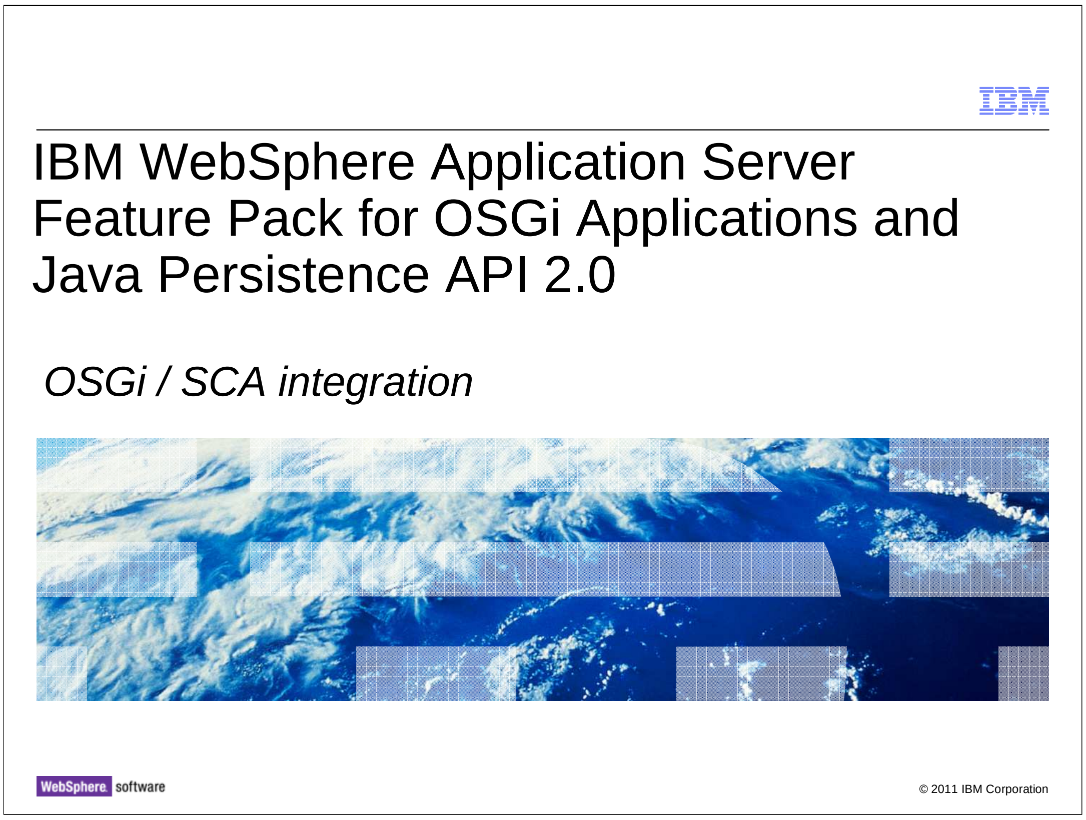
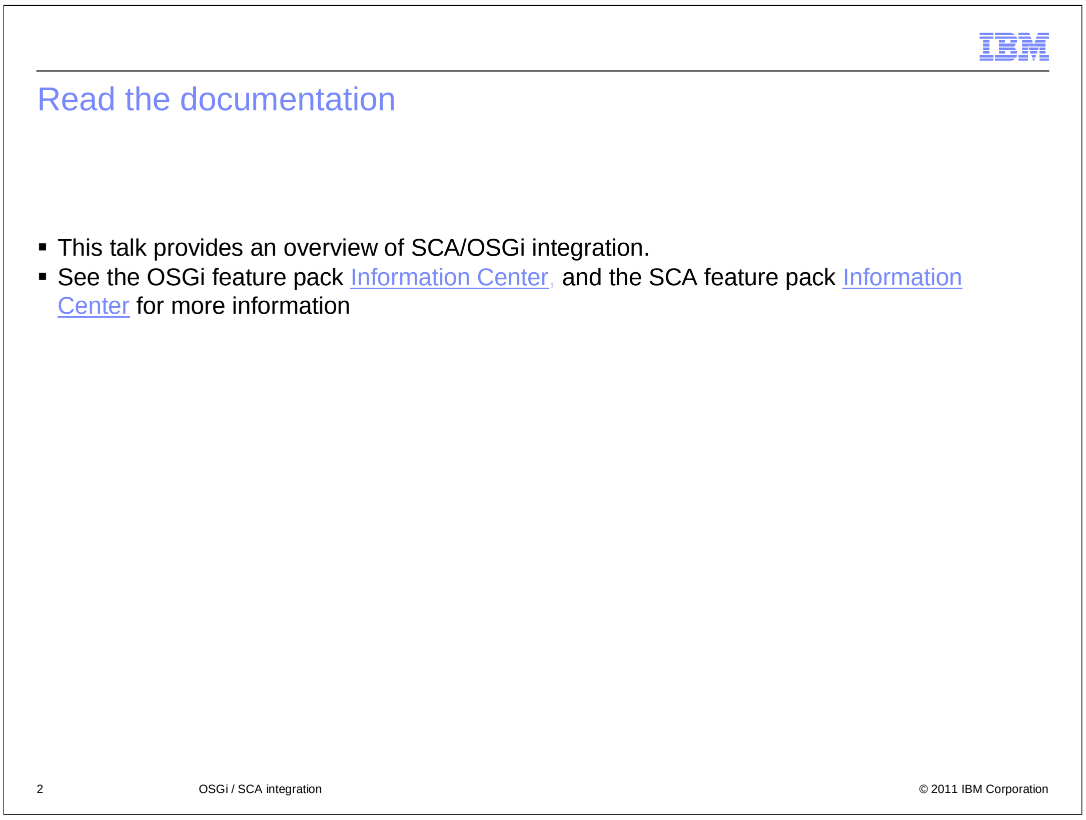


IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API 2.0

OSGi / SCA integration

WebSphere software

This module will discuss OSGi – SCA integration in the IBM® WebSphere® Application Server Feature Pack for OSGi Applications and Java™ Persistence API 2.0

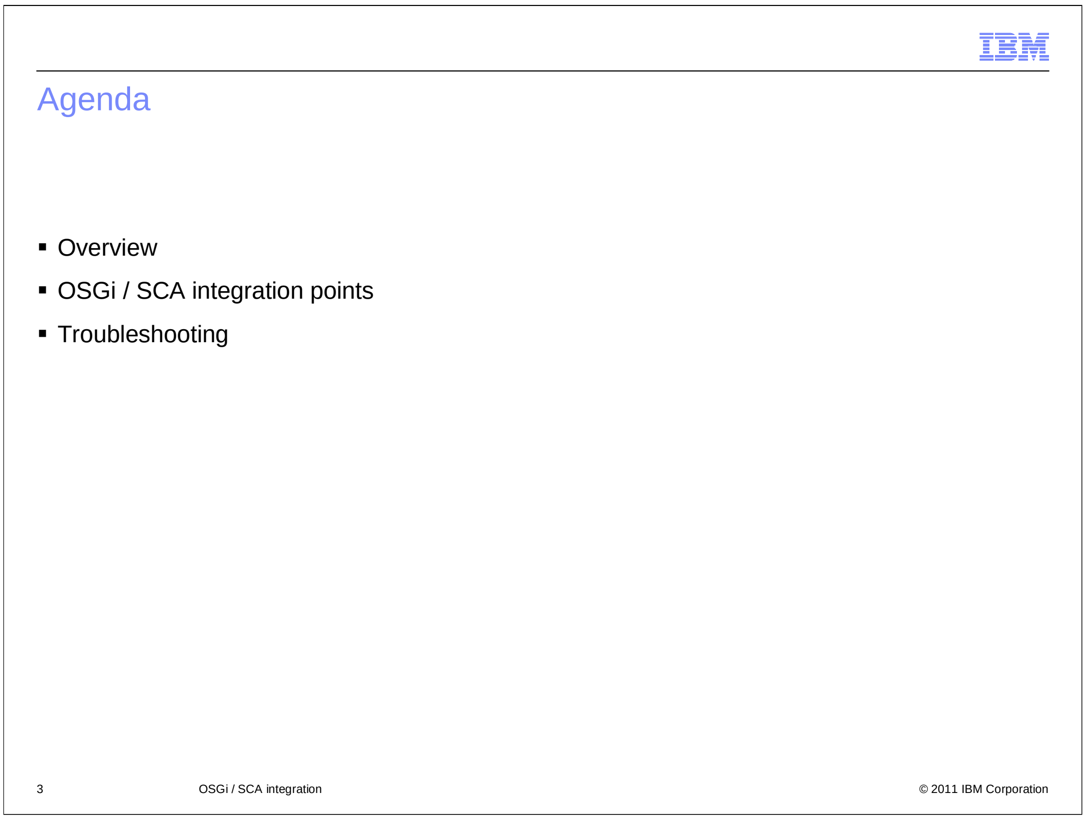


Read the documentation

- This talk provides an overview of SCA/OSGi integration.
- See the OSGi feature pack Information Center, and the SCA feature pack Information Center for more information

2 OSGi / SCA integration © 2011 IBM Corporation

This module is an overview of the SCA – OSGi integration but to take advantage of these features one should see the Information Centers of each.

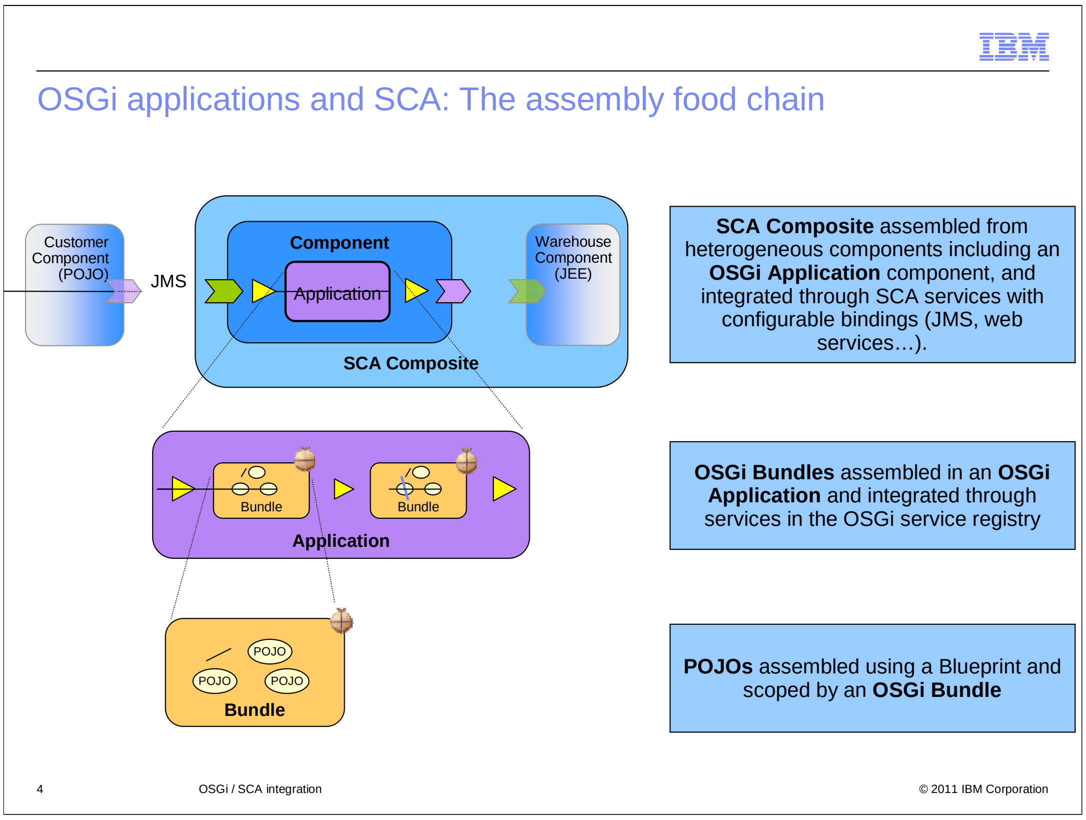


Agenda

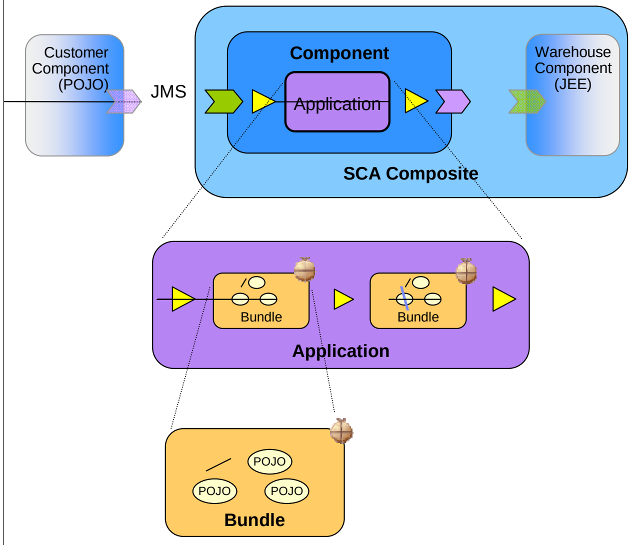
- Overview
- OSGi / SCA integration points
- Troubleshooting

3 OSGi / SCA integration © 2011 IBM Corporation

This module will provide a quick overview, outline the integration points between the two features, and provide some tips on troubleshooting.



OSGi applications and SCA: The assembly food chain



SCA Composite assembled from heterogeneous components including an **OSGi Application** component, and integrated through SCA services with configurable bindings (JMS, web services…).

OSGi Bundles assembled in an **OSGi Application** and integrated through services in the OSGi service registry

POJOs assembled using a Blueprint and scoped by an **OSGi Bundle**

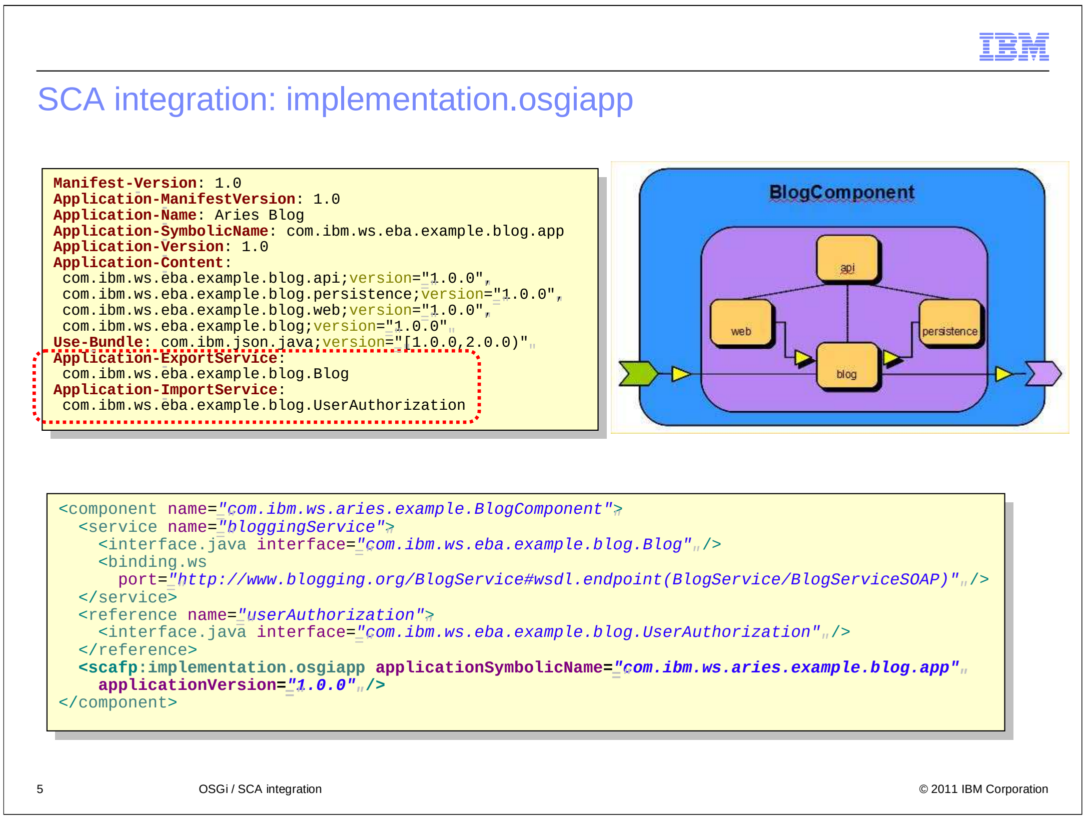
4 OSGi / SCA integration © 2011 IBM Corporation

Other modules in this series show how Plain Old Java Beans (POJOs) can be assembled and configured in a blueprint bundle and how multiple bundles including web and persistence bundles can be assembled into an isolated OSGi Application.

There is a further level of assembly available. OSGi applications may be represented as an SCA component. Within this composite the OSGi Application is a component that can be wired to other components with different implementation types.

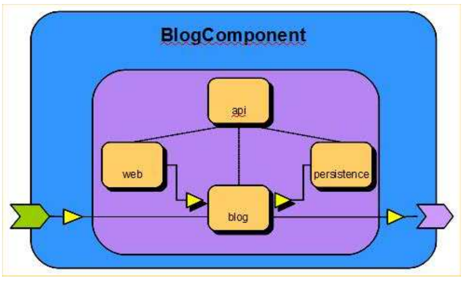
For example, an SCA composite can contain an OSGi-application component, a JEE component containing EJBs, a BPEL component and so on. Each component within an SCA composite declares abstract services and references to which concrete bindings can be applied. It is through these services and references that the components of an SCA composites are wired together. The OSGi Application architecture was designed with this form of assembly in mind so that the services and references declared in a Blueprint XML configuration can be exposed through the Application manifest to be visible outside the application. Such exposed services and references can then be mapped to SCA services and references with the full range of available SCA bindings applied to them.

This enables OSGi applications to participate in two new scenarios. The first one being assembly into heterogeneous composites of OSGi and non-OSGi components and the second one being remoting of OSGi application services through SCA services with a variety of bindings including JMS, SOAP/HTTP, IIOP and JSON-RPC.



SCA integration: implementation.osgiapp

```
Manifest-Version: 1.0
Application-ManifestVersion: 1.0
Application-Name: Aries Blog
Application-SymbolicName: com.ibm.ws.eba.example.blog.app
Application-Version: 1.0
Application-Content:
 com.ibm.ws.eba.example.blog.api;version="1.0.0",
 com.ibm.ws.eba.example.blog.persistence;version="1.0.0",
 com.ibm.ws.eba.example.blog.web;version="1.0.0",
 com.ibm.ws.eba.example.blog;version="1.0.0"
Use-Bundle: com.ibm.json.java;version="[1.0.0,2.0.0)"
Application-ExportService:
 com.ibm.ws.eba.example.blog.Blog
Application-ImportService:
 com.ibm.ws.eba.example.blog.UserAuthorization
```

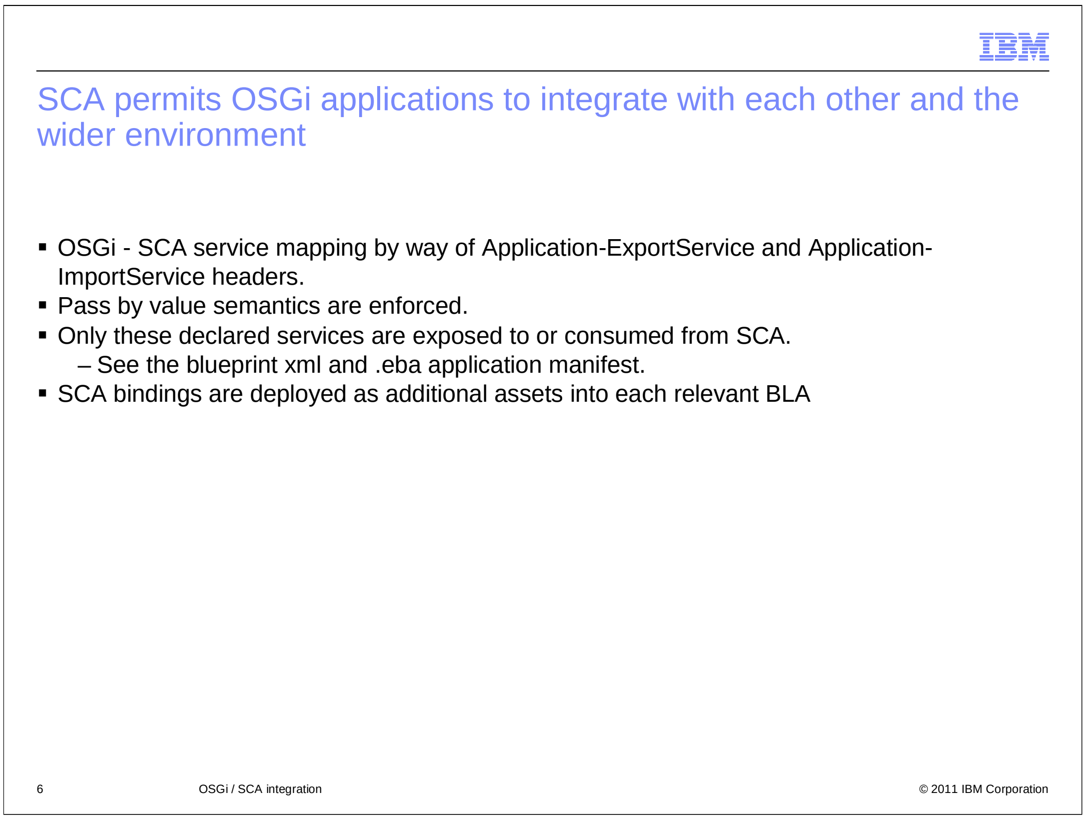


```
<component name="com.ibm.ws.aries.example.BlogComponent">
  <service name="bloggingService">
    <interface.java interface="com.ibm.ws.eba.example.blog.Blog"/>
    <binding.ws
      port="http://www.blogging.org/BlogService#wsdl.endpoint(BlogService/BlogServiceSOAP)"/>
  </service>
  <reference name="userAuthorization">
    <interface.java interface="com.ibm.ws.eba.example.blog.UserAuthorization"/>
  </reference>
  <scafp:implementation.osgiapp applicationSymbolicName="com.ibm.ws.aries.example.blog.app"
    applicationVersion="1.0.0"/>
</component>
```

5 OSGi / SCA integration © 2011 IBM Corporation

This slide shows an OSGi application packaged as an SCA component. Its application manifest in the top left exports a Blog service and imports a UserAuthorization service.

The xml shown at the bottom represents the SCA component declaration. Note how the application symbolic name and version appears in both, as well as the imported and exported services.



SCA permits OSGi applications to integrate with each other and the wider environment

- OSGi - SCA service mapping by way of Application-ExportService and Application-ImportService headers.
- Pass by value semantics are enforced.
- Only these declared services are exposed to or consumed from SCA.
  - See the blueprint xml and .eba application manifest.
- SCA bindings are deployed as additional assets into each relevant BLA

6 OSGi / SCA integration © 2011 IBM Corporation

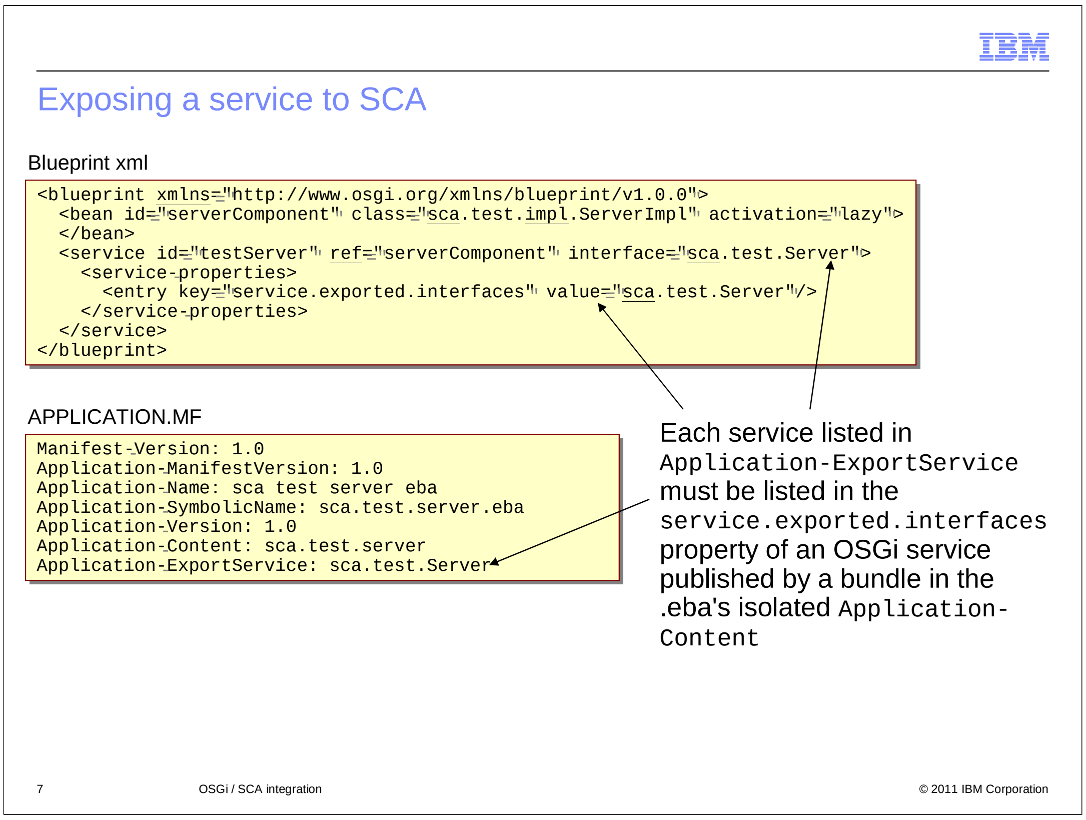
OSGi services are mapped to SCA services by way of the Application-ExportService and Application-ImportService headers in the .eba application manifest, META-INF/APPLICATION.MF.

OSGi applications can communicate with each other, or any other SCA composites by way of this mechanism.

SCA can integrate distributed components, and so enforces pass by value semantics.

The OSGi application developer needs to know up front which services and references will be exposed to SCA. This knowledge is reflected in the blueprint xml and the .eba application manifest.

SCA bindings are written as .composite files, packaged into .jar files and deployed as additional assets into each relevant BLA.



Exposing a service to SCA

Blueprint xml

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <bean id="serverComponent" class="sca.test.impl.ServerImpl" activation="lazy">
  </bean>
  <service id="testServer" ref="serverComponent" interface="sca.test.Server">
    <service-properties>
      <entry key="service.exported.interfaces" value="sca.test.Server"/>
    </service-properties>
  </service>
</blueprint>
```

APPLICATION.MF

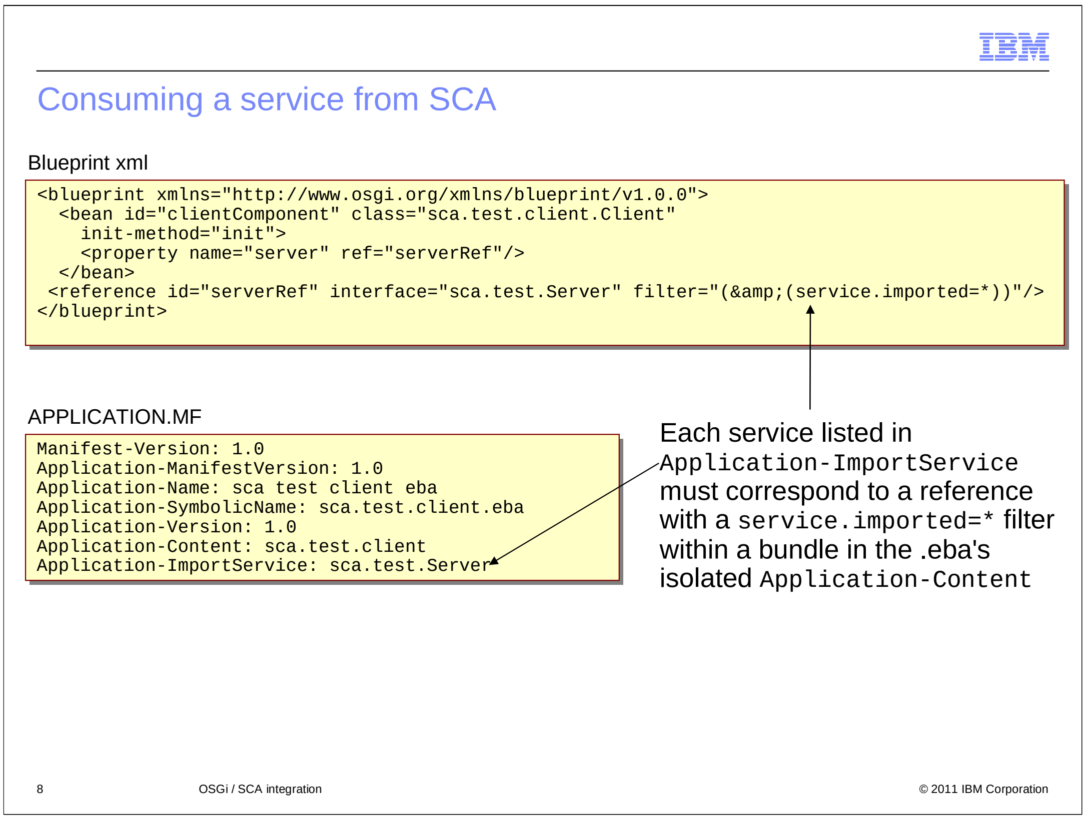
```
Manifest-Version: 1.0
Application-ManifestVersion: 1.0
Application-Name: sca test server eba
Application-SymbolicName: sca.test.server.eba
Application-Version: 1.0
Application-Content: sca.test.server
Application-ExportService: sca.test.Server
```

Each service listed in `Application-ExportService` must be listed in the `service.exported.interfaces` property of an OSGi service published by a bundle in the .eba's isolated `Application-Content`

7 OSGi / SCA integration © 2011 IBM Corporation

This slide shows a service declared in blueprint xml and exported to SCA by way of the application manifest. Note that each exported service must have a service.exported.interfaces property with at least one interface as its value.

By default, OSGi services use pass-by-reference semantics. By adding the service.exported.interfaces property, the developer indicates that these interfaces are intended to be exposed remotely, and that they follow pass-by-value semantics instead.



Consuming a service from SCA

Blueprint xml

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <bean id="clientComponent" class="sca.test.client.Client"
    init-method="init">
    <property name="server" ref="serverRef"/>
  </bean>
 <reference id="serverRef" interface="sca.test.Server" filter="(&amp;(service.imported=*))"/>
</blueprint>
```

APPLICATION.MF

```
Manifest-Version: 1.0
Application-ManifestVersion: 1.0
Application-Name: sca test client eba
Application-SymbolicName: sca.test.client.eba
Application-Version: 1.0
Application-Content: sca.test.client
Application-ImportService: sca.test.Server
```

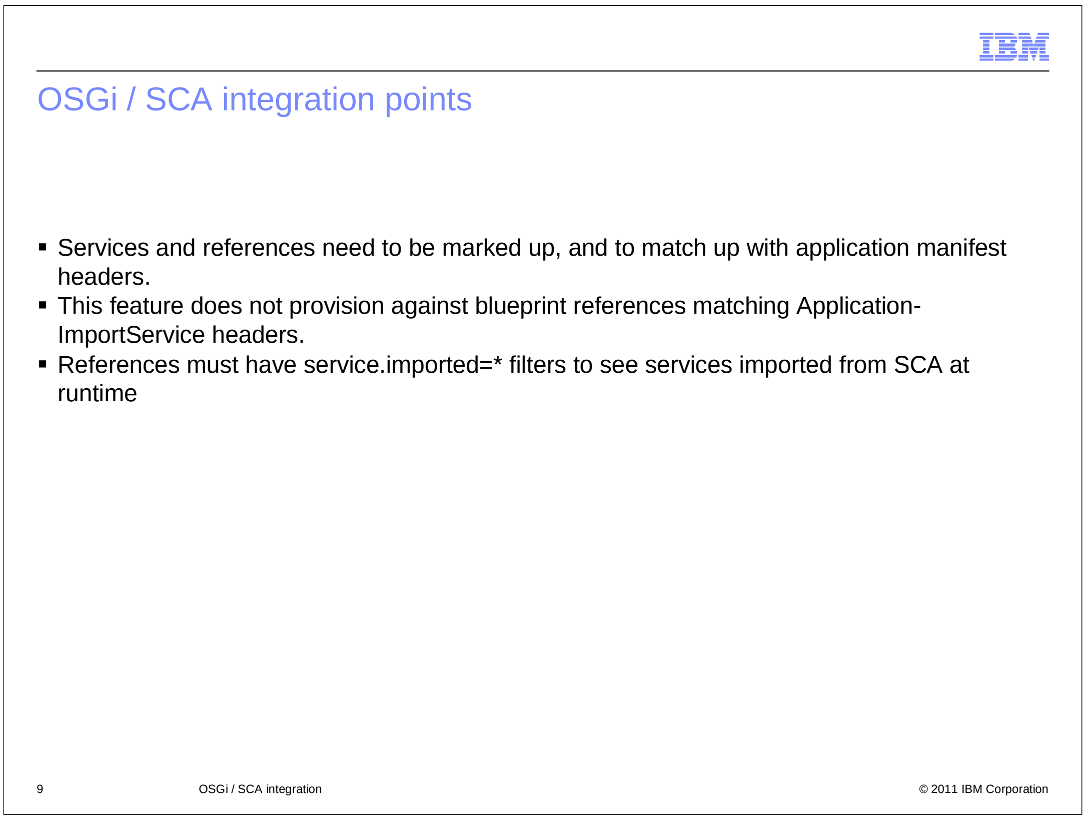
Each service listed in `Application-ImportService` must correspond to a reference with a `service.imported=*` filter within a bundle in the .eba's isolated `Application-Content`

8 OSGi / SCA integration © 2011 IBM Corporation

Here's the corresponding consumption of a service from SCA.

The service proxy representing an SCA reference advertises itself as a remote service by using the service property service.imported=true. This allows consumers to distinguish local services, which use pass-by-reference semantics, from remote services, which use pass-by-value semantics.

By default, a blueprint reference will not find remote services. It must request them using a filter, as shown above. In this way the developer of the consuming application indicates their awareness that these services are remote and use pass-by-value semantics.



OSGi / SCA integration points

- Services and references need to be marked up, and to match up with application manifest headers.
- This feature does not provision against blueprint references matching Application-ImportService headers.
- References must have service.imported=* filters to see services imported from SCA at runtime

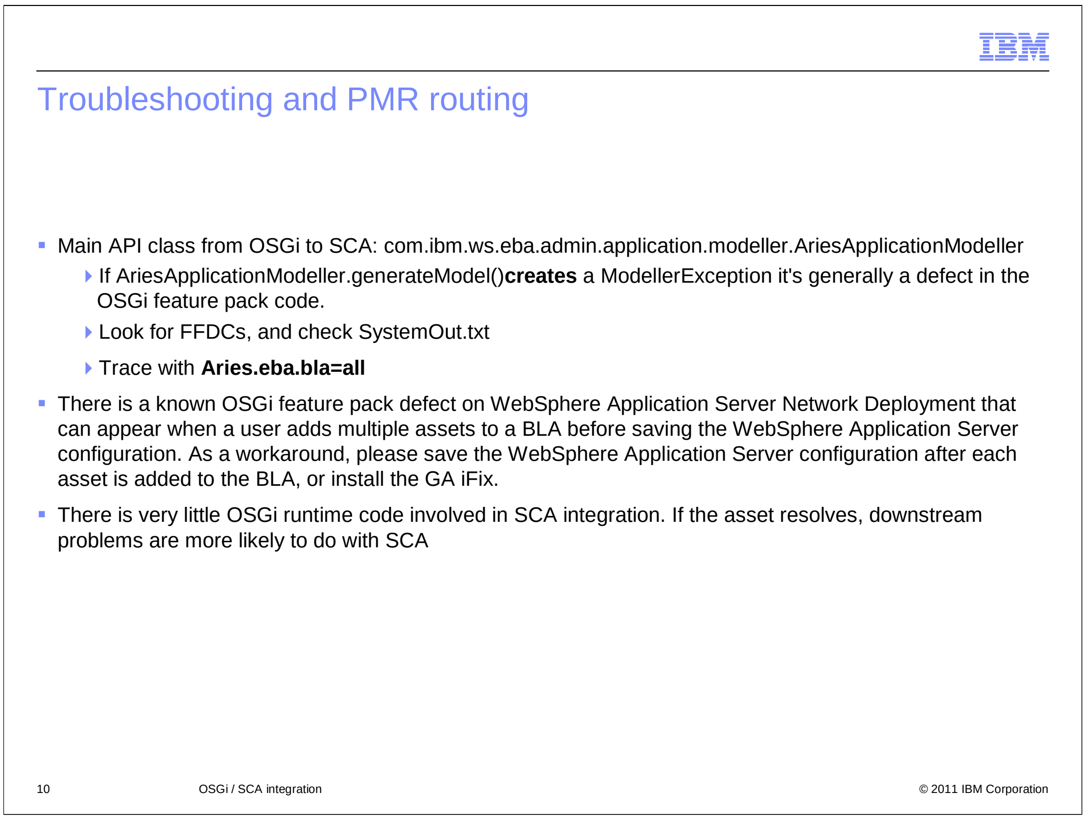
9 OSGi / SCA integration © 2011 IBM Corporation

Services and references need to be marked up, and to match up with application manifest headers.

Misconfigured applications will frequently fail to resolve, preventing .eba asset import.

When an .eba application is resolved, services are not provisioned against blueprint references matching Application-ImportService headers.

SCA proxies representing imported services are hidden at runtime from references lacking service.imported=* filters.



Troubleshooting and PMR routing

- Main API class from OSGi to SCA: com.ibm.ws.eba.admin.application.modeller.AriesApplicationModeller
  - If AriesApplicationModeller.generateModel()**creates** a ModellerException it's generally a defect in the OSGi feature pack code.
  - Look for FFDCs, and check SystemOut.txt
  - Trace with **Aries.eba.bla=all**
- There is a known OSGi feature pack defect on WebSphere Application Server Network Deployment that can appear when a user adds multiple assets to a BLA before saving the WebSphere Application Server configuration. As a workaround, please save the WebSphere Application Server configuration after each asset is added to the BLA, or install the GA iFix.
- There is very little OSGi runtime code involved in SCA integration. If the asset resolves, downstream problems are more likely to do with SCA

10 OSGi / SCA integration © 2011 IBM Corporation

The main API class exposed by the OSGi to the SCA feature pack is com.ibm.ws.eba.admin.application.modeller.AriesApplicationModeller.

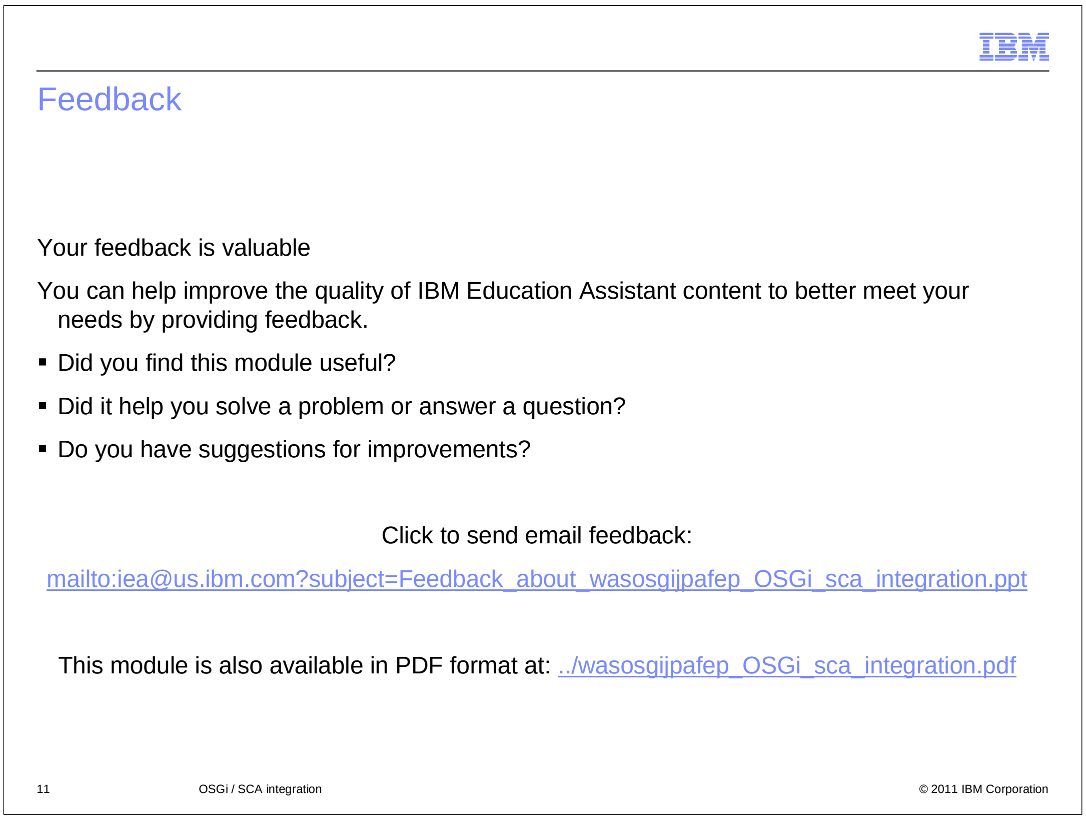
Calls to AriesApplicationModeller.generateModel() that result in ModellerException being thrown generally indicate a defect in the OSGi feature pack code.

Such exceptions will appear as FFDCs, and likely also in SystemOut.txt.

Trace with Aries.eba.bla=all would be very helpful in this case.

There is currently no evidence that there are any such defects.

At GA of the feature pack, there are known OSGi feature pack defects that can appear when a user adds multiple assets to a BLA, including at least one .eba asset, before saving WebSphere Application Server configuration. As a workaround, save the WebSphere Application Server configuration after each asset is added to the BLA.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

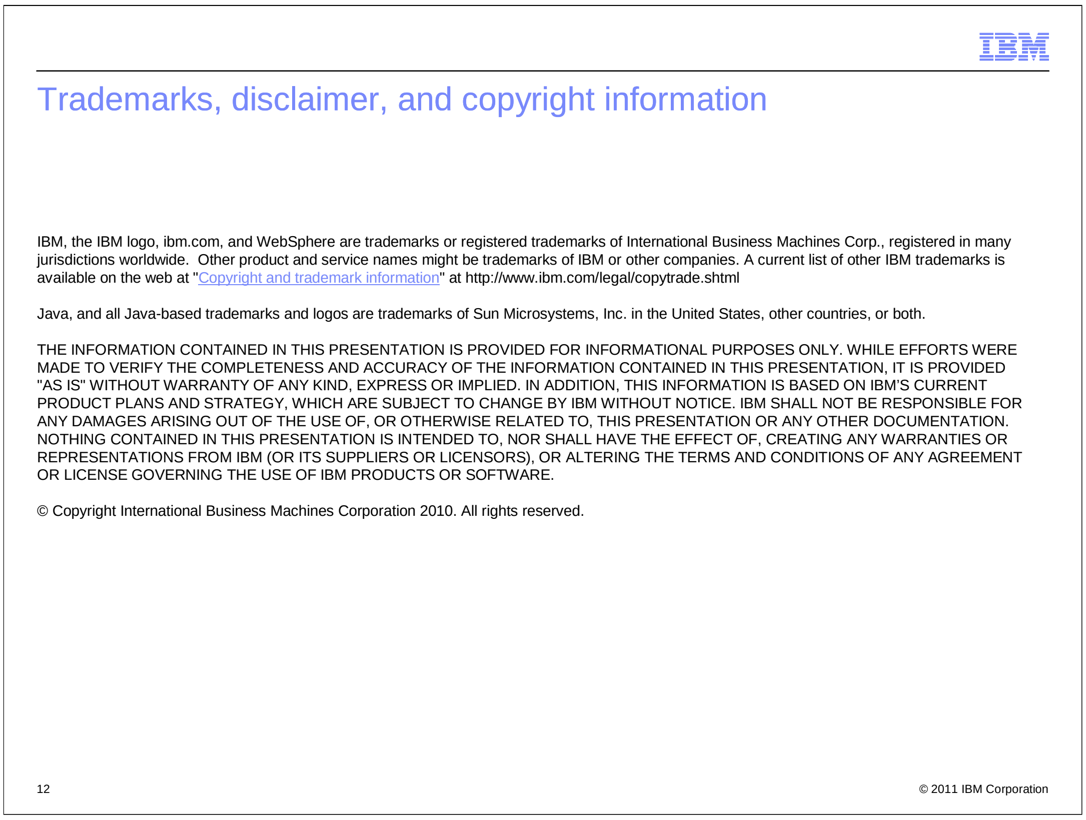
Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_wasosgijpafep_OSGi_sca_integration.ppt

This module is also available in PDF format at: ../wasosgijpafep_OSGi_sca_integration.pdf

11 OSGi / SCA integration © 2011 IBM Corporation

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, disclaimer, and copyright information