave Booz	IBM Corporation	booz@us.ibm.com
Rashmi Hunt	IBM Corporation	hrashmi@us.ibm.com
Michael Keith	Oracle Corporation	michael.keith@oracle.com
Michael Rowley	BEA Systems, Inc.	mrowley@bea.com

Copyright Notice

License

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.

Table of Contents

1 Overview

EJB session beans are a common technology used to implement business services. The ability to integrate SCA with session bean based services is useful because it preserves the investment incurred during the creation of those business services, while enabling the enterprise to embrace the newer SCA technology in incremental steps. The simplest form of integration is to simply enable SCA components to invoke session beans as SCA services. There is also a need to expose SCA services such that they are consumable by programmers skilled in the EJB programming model. This enables existing session bean assets to be enhanced to exploit newly deployed SCA services without the EJB programmers having to learn a new programming model.

This document explains the EJB SCA binding. This proposal describes how to integrate a previously deployed session bean into an SCA assembly, and how to expose SCA services to clients which use the EJB programming model.

The EJB programming model supports stateful and stateless session beans. Stateful session beans can implement a conversational interaction with their clients. Stateless session bean are not conversational and instances may receive calls from any number of clients in any order.

The EJB binding supports the stateless session bean model as well as the stateful session bean model.

The EJB Session Bean binding enables:

- SCA developers to treat previously deployed session beans as SCA services, by wiring them into an SCA assembly (SCA reference).
- SCA service deployers to expose a SCA service as a session bean for consumption by Java EE applications.

The use of EJBs and EJB modules as SCA component implementations is beyond the scope of this specification and is described in the Java EE integration specification [1]. The following diagram shows the use of the EJB SCA binding on both services and references.

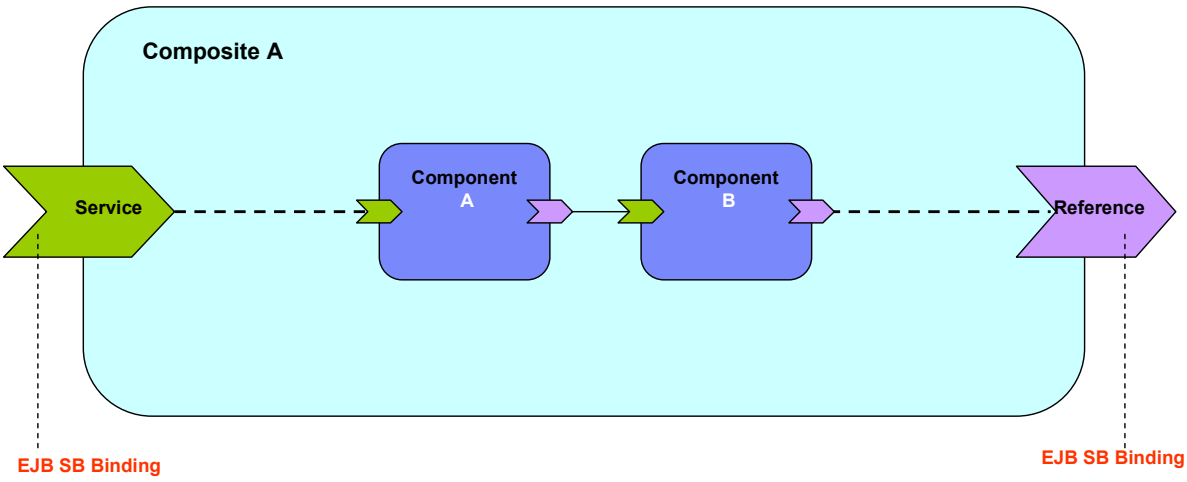


Figure 1: EJB Binding used on Services and References

2 Session Bean Binding

2.1 Session Bean Binding Schema

The EJB session bean binding element is defined by the following pseudo-schema.

```
<binding.ejb
   homeInterface="NCName"?
   ejb-link-name="NCName"?
   session-type="stateful or stateless"?
   ejb-version="EJB2 or EJB3"?
   name="NCName"?
   policySets=" sca:listOfQNames"?
   requires="sca:listOfQNames"?
   uri="anyURI"?
>
   <!-- additional elements here -->

</binding.ejb>
```

- ***/binding.ejb/@homeInterface*** - the homeInterface attribute of the EJB binding is the session bean's home interface, and is used when exposing SCA services as EJB 2.x session beans.

- ***/binding.ejb/@ejb-link-name*** - the ejb-link-name attribute provides a means for integrating EJB reference resolution with SCA. When used on a binding for a reference, it allows a SCA client to bind to an EJB that is packaged in the same Java EE EAR file as the SCA client. When used on a service binding, it exposes an <ejb-link/> target for Java EE clients that want to use Java EE assembly to wire to the SCA service. This attribute is functionally equivalent to using the <ejb-link/> subelement of the <ejb-ref/> element in an EJB deployment descriptor. The value of this attribute is supplied by an application assembler, and is in the form as specified by the Java EE specification (i.e. <jar-name>#<ejb-name>).

- ***/binding.ejb/@session-type*** – the session-type attribute is used to indicate the kind of session bean contract to be used when referencing a session bean or when exposing as a session bean. The default is "Stateless". Admissible values are "Stateless" and "Stateful". It is not necessary to specify the attribute, if it can be inferred from the interface of the reference or service. If the latter is conversational, the stateful session bean contract applies, and if it is not conversational, the stateless session bean contract applies. A mismatch of the attribute value and interface provided meta-data will raise an IllegalStateException at latest at runtime. See also the section Interface Mapping.

- **/binding.ejb/@ejb-version** – the ejb-version attribute is used to indicate the EJB client view exposed by the EJB binding when used on an SCA service. This attribute has no meaning when used on a reference. The value 'EJB2' indicates that an EJB client MUST interact with the binding using the EJB 2.x client view. The value 'EJB3' indicates the desire to expose an EJB 3.0 client view.

The base SCA binding schema provides an attribute called **uri**, that is used to denote the URI of an endpoint. In the context of the SCA EJB binding, the **uri** attribute is defined as follows:

- ***/binding.ejb/@uri*** – optional attribute that specifies the URI of a session bean endpoint. For EJB 2.x, this is the endpoint of the session home. For interoperability the form of the URI is defined by CORBA in the CORBA Services specification [2], and is a standard URI form for referring to remotable CORBA objects. Briefly, the corbaname URI format looks like this:
    - corbaname:iiop:<hostName>:<port>/<key string>#<path to home>

  Typically, a corbaname URI doesn't include all these components. The following example shows a corbaname URI that uses the default ORB configuration to find an EJB home at ejb/MyHome in the JNDI directory:
    - corbaname:rir:#ejb/MyHome

  Other forms of URI specification are admissible when interoperability is of no concern.

2.2 Interface Mapping

When used with the EJB binding, a service or reference interface must be compatible with a session bean interface, according to the following rules:

- The interface offered by a reference MUST be remotable if the remote session bean interface is being accessed, and MUST be local if the local session bean interface is being accessed.
- The methods on the session bean MUST be a compatible superset of the methods in the interface used by the reference.
- The interface used by a reference MAY NOT contain any methods inherited from EJBObject or EJBLocalObject.
- Compatibility for an individual method is defined by the SCA Assembly Model Specification [4], and can be stated simply as compatibility of the signature. That is, the method name, input types, and output types MUST be identical.
- The order of the input and output types also MUST be identical.
- Except for RemoteExceptions, the set of Faults and Exceptions declared by the SCA reference interface MUST be the same or a superset of those specified by the EJB interface.
- The interface offered by a service or reference with an EJB binding MUST NOT be *conversational* in the case of a stateless session bean and MUST be conversational in the case of a stateful session bean (exposure and consumption).

- The interface offered by a service or reference MAY be an SCA business interface or an EJB 3.0 remote or local interface. The interface is considered non-conversational unless one of the following conditions applies:
    - The interface is marked as conversational using the @Conversational SCA annotation.
    - The binding.ejb element has an attribute session-type with value "Stateful".
- For bindings that consume EJB 3.0 beans, any method marked with the @endsConversation annotation in the interface used by a reference MUST map to a method marked with @Remove in the session bean's implementation class. The interface used by a reference SHOULD contain @endsConversation annotations on all methods that map to @Remove methods in the session bean's implementation class. This assures that the resources associated with the binding are freed when the bean is no longer needed.

2.2.1 EJBObject and EJBLocalObject Interfaces

The interfaces exposed from EJB 2.X beans inherit from either EJBObject or EJBLocalObject. EJBObject and EJBLocalObject contain methods directed toward the management of bean instances, meaning that the exposed 2.X interfaces mix business and infrastructure methods in a way that makes them poorly suited for use in SCA assemblies. EJB 2.X beans developed using the "Business Interface Pattern" will already have an interface that is suitable for SCA assembly. In other cases, a suitable interface may be quickly derived from the SessionBean interface. However, the session bean interface itself cannot be used as the interface of a reference binding.

When SCA Services are exposed as EJB 2.X session beans, the exposed interface will inherit from EJBObject or EJBLocalObject. Section 2.4 describes the behavior associated with each inherited method.

2.2.2 Stateful Session Bean Home Interface

SCA services have no support for a concept like EJB home interfaces. Existing EJB 2.x stateful session beans may however rely on the use of home interface Create<METHOD> methods for initialization. In order to accommodate the use of the home interface for stateful session beans, the following rules apply:

- Methods offered by the reference interface that are of the form create<METHOD>(<arg>*) and that do not match any method on the EJB local or remote business interface, according to the rules above, but do match a create<METHOD> method on the bean's corresponding local or remote home interface are mapped to that matching home interface method. This mapping ignores javax.ejb.CreateExceptions.
- A call to such a method on a given service reference starts an SCA conversation and creates a new session object for the stateful session bean by forwarding the call to the mapped home interface create<METHOD> method.

See also the section Conversational Service of a Stateful Session Bean for more details on conversations over stateful session beans via the EJB binding.

2.3 Reference Binding

When used on a reference, the EJB binding specifies the means for connecting an SCA component to a previously deployed or co-deployed session bean.

The reference interface used with the EJB binding can be either a remote or local session bean interface. SCA deployment logic and the binding implementation will introspect the reference interface class to determine whether it is local or remote. If an SCA component needs to access both the local and remote interface of a session bean, then this should be modeled in SCA assembly through two references, one with the local interface and one with the remote interface.

The ***/binding.ejb/@ejb-link-name*** and ***/binding.ejb/@uri*** attributes are mutually exclusive when used on an SCA reference because they represent alternate ways to provide the same configuration.

The following example shows a reference binding using a corbaname URI:

```
<reference name="CandidateCheck">
   <interface.java interface="com.app.jobbank.CandidateCheck"/>
   <binding.ejb uri="corbaname:rir:#ejb/CandidateCheckHome"/>
</reference >
```

The specific **uri** would be supplied prior to the completion of deployment.

The following example is a reference binding using an ejb-link.

```
<reference name="CandidateCheck">
 <interface.java interface="com.app.jobbank.CandidateChk"/>
 <binding.ejb ejb-link-name="candidateEJB.jar#CandidateChk"/>
</reference >
```

2.3.1 Conversational Nature of Stateful Session Beans

Stateful session beans fit nicely into the SCA concept of a conversation (see SCA Assembly Specification [4]). This section defines the rules for mapping Stateful Session beans to SCA conversations.

When using an EJB 3 client view, the following rules apply:

- If there is no ongoing conversation, any call to a business method of the reference interface creates a new session object (see [2]) for the bean and associates it with the newly created SCA conversation.
- When the SCA conversation with a stateful session bean ends, for example as a result of a call to CallableReference.getConversation().end(), the associated session object will be removed.

- In order to assure the prompt release of resources associated with the referenced session bean, clients are responsible for signaling the end of the conversation by calling a method that maps to a method marked with the @Remove annotation in the session bean's implementation class. Calls to CallableReference.getConversation().end() can at most release resources associated with the binding itself, and are not responsible for ending the EJB conversation.

When using the EJB 2.x client view, the following rules apply:

- To start a conversation, the create<METHOD> mapping, as described in Interface Mapping section, serves as starting point of an SCA conversation.

- Calling a business method before initializing the conversation will raise an IllegalStateException. Similarly, a call to a reference interface method that was matched against an EJB create<METHOD> method during an ongoing conversation will raise an IllegalStateException.

- When the SCA conversation with a stateful session bean ends, for example as a result of a call to ServiceReference.endSession(), the associated session object will be removed by calling the EJB Home remove() method.

2.3.2 Exception Handling

Exception handling for conversations with session beans has been specified in chapter 13 of the EJB 3 specification [2] and in Chapter 18 of the EJB 2.1 specification [2]. The reference binding for session beans can be imagined to consist of two consecutive invocation paths:

1. SCA business interface to EJB business interface (if different)
2. EJB Business interface to session bean instance

For the second invocation path, the rules laid out in the EJB specification apply. For the first invocation path, the following rules apply:

1. any business exception (see [3]) will be re-thrown by the binding implementation while keeping the current conversation ongoing.
2. any other exception will be wrapped in a ServiceRuntimeException which will be thrown by the binding implementation. Any ongoing conversation will be terminated.

2.3.3 Packaging

There is no requirement to package the session bean home interface or client stubs with an SCA component that uses the Session bean binding. The Sesseion Bean binding implementation should be able to dynamically lookup, create and invoke the bean without the usual EJB client classes.

2.4 Service Binding

When used on an SCA service, the EJB SCA binding causes the service to be exposed as a session bean. This enables a client that is using the EJB programming model to call the SCA service using its native programming model.

The ***/binding.ejb/@homeInterface*** attribute is used to indicate the Session Home interface that an EJB client will use to bootstrap itself with the SCA service, just as it would with any other session bean. The current specification allows for home interfaces that have exactly one create<METHOD> with no arguments.

The following is an example of a service using the EJB binding.

```
<service name="JobBank">
   <interface.java interface="com.app.jobbank.JobBankService"/>
   <binding.ejb
   uri="corbaname:rir:#ejb/JobBankServiceHome"
   homeInterface="com.app.jobbank.JobBankServiceHome"
      ejb-link-name="jobbankEJB.jar#JobBankComponent"/>
</service>
```

A corresponding local home interface com.app.jobbank.JobBankServiceHome looks like this:

```
package com.app.jobbank;

import javax.ejb.CreateException;
import javax.ejb.EJBLocalHome;

public interface JobBankServiceHome extends EJBLocalHome {
    JobBankService create()  throws CreateException;
}
```

Similarly, the remote home interface can be formulated by extending javax.ejb.EJBHome and making sure to declare a RemoteException:

```
package com.app.jobbank;

import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface JobBankServiceHome extends EJBHome {
    JobBankService create()  throws CreateException, RemoteException;
}
```

In the corbaname used in this example, the first part of the URI (up to the #) would logically be supplied by the target deployment environment. See [4] for a discussion of base URIs provided by an SCA domain configuration. The remainder of the name would be provided prior to completion of deployment. The example above shows the URI that a client would use after deployment. Prior to deployment, it should be possible for an assembler or developer to specify only the last portion of the URI (i.e. everything following the #).

306 The service interface used with the EJB binding can be either a remote or local session bean
307 interface. SCA deployment logic and the binding implementation will introspect the interface
308 class to determine whether it is local or remote. If an SCA component needs to be exposed as
309 both a local and remote session bean, then this should be modeled in SCA assembly through two
310 services, one with the local interface and one with the remote interface.

311 When used on a service binding, **ejb-link-name** and **uri** are NOT mutually exclusive. They each
312 provide a means for wiring to the SCA service depending on the locality of the client EJB
313 reference. For example, an SCA service packaged with an JEE EJB application could be exposed
314 for consumption by local EJB clients (using the ejb-link-name element) and remote EJB clients
315 (using the uri).

316 The service interface used with the EJB binding can be conversational. If so, the SCA service will
317 be exposed by a stateful session bean contract, so that EJB clients will be able to maintain
318 conversations across multiple method invocations, according to the EJB specification.

319 In that case, the creation of a Session Object (see [2]) marks the start of the conversation with
320 the SCA service and the removal of the Session Object marks the end the conversation.

321 If the service interface is not conversational, the SCA service will be exposed by the stateless
322 session bean contract, according to the EJB specification. In particular, there will be no
323 conversational service exposure, but instead, every stateless bean method invocation
324 corresponds to a non-conversational SCA service method invocation.

325 From the perspective of an EJB client (local and remote), SCA services that are exposed as
326 session beans (stateful or stateless) are not distinguishable from ordinary session beans.

327 Specifically, this means that a local client will be able to reference the SCA service as a session
328 bean using ejb-(local)-ref declarations in the appropriate locations and by issuing JNDI lookups
329 or relying on dependency injection mechanisms. If the service is exposed as EJB 2.x session
330 bean, by virtue of a home interface specification, the client needs to be aware of the EJB 2.x
331 home interface contract.

332 Similarly remote EJB clients are expected to be able to consume SCA services that are exposed
333 as session beans just as they are able to consume ordinary session beans.

334

335 2.4.1 Handling methods from EJBObject and EJBLocalObject

336

337 This section describes the behavior of the methods that EJB 2.X service bindings inherit from the
338 EJBObject and EJBLocalObject interfaces.

339

Method	Behavior
Remove	For conversational services, this is functionally equivalent to a call to ServiceReference.destroy. Any resources associated with the binding, and the component to which it promotes are released. For non-conversational services, this is a no-op.
getPrimaryKey	Throws an EJBException
isIdentical	Tests whether the service component, to which the binding of the current promotes, is the same instance as the one to which the specified object promotes.
getEJB(Local)Home	Returns an implementation of the interface

	specified as ***/binding.ejb/@homeInterface***. The instance may be used to create or remove bean instances.

340
341
342

3 Appendix A - Use Cases

The following use cases provide some examples of the usage of the SCA EJBSessionBean binding.

3.1 Consuming an Existing EJB SOA Service

An SCA service is developed that needs to call a business service which is already deployed and running in a Java EE server. The SCA service will be deployed into an SCA runtime somewhere in the enterprise that is not necessarily a Java EE runtime. The business service was implemented as a session bean. The SCA service defines a reference to the business service, and the deployer attaches an EJB binding to the reference. In this use case, the EJB remote interface is the business interface.

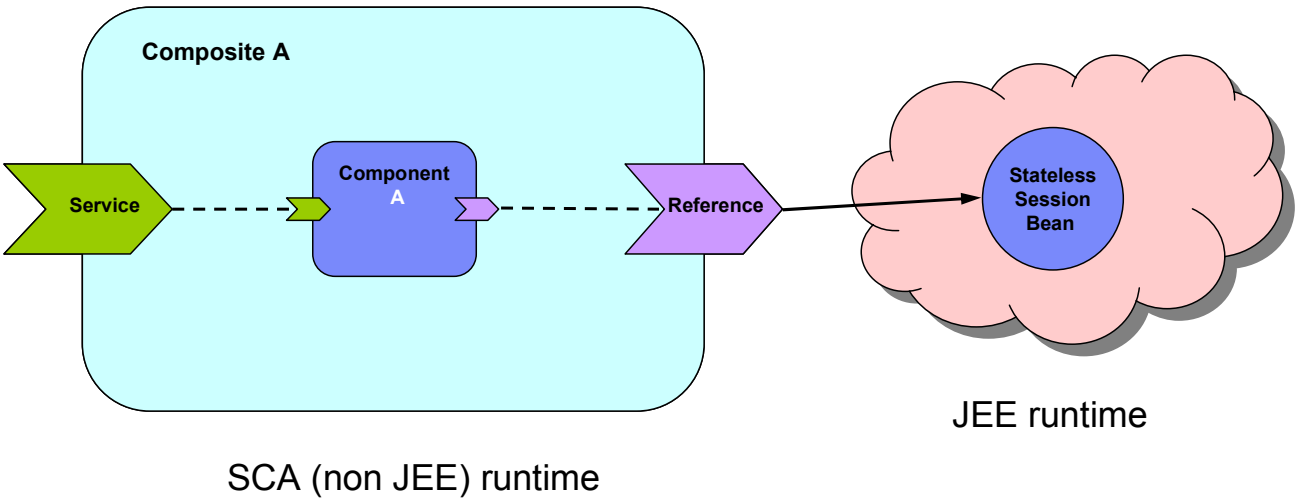


Figure 2: SCA Reference invoking EJB Session Bean

The reference in the deployed sca.composite file looks like this:

```
<reference name="CandidateCheck">
 <interface.java interface="com.app.jobbank.CandidateChk"/>
 <binding.ejb uri="corbaname:rir:#ejb/CandidateChkHome"/>
</reference >
```

3.2 Exposing an SCA Service with an EJB SCA Binding

An SCA service is developed that will be called from a Java EE environment. The Java EE programmer doesn't know the SCA programming model and therefore wants to use the Java EE programming model that he knows in order to invoke the SCA service (i.e. new initialContext(), nc.lookup(), etc.). In this case, the SCA service has to be deployed into a runtime that is capable of supporting the EJB binding. Note that deployment of this service can result in the generation and deployment of a session bean, along with its home interface. This aspect is significantly different from the previous use case.

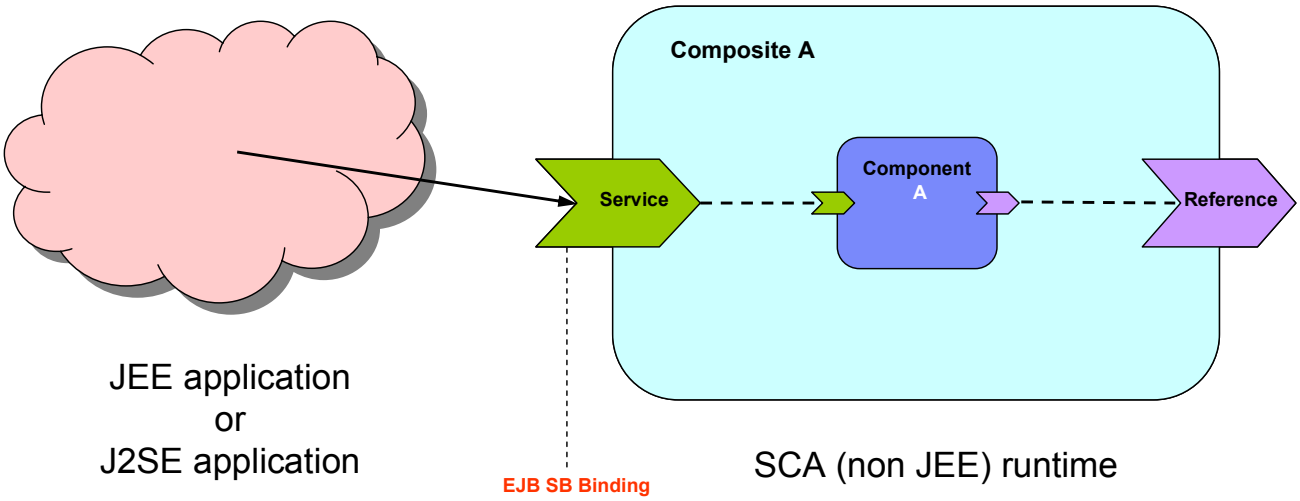


Figure 3: SCA Service accessed as an EJB Session Bean

Since the client will use the standard Java EE programming model, the client needs to know the home interface of the SCA service. The service in the sca.composite file will look like this:

```
<service name="CompanyInfo">
 <interface.java interface="com.app.jobbank.CompanyInfo"/>
 <binding.ejb uri="corbaname:rir:#ejb/CompanyInfoHome"
   homeInterface="com.app.jobbank.CompanyInfoHome"
   ejb-version="EJB2"/>
 <reference>CompanyInfoComponent/CompanyInfo</reference>
</service>
```

The client code as per the standard Java EE programming model looks like this:

```
Context initialContext = new InitialContext(env);
CompanyInfoHome companyInfoHome= (CompanyInfoHome)
     initialContext.lookup("corbaname:rir:#ejb/CompanyInfoHome");

CompanyInfo companyInfo = companyInfoHome.create();
companyInfo.getCompanyInfo( "ACME Corp");
```

3.3 Consuming Existing Local EJB SOA Services

This use case is similar to the use case in section 3.1, except that the SCA service is going to be deployed into a Java EE capable JVM, and it is the same JVM as the EJB service. In this use case, the EJB's local interface is used as the business interface.

Note that the SCA client could also use the EJB remote interface. If an SCA component wanted to access both the local and remote interface, then it would declare 2 references (one with the local interface, one with the remote interface).

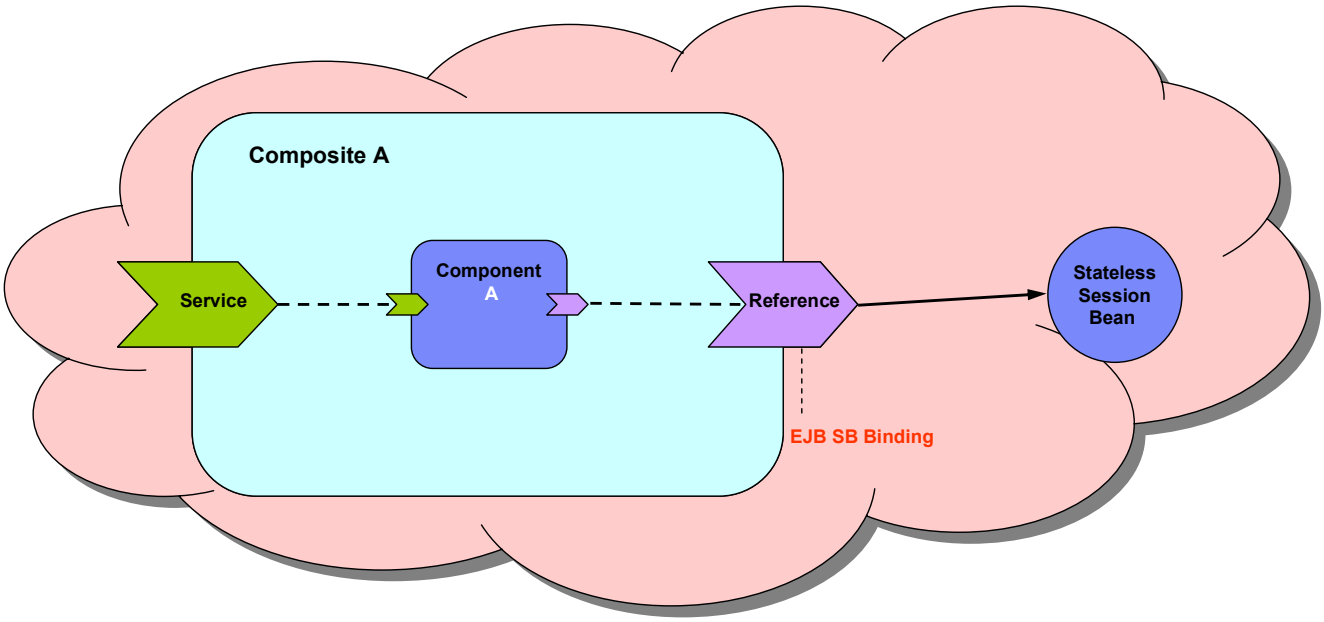


Figure 4: SCA reference consuming a Local EJB service

The example below shows the usage of a local interface in the reference definition.

```
<reference name="CandidateCheck">
 <interface.java interface="com.app.jobbank.CandidateCheckLocal"/>
 <binding.ejb
      uri="corbaname:rir:#ejb/CandidateCheckHome"/>
</reference>
```

3.4 Exposing an SCA Service with a Local SLSB SCA Binding

This use case is similar to the use case in section 3.2, except that the SCA service is going to be deployed into the same JVM as the client. This use case allows for the possibility that the SCA service is exposed as a local EJB interface. Note that deployment of this service will effectively

result in the generation and deployment of a session bean with a local interface and a local home interface.

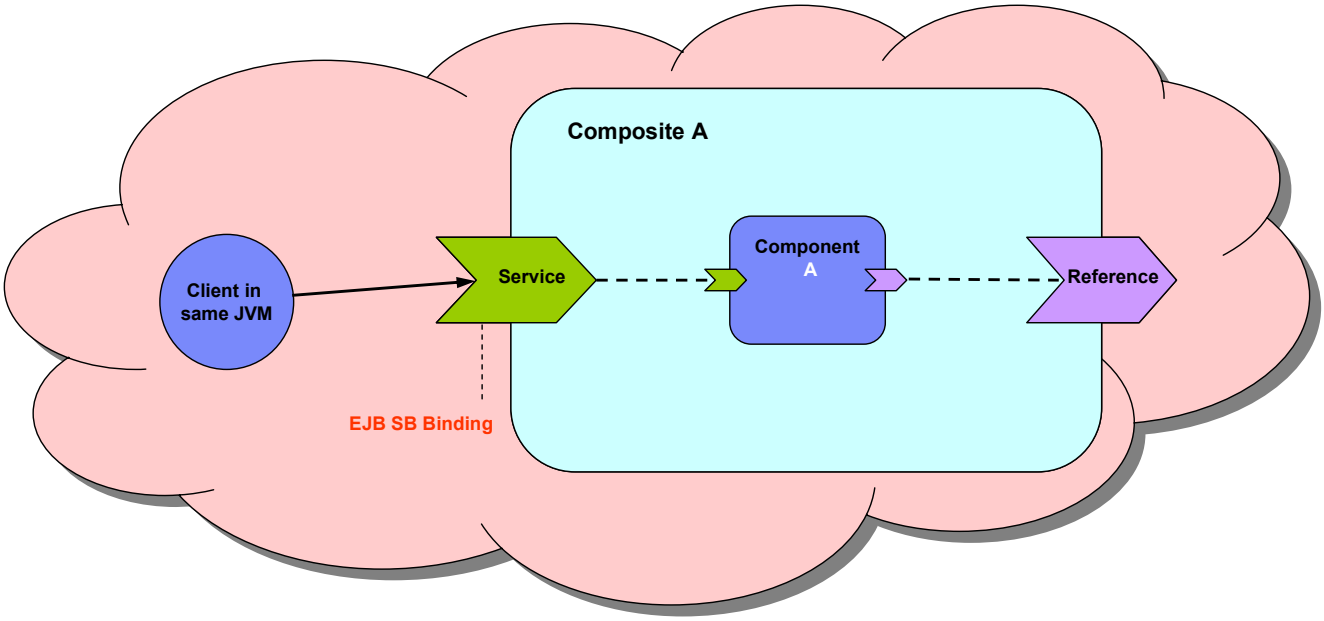


Figure 5: SCA Service exposed as a Local session bean

The following is an example:

```
<service name="CompanyInfo">
 <interface.java interface="com.app.jobbank.CompanyInfoLocal"/>
 <binding.ejb uri="corbaname:rir#ejb/CompanyInfoHome"
     homeInterface="com.app.jobbank.CompanyInfoLocalHome"/>
 <reference>CompanyInfoComponent/CompanyInfo</reference>
</service>
```

3.5 Consuming an EJB Service inside a Java EE EAR file

This use case is similar to sections 3.1 and 3.3, except that the SCA service is going to be packaged inside a Java EE EAR file. By packaging it in this way, the SCA reference binding can be configured as if it were an <ejb-ref> with the <ejb-link> subelement.

The following is an example of the SCA reference binding.

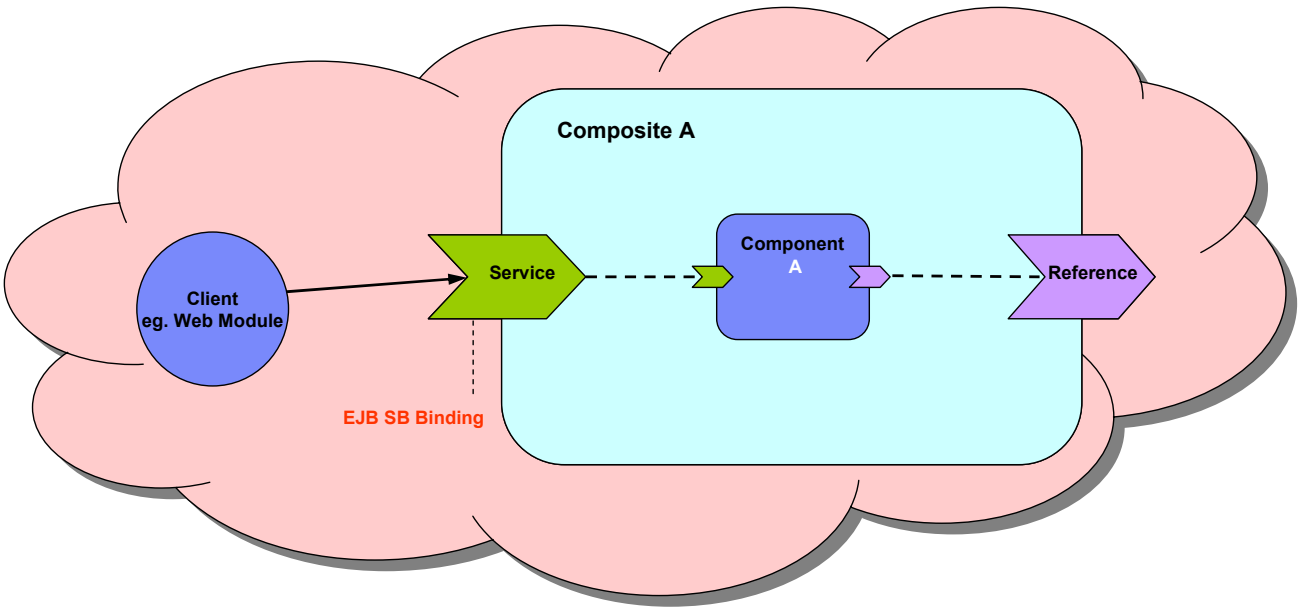
```
<reference name="CandidateCheck">
 <interface.java interface="com.app.jobbank.CandidateChk"/>
 <binding.ejb ejb-link-name="candidateEJB.jar#CandidateChk"/>
</reference >
```

The following is an <ejb-ref/> snippet that is functionally equivalent to the SCA reference above.

```
<ejb-ref>
 <ejb-ref-name>CandidateCheck</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.app.jobbank.CandidateChkHome</home>
 <remote>com.app.jobbank.CandidateChk</remote>
 <ejb-link>candidateEJB.jar#CandidateChk</ejb-link>
</ejb-ref>
```

3.6 Exposing an SCA Service inside a Java EE EAR file

This use case is similar to sections 3.2 and 3.4, except that the SCA service is going to be deployed inside a Java EE EAR file so that it can be referenced by an EJB client, using the EJB assembly model.



Caller and SCA Composite within one EAR file

Figure 6: SCA Service with client within one EAR file

The following is an example of the SCA service binding.

```
<service name="CompanyInfo">
 <interface.java interface="com.app.jobbank.CompanyInfo"/>
 <binding.ejb
   homeInterface="com.app.jobbank.CompanyInfoHome"
   ejb-link-name="companyInfoEJB.jar#CompanyInfoComponent"/>
 <reference>CompanyInfoComponent/CompanyInfo</reference>
</service>
```

The following is an example of an EJB deployment descriptor created by the client that is wired to the SCA Service binding.

```
<ejb-ref>
 <ejb-ref-name>ejb/CompanyInfo</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.app.jobbank.CompanyInfoHome</home>
 <remote>com.app.jobbank.CompanyInfo</remote>
 <ejb-link>companyInfoEJB.jar#CompanyInfoComponent</ejb-link>
</ejb-ref>
```

Note: There is a variant of this use case that should be considered. If the SCA service is in the same EJB module as the client, then the ejb-link specified by the client does not have to include the EJB module jar name.

4 Appendix B - EJB Binding Schema

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:sca="http://www.osoa.org/xmlns/sca/1.0"
        targetNamespace="http://www.osoa.org/xmlns/sca/1.0"
        elementFormDefault="qualified">
   <include schemaLocation="sca-core.xsd"/>

   <element name="binding.ejb" type="sca:EJBSessionBeanBinding"
            substitutionGroup="sca:binding" />

    <simpleType name="BeanType">
        <restriction base="string">
            <enumeration value="stateless"/>
            <enumeration value="stateful"/>
        </restriction>
    </simpleType>

    <simpleType name="VersionValue">
        <restriction base="string">
            <enumeration value="EJB2"/>
            <enumeration value="EJB3"/>
        </restriction>
    </simpleType>


     <complexType name="EJBSessionBeanBinding">
       <complexContent>
        <extension base="sca:Binding">
         <sequence>
            <any namespace="##other" processContents="lax"
                 minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="homeInterface" type="NCName" use="optional"/>
         <attribute name="ejb-link-name" type="NCName" use="optional"/>
         <attribute name="session-type" type="sca:BeanType" use="optional"
            default="stateless"/>
         <attribute name="ejb-version" type="sca:VersionValue" use="optional"
            default="EJB2"/>
         <anyAttribute namespace="##any" processContents="lax"/>
        </extension>
       </complexContent>
     </complexType>

</schema>
```

5 References

[1] SCA Java EE Implementation Specification

(to be published – see
http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications)

[2] Enterprise JavaBeans Specification

http://java.sun.com/products/ejb/docs.html

[3] SCA Java Common Annotations and APIs Specification

http://www.osoa.org/download/attachments/35/SCA_JavaAnnotationsAndAPIs_V100.pdf

[4] SCA Assembly Model Specification

http://www.osoa.org/download/attachments/35/SCA_AssemblyModel_V100.pdf