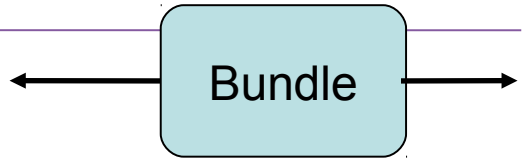
WebSphere Application Server v7 OSGi Application Best Practices

Emily Jiang, WebSphere Application Server OSGi Developer

Agenda

- OSGi general best practices
- WebSphere Application Server specific OSGi best Practices

OSGi Bundles and Class Loading



OSGi Bundle – A jar containing:

- Classes and resources.
- OSGi Bundle manifest.

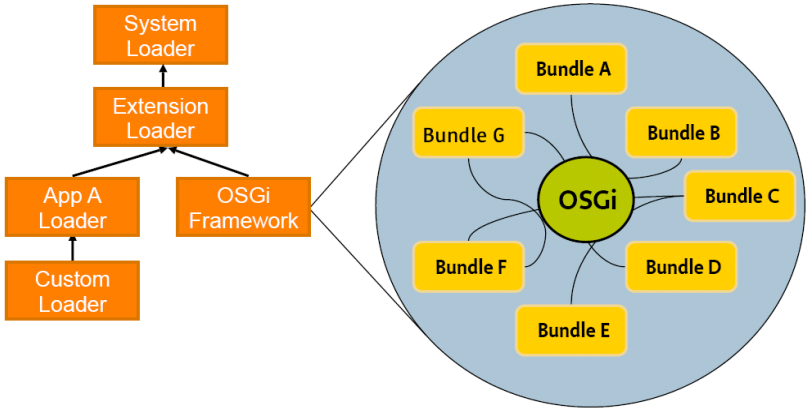
What's in the manifest:

- Bundle-Version: Multiple versions of bundles can live concurrently.
- Import-Package: What packages from other bundles does this bundle depend upon?
- Export-Package: What packages from this bundle are visible and reusable outside of the bundle?

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: MyService bundle
Bundle-SymbolicName: com.sample.myservice
Bundle-Version: 1.0.0
Bundle-Activator: com.sample.myservice.Activator
Import-Package: com.something.i.need;version="1.1.2"
Export-Package: com.myservice.api;version="1.0.0"
```

Class Loading

- Each bundle has its own loader.
- No flat or monolithic classpath.
- Class sharing and visibility decided by declarative dependencies, not by class loader hierarchies.
- OSGi framework works out the dependencies including versions.



BP1 - Use Import-Package not Require-Bundle

Require-Bundle

- Tightly coupled with a particular bundle with the specified symbolic name and version
- High coupling between bundles
- Import all packages
- Bundle version management

MANIFEST.MF

...

Require-Bundle: com.ibm.ws.service;bundle-version=2.0.0

- Import-Package
  - Can wire to any bundles exporting the specified package
  - Loose coupling between bundles
  - Only import the package you need
  - Package version management

MANIFEST.MF

...

Import-Package: com.ibm.ws.service.api;version=2.0.0

BP2 - Avoid split packages

- Split package
  - A package is exported by two bundles at the same version and the set of classes provided by each bundle differs.
- Why?
  - Leads to the use of Require-Bundle, compromising the extent to which systems using the bundles can be extended and maintained.
- How?
  - Keep all of the classes from any one package in a single bundle

BP2 - Split Package examples

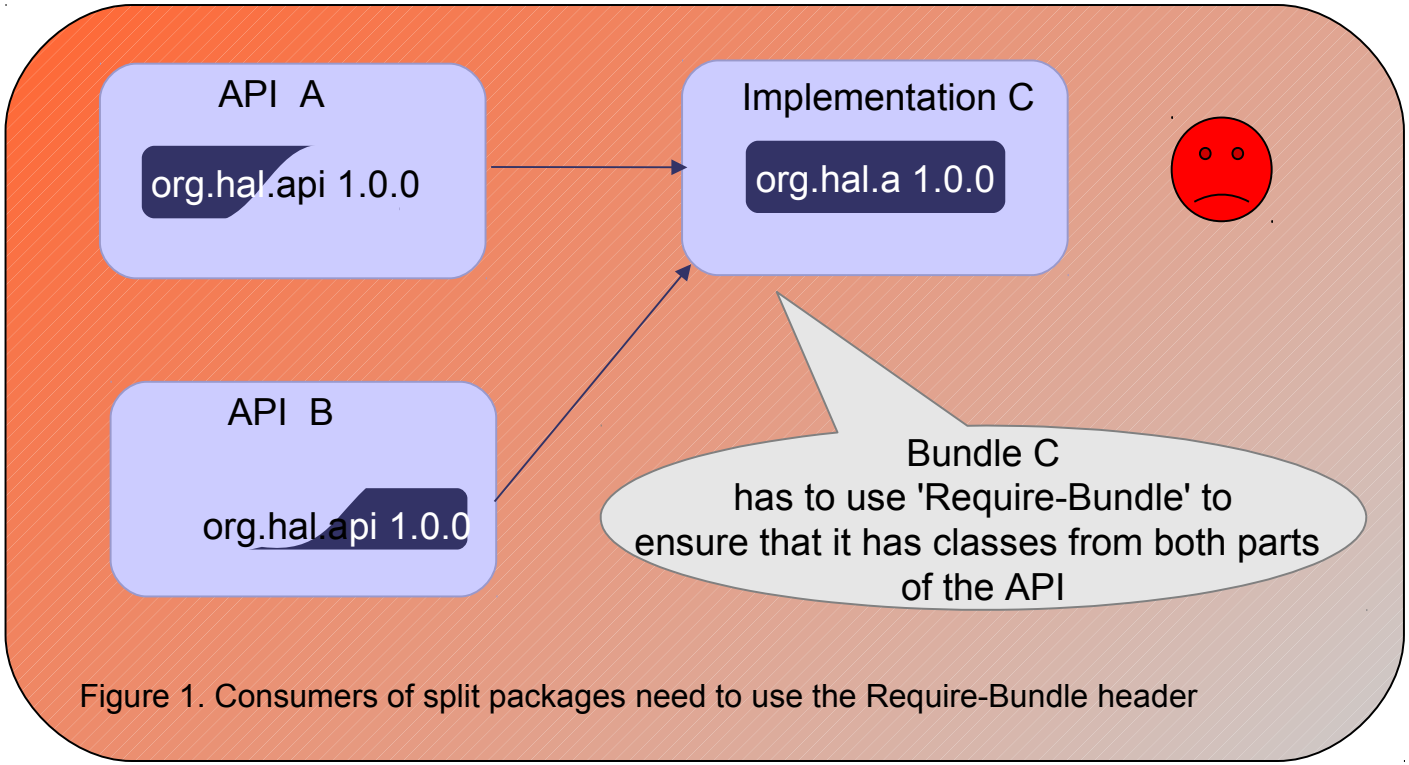


Figure 1. Consumers of split packages need to use the Require-Bundle header

BP2 - Split Package examples

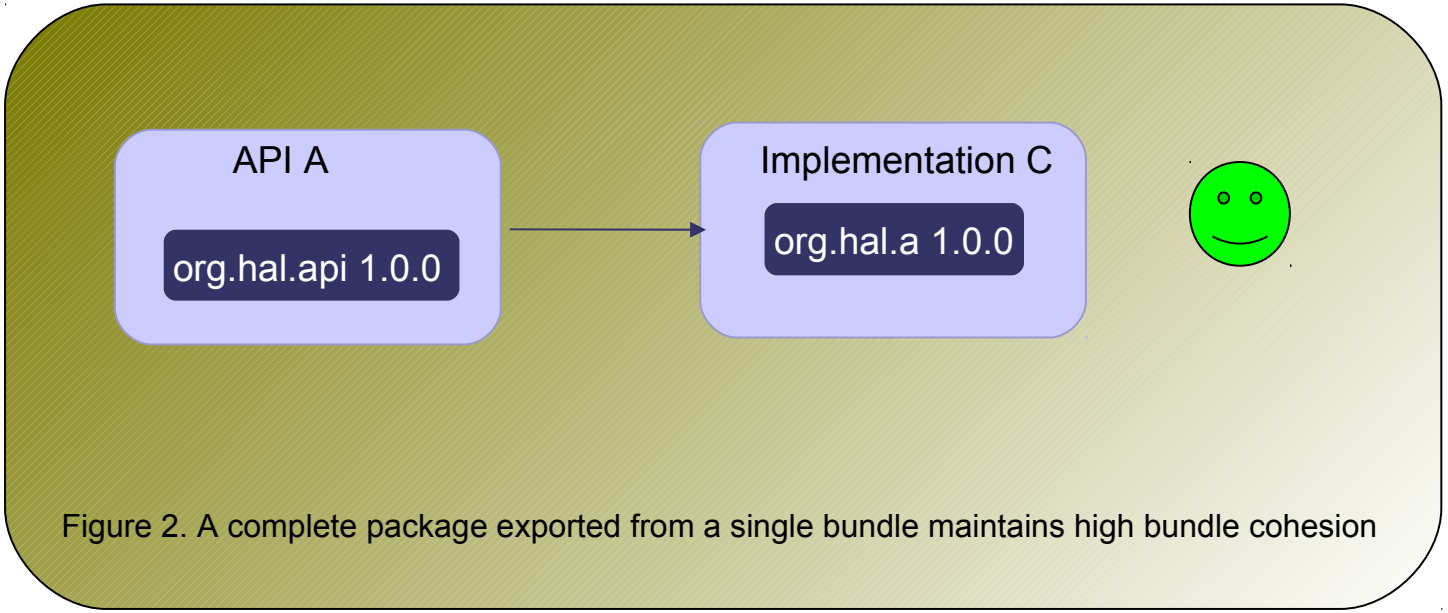


Figure 2. A complete package exported from a single bundle maintains high bundle cohesion

BP3 - Version bundles and packages

- What is semantic versioning?
  - Uses a major.minor.micro.qualifier numbering scheme
    - Major - Packages with versions that have different major parts are not compatible both for providers as well as consumers.
    - Minor – Backward compatible with the same major value of the same package
    - Micro – bug fixing
    - Qualifier – identifier such as timestamp
  - Changes in major: a binary incompatible, minor: enhanced API, micro: no API changes

- Why?
  - Clients can protect themselves against API changes that might break them.

BP3 - Version bundles and packages examples

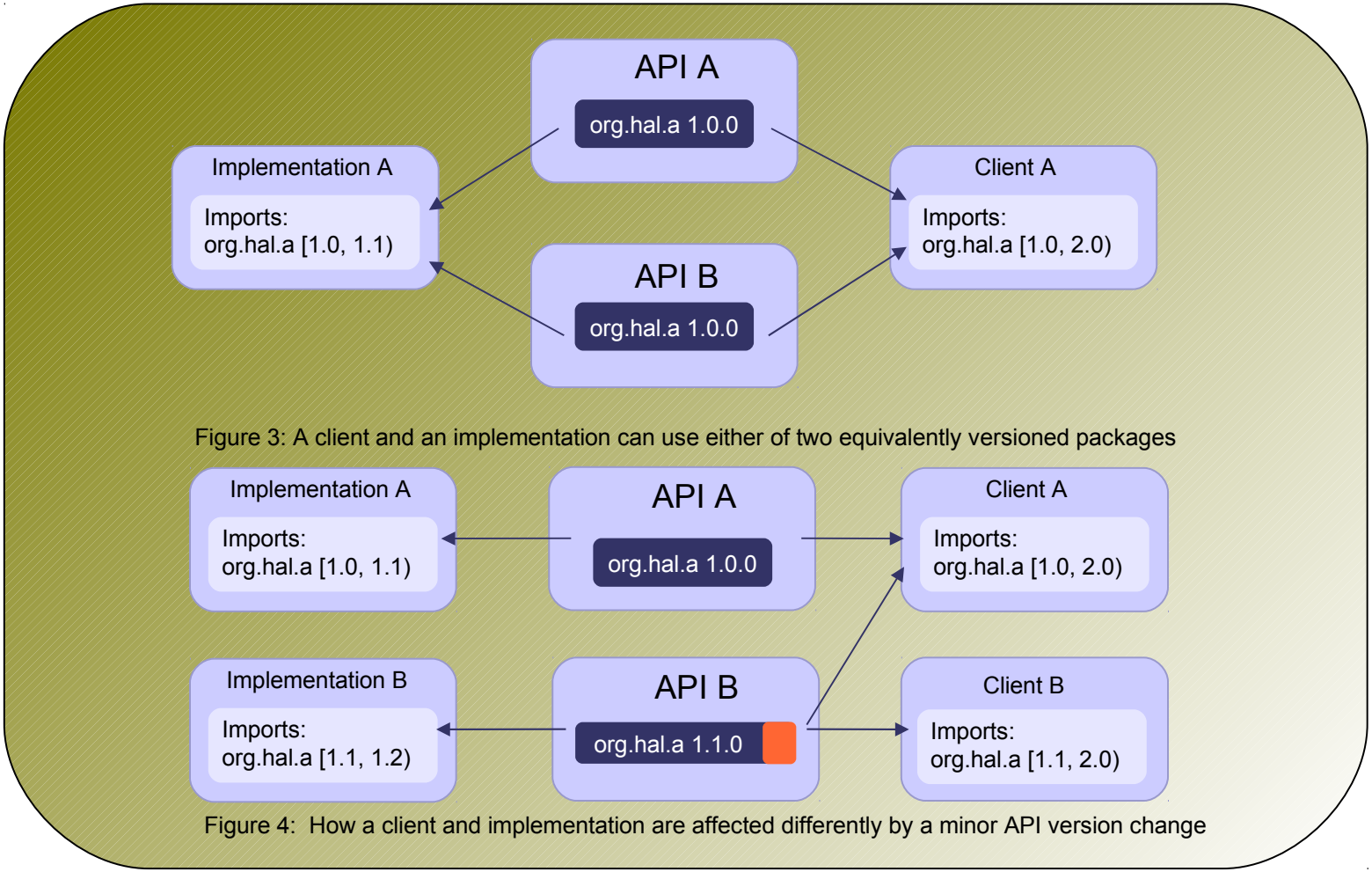


Figure 3: A client and an implementation can use either of two equivalently versioned packages

Figure 4: How a client and implementation are affected differently by a minor API version change

BP3 - Version bundles and packages examples

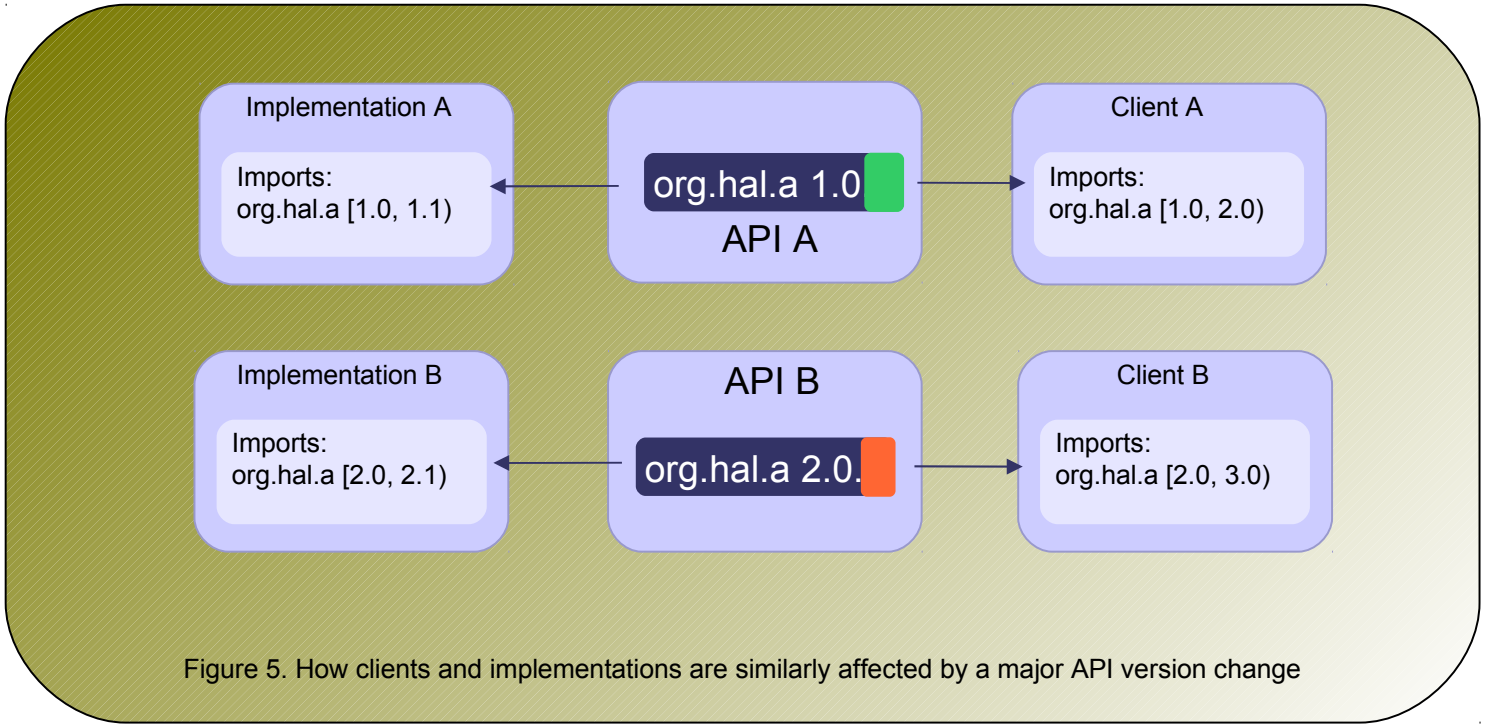


Figure 5. How clients and implementations are similarly affected by a major API version change

BP4 - Separate API from Implementations

- Why?
  - Great flexibility
  - Many implementation bundles → enable more services provided
  - Reduce package dependencies → reduce circular dependencies
- How?
  - Put API classes in one bundle
  - Put implementation classes in a separate bundle

BP4 - Separate API and implementation examples

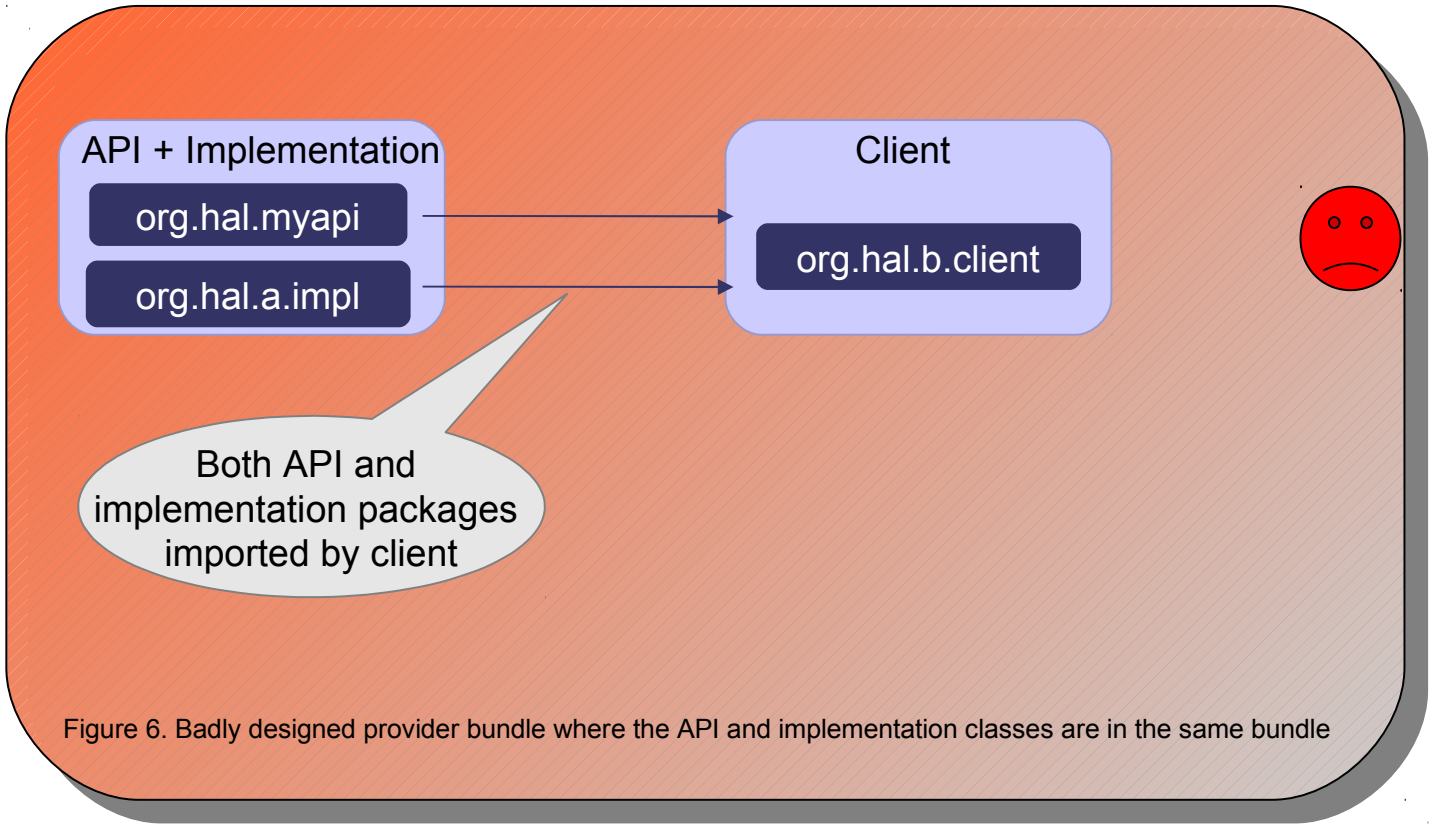
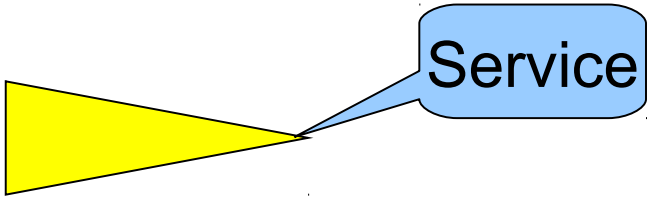


Figure 6. Badly designed provider bundle where the API and implementation classes are in the same bundle

BP5 - Share services not implementations

- Use the OSGi service registry to construct instances
- Why?
  - Able to obtain an instance of an implementation without knowing which one
  - Achieve a loosely coupling of client, API and implementation
- How?



  - Register an instance of the API interface in the OSGi service registry
  - Register an implementation of the OSGi ServiceFactory interface in the OSGi service registry.

BP4 & 5 - examples

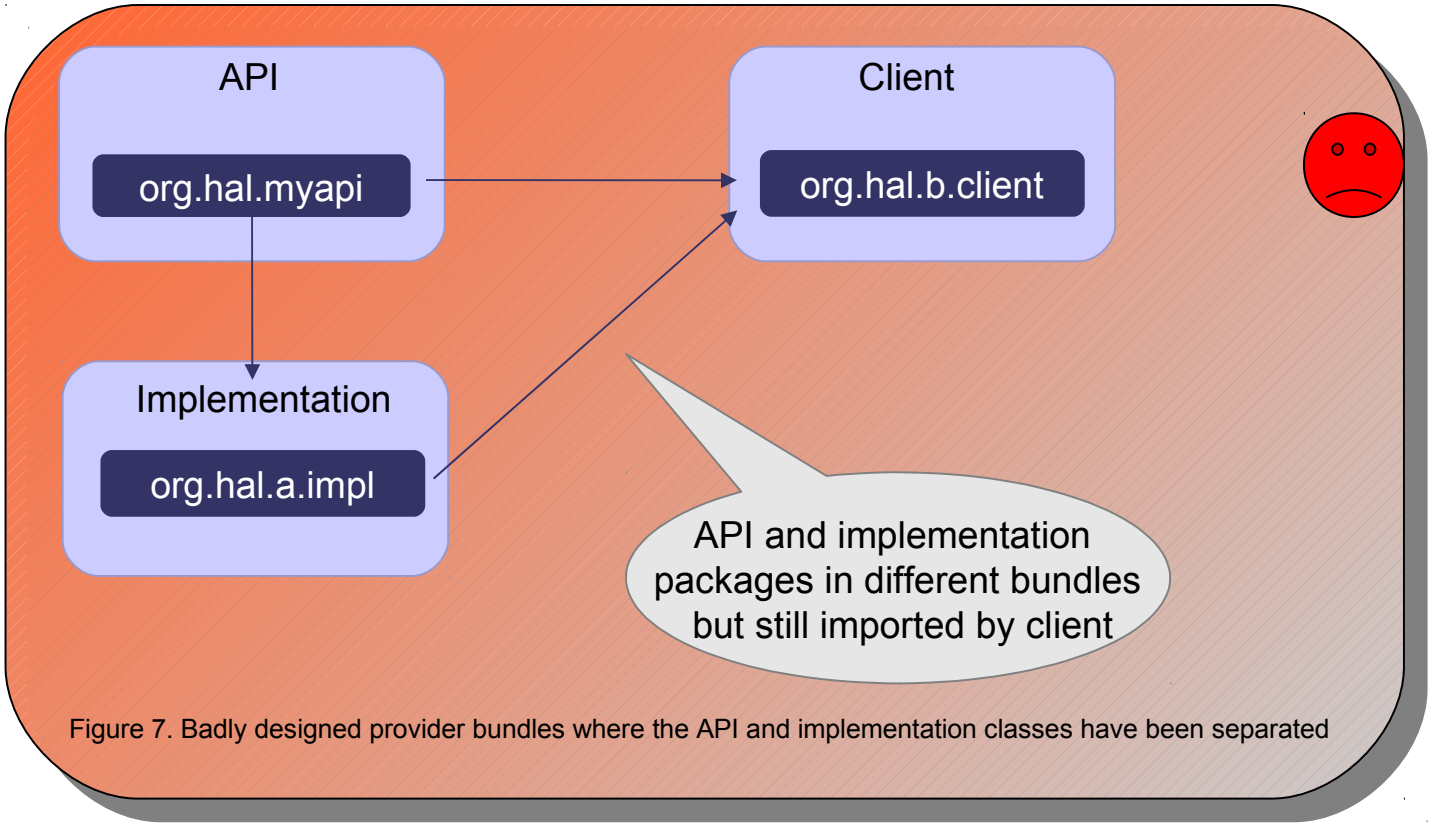


Figure 7. Badly designed provider bundles where the API and implementation classes have been separated

BP4 & 5 - examples

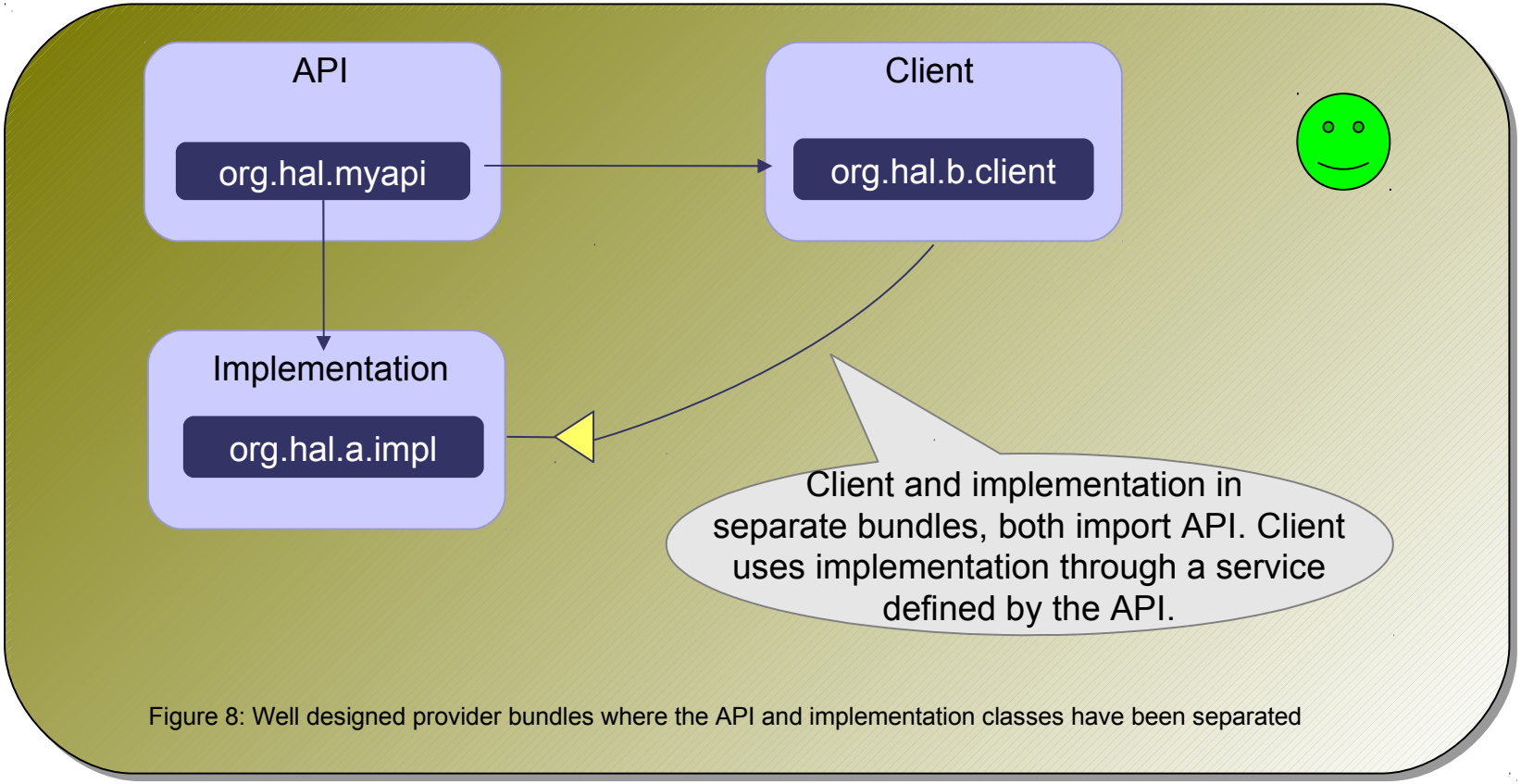


Figure 8: Well designed provider bundles where the API and implementation classes have been separated

BP6 – Make Bundles Loosely Coupled & Highly Cohesive

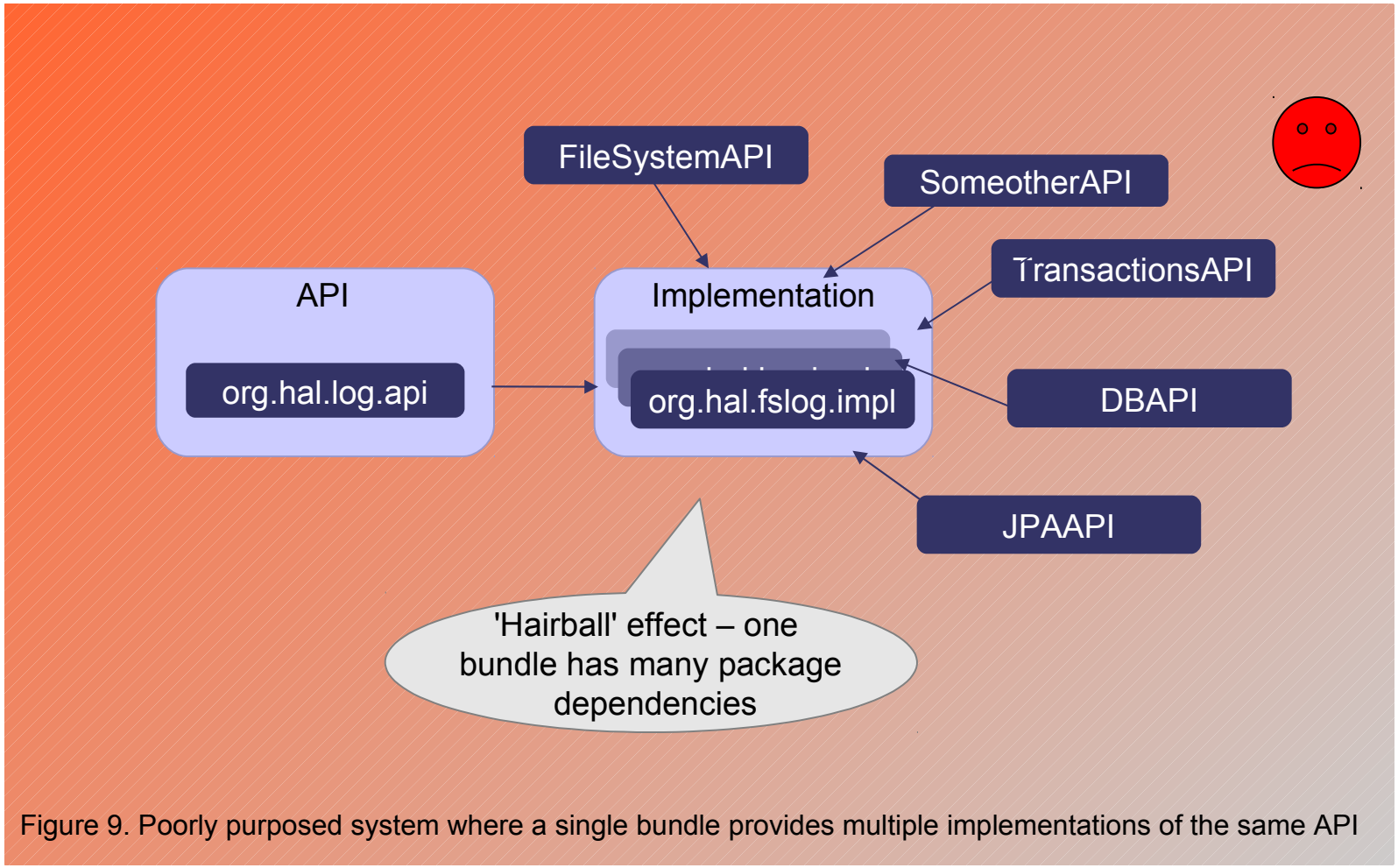


Figure 9. Poorly purposed system where a single bundle provides multiple implementations of the same API

BP6 – Make Bundles Loosely Coupled & Highly Cohesive

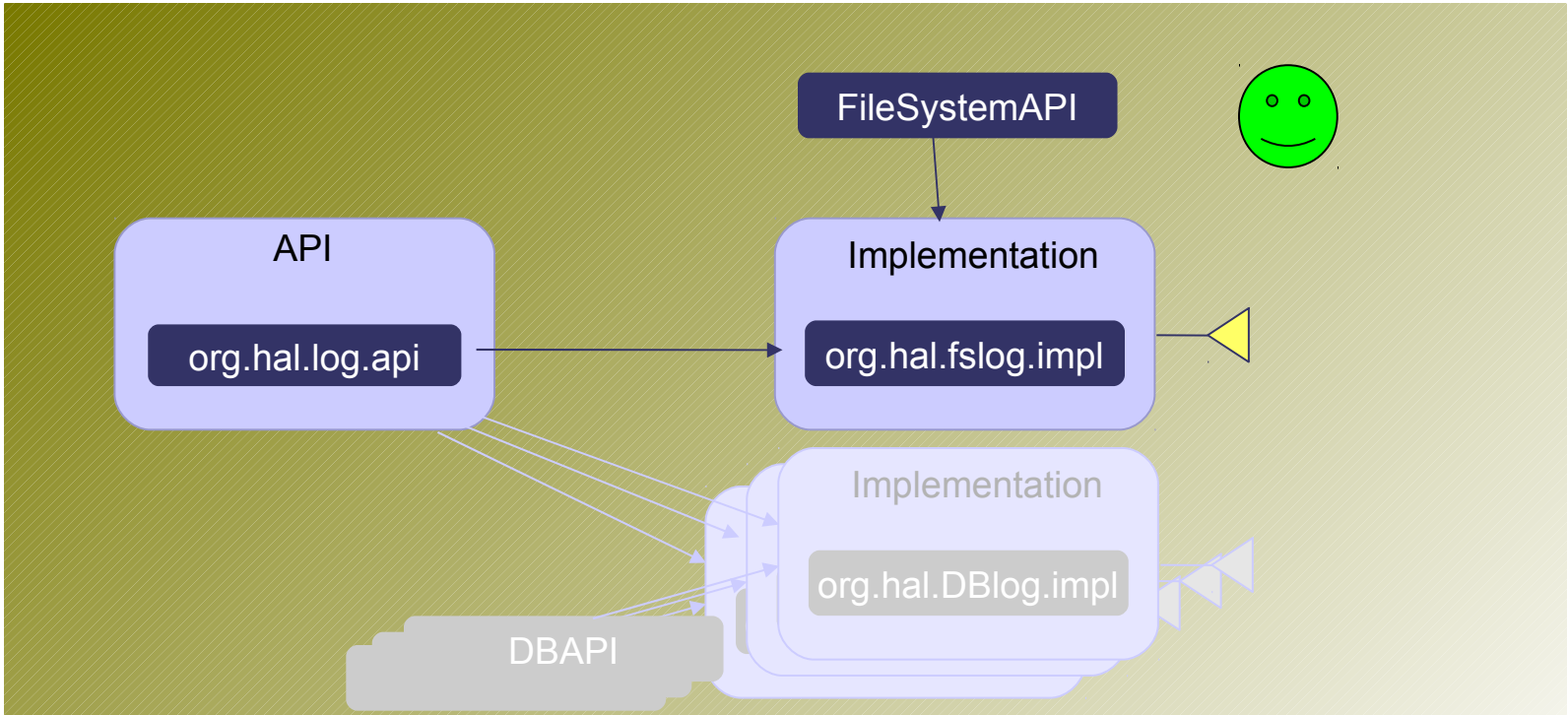


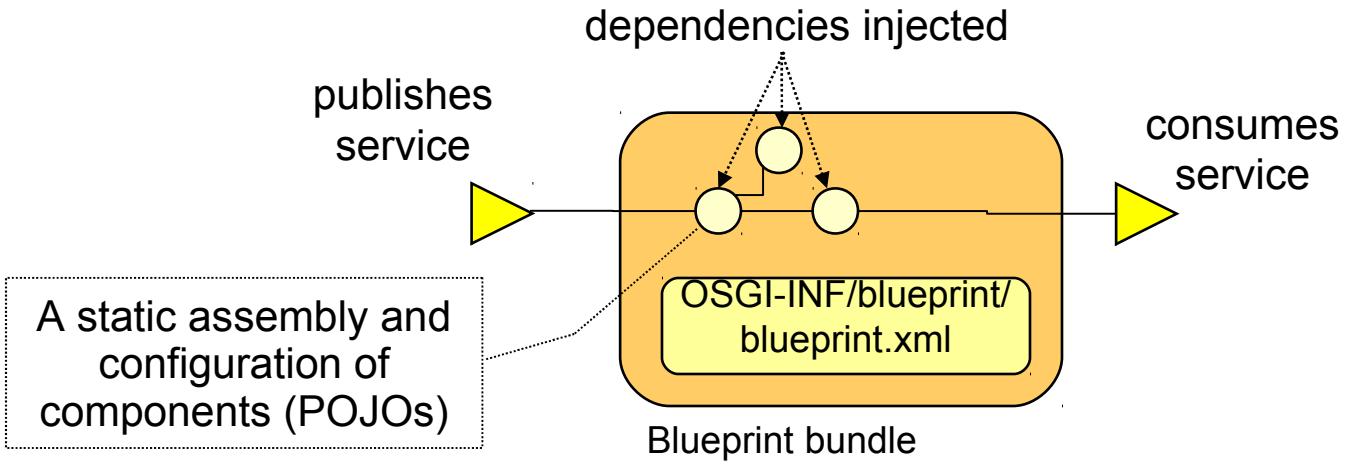
Figure 10. A well purposed system where each implementation of an API is provided by a separate bundle

Using services can be hard!

```
private BundleContext ctx;


private AtomicReference<LogService> ls = new
AtomicReference<LogService>();
private AtomicReference<ServiceReference> lr = new
AtomicReference<ServiceReference>();


public void start(BundleContext ctx) throws InvalidSyntaxException
{
this.ctx = ctx;
ctx.addServiceListener(this,
"(objectClass=org.osgi.service.log.LogService)");
ServiceReference ref =
ctx.getServiceReference(LogService.class.getName());
if (ref != null) {
ls.set((LogService) ctx.getService(ref));
lr.set(ref);
}
}
```

```
@Override
public void serviceChanged(ServiceEvent event)
{
ServiceReference ref = event.getServiceReference();


if (ls.get() == null && event.getType() == ServiceEvent.REGISTERED) {
ls.set((LogService) ctx.getService(ref));
} else if (ls.get() != null && event.getType() ==
        ServiceEvent.UNREGISTERING &&
ref == lr.get()) {
ref = ctx.getServiceReference(LogService.class.getName());
if (ref != null) {
ls.set((LogService) ctx.getService(ref));
lr.set(ref);
}
}
}
```

BP7 - Use Blueprint

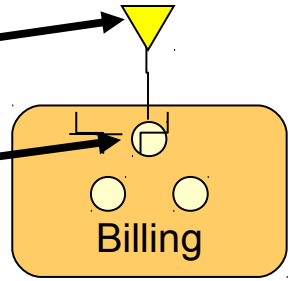


- Specifies a Dependency Injection container, standardizing established Spring conventions
- Configuration and dependencies declared in XML “module blueprint”, which is a standardization of Spring “application context” XML.
  - Extended for OSGi: publishes and consumes components as OSGi services
- Simplifies unit test outside either Java EE or OSGi r/t.
- **The Blueprint DI container is a part of the server runtime (compared to the Spring container which is part of the application.)**

BP7 - Blueprint service-bundle examples

Billing service bundle

```
<blueprint>
  <service ref="service" interface =
        "org.example.bill.BillingService" />
  <bean id="service" scope="prototype"
    class="org.example.bill.impl.BillingServiceImpl" />
</blueprint>
```



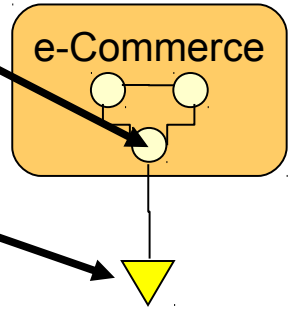
-"prototype" scope indicates a new instance is created by the container for each use.
-"singleton" scope is the default.

```
public interface BillingService {
    void bill(Order o);
}
```

BP7- Blueprint client-bundle examples

e-Commerce bundle

```
<blueprint>
  <bean id="shop" class="org.example.ecomm.ShopImpl">
    <property name="billingService" ref="billingService" />
  </bean>
  <reference id="billingService"
    interface="org.example.bill.BillingService" />
</blueprint>
```




```
public class ShopImpl {

private BillingService billingService;
void setBillingService(BillingService srv) {
    billingService = srv;
    }

void process(Order o) {
    billingService.bill(o);
    }
}
```

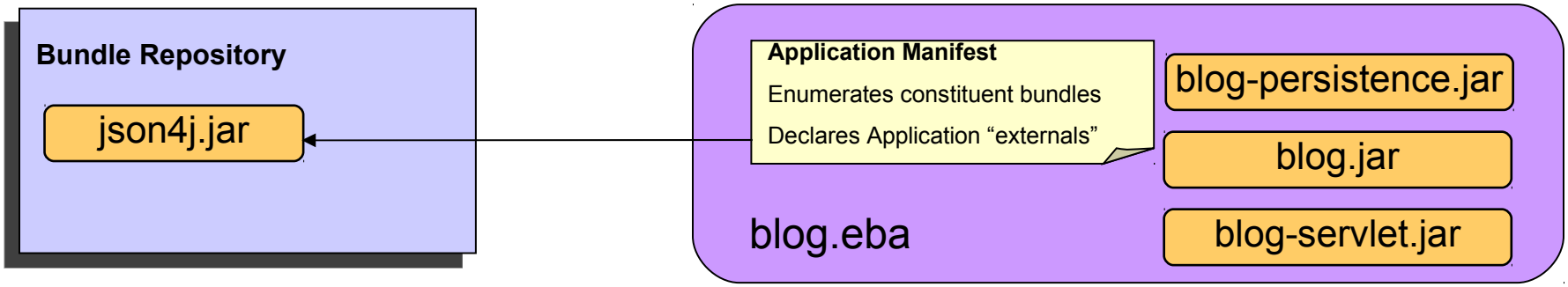
-injected service reference
-service can change over time
-can be temporarily absent
  without the bundle caring
-managed by Blueprint container

Agenda

- OSGi general best practices
- WebSphere Application Server Specific OSGi best Practices

New: “Enterprise Bundle Archive” (EBA)

- An isolated, cohesive application consisting of a collection of bundles, is deployed as a logical unit in a “.eba” archive
  - An “OSGi Application”.
- Constituent bundles may be contained (“by-value”) or referenced from a bundle repository.
- Services provided by the application are isolated to the application unless explicitly exposed through EBA-level application manifest
- Config by exception - absence of APPLICATION.MF means:
  - application content is the set of bundles contained by-value plus any repository-hosted dependencies identified during deployment.



BP8 - Use blueprint to enable service-based provisioning

- Use Blueprint to determine the service a bundle provides and requires.
- Use Blueprint during deployment to provision implementation bundles from the application archive or bundle repositories.

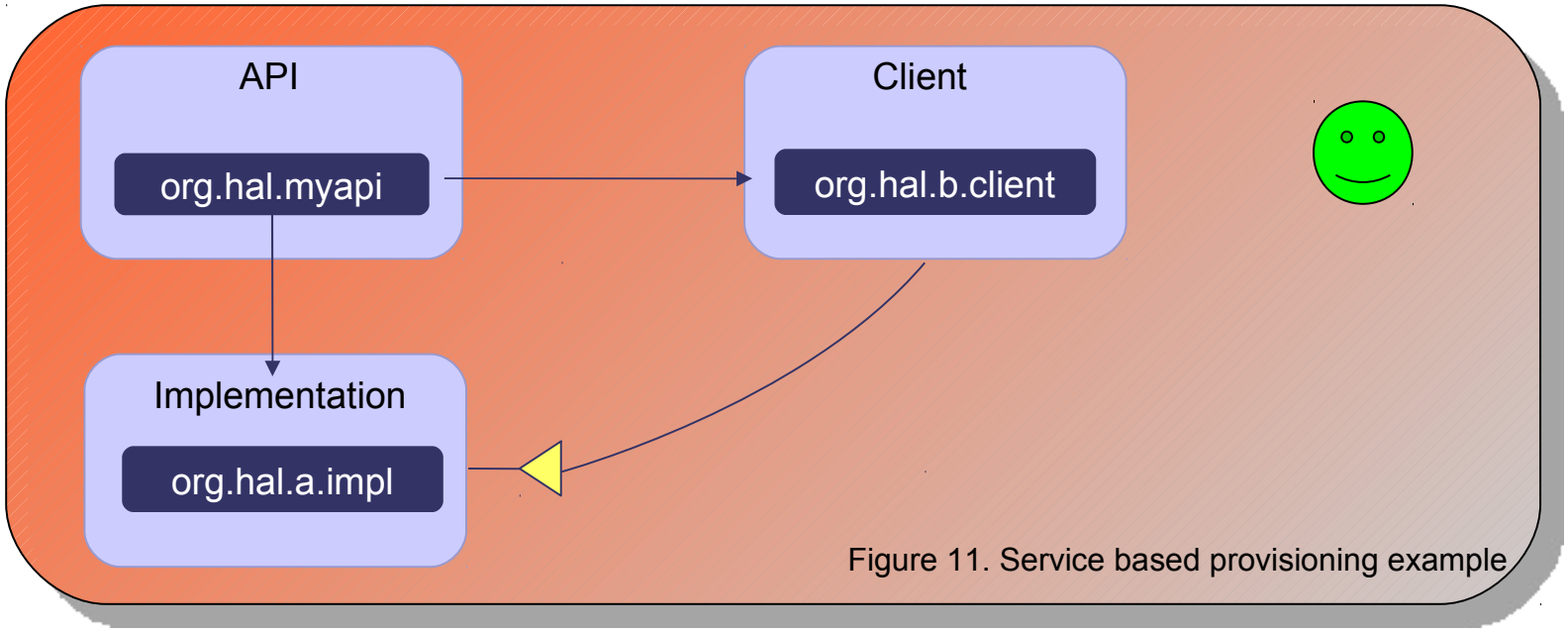


Figure 11. Service based provisioning example

BP9 - Use blueprint to enable SCA integration

- What is SCA (Service Component Architecture)?
  - Assemble OSGi applications with other application types (for example, Java EE).
  - Expose OSGi application services via various transports and protocols and to enable service dependencies to call out via those transports and protocols
- How to interact with SCA?
  - Use Blueprint to expose services to SCA
  - A blueprint façade can be used to describe non-blueprint bundle services to SCA

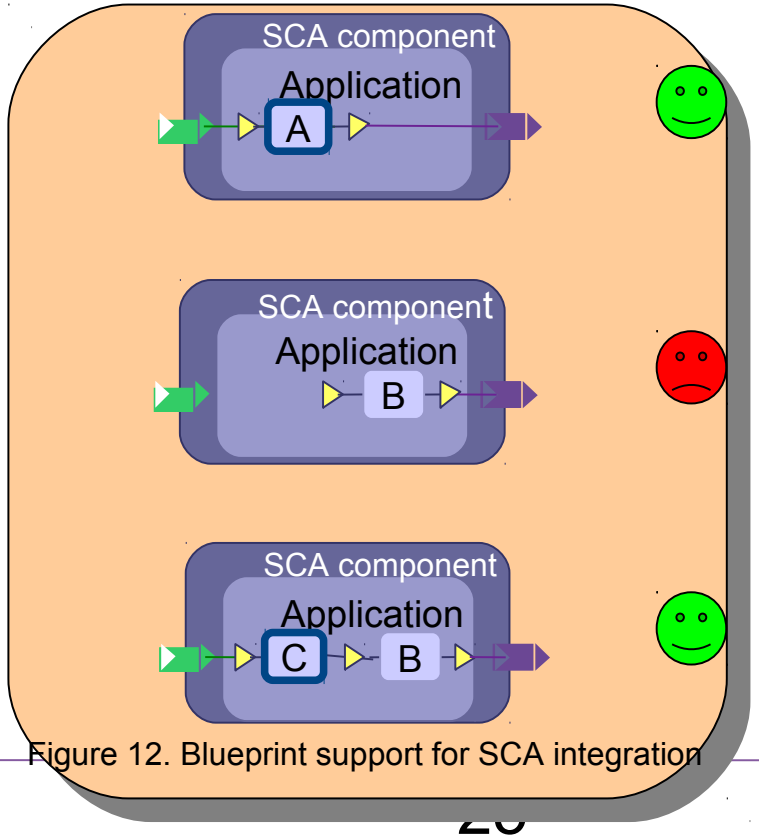
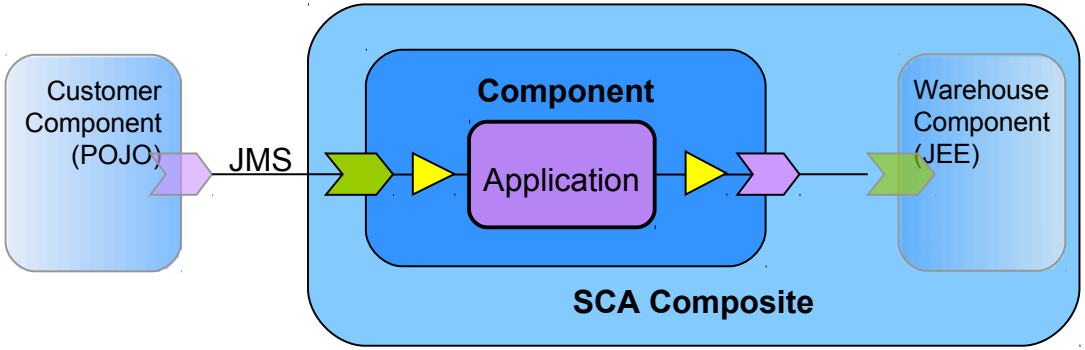
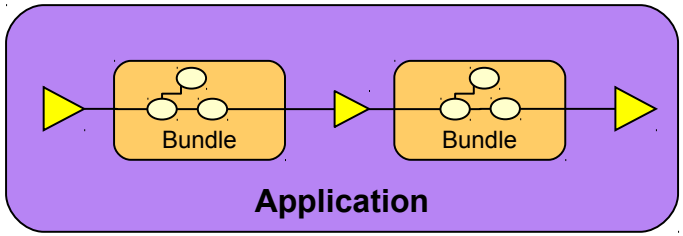


Figure 12. Blueprint support for SCA integration

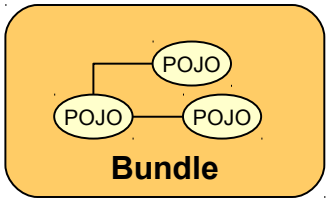
BP10 - Make best use of the component models on offer



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 context and scoped by an **OSGi Bundle**.

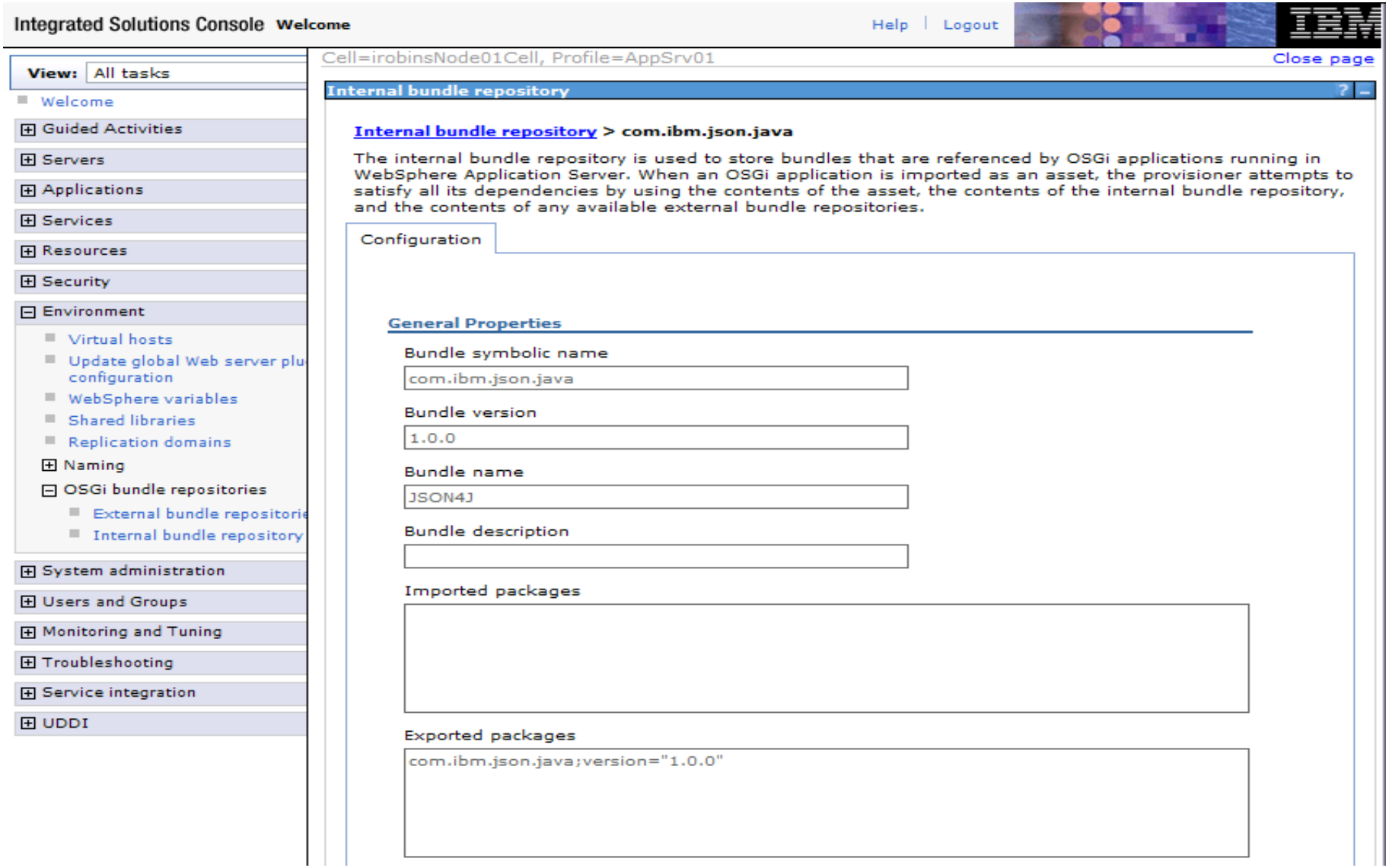
BP11 – Take good care of APPLICATION.MF

- Lists the bundles contained in the EBA under Application-Content, so that they are marked as isolated bundles
- Always provide your own APPLICATION.MF
- In production environment, upload bundles in bundle repositories except WARs

```
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.MF

BP12 – Use bundle repositories



BP12 – Create Web Application Bundles (WABs) not WARs

- What?
  - When developing enterprise OSGi applications, create a Web application bundle (WAB) and do not rely on the WAR to WAB conversion.
- Why?
  - If making WARs, the automatic conversation will need to convert WAR to WAB
    - No versioning in Import-Package
    - Some identity missing
    - Less flexible in future maintenance

BP13 - Only Use-Bundle when you must

- What?
  - Use-Bundle expresses preferable one bundle over any others that export the same package.
- Why?
  - Introduce additional provisioning work if you include the Use-Bundle for no specific reasons
- When?
  - Specify Use-Bundle when two applications need to share the same version of a class to ensure they are wired to the same providing bundle.

```
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.MF

BP13 - Only Use-Bundle when must - Example

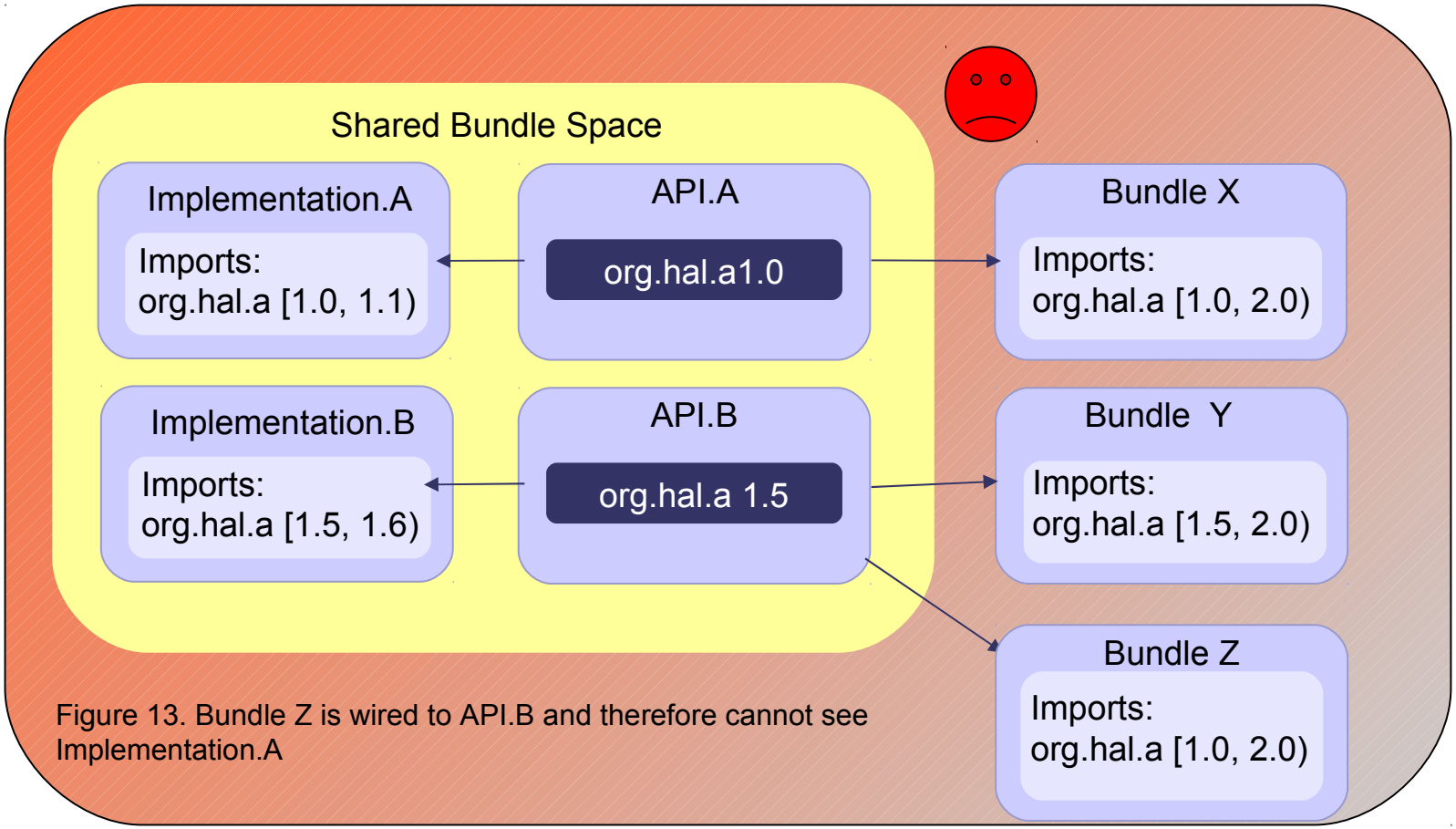


Figure 13. Bundle Z is wired to API.B and therefore cannot see Implementation.A

BP13 - Only Use-Bundle when must - Example

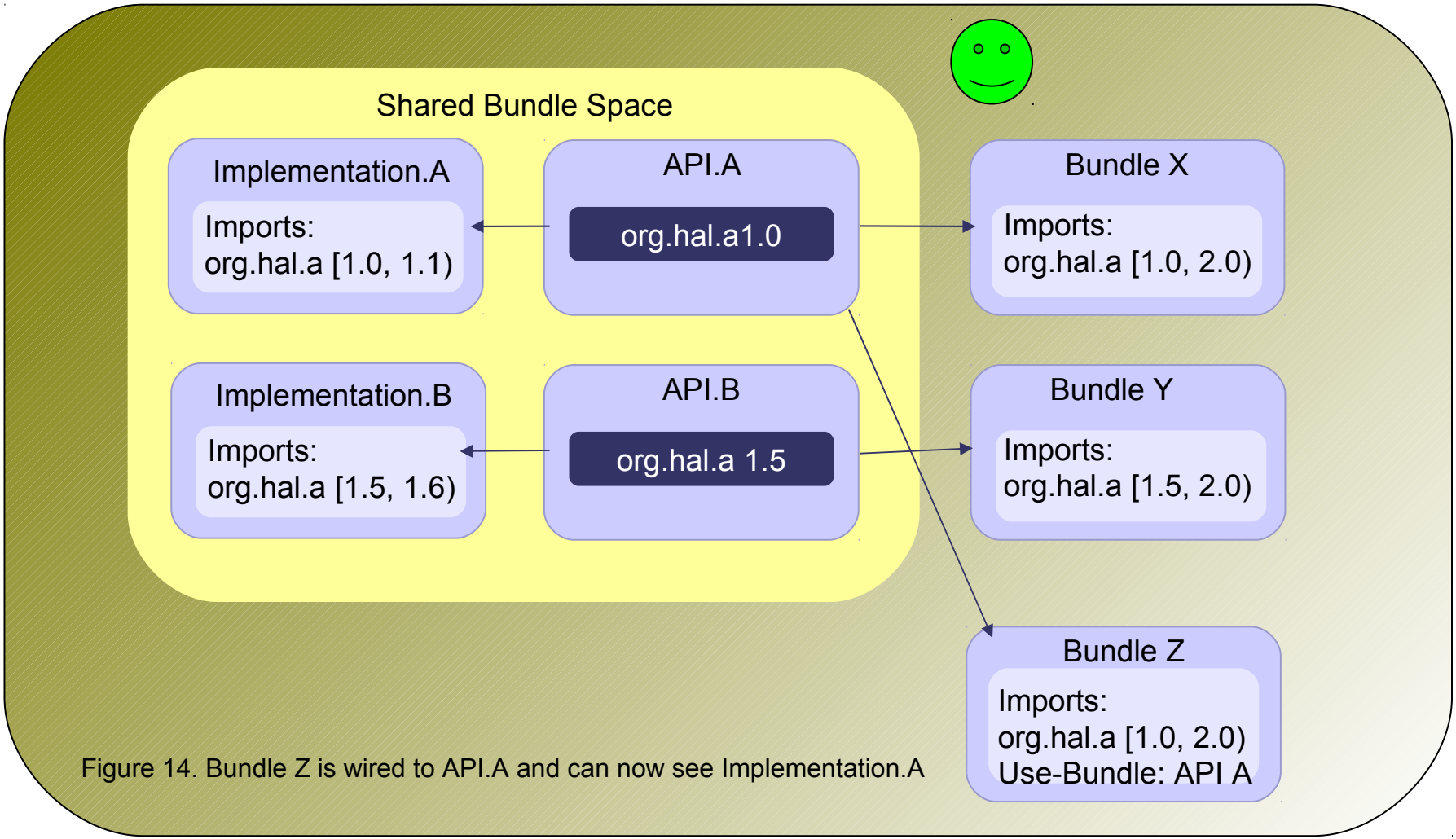


Figure 14. Bundle Z is wired to API.A and can now see Implementation.A

BP14 - Use persistence bundles to share your persistence units

- What?
  - When using JPA in an application, package the bundle as an OSGi persistence bundle instead of merging into other bundles.
- Why?
  - Persistence bundles can also be shared by a large number of OSGi based persistence clients
- Example

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Blog Persistence Bundle
Bundle-SymbolicName:
com.ibm.ws.eba.example.blog.persistence
Bundle-Version: 1.1.0
Bundle-Vendor: IBM
Bundle-ActivationPolicy: lazy
Meta-Persistence:
Import-Package:
com.ibm.ws.eba.example.blog.comment.persistence.api;version="[1.0.0,1.1.0)",

com.ibm.ws.eba.example.blog.persistence.api;version="[1.0.0,1.1.0)",
 javax.persistence;version="1.0.0"
```

BP15 – Let container to do the heavy lift

- What?
  - Use container services to build applications and supply enterprise qualities of service rather than writing code to manage services.
- Why?
  - Reliable
  - Easier to maintain business code

- How?
  - Use blueprint
  - Use JPA manage your resource
  - Use Declarative Transaction

Summary

OSGi Application Best Practices

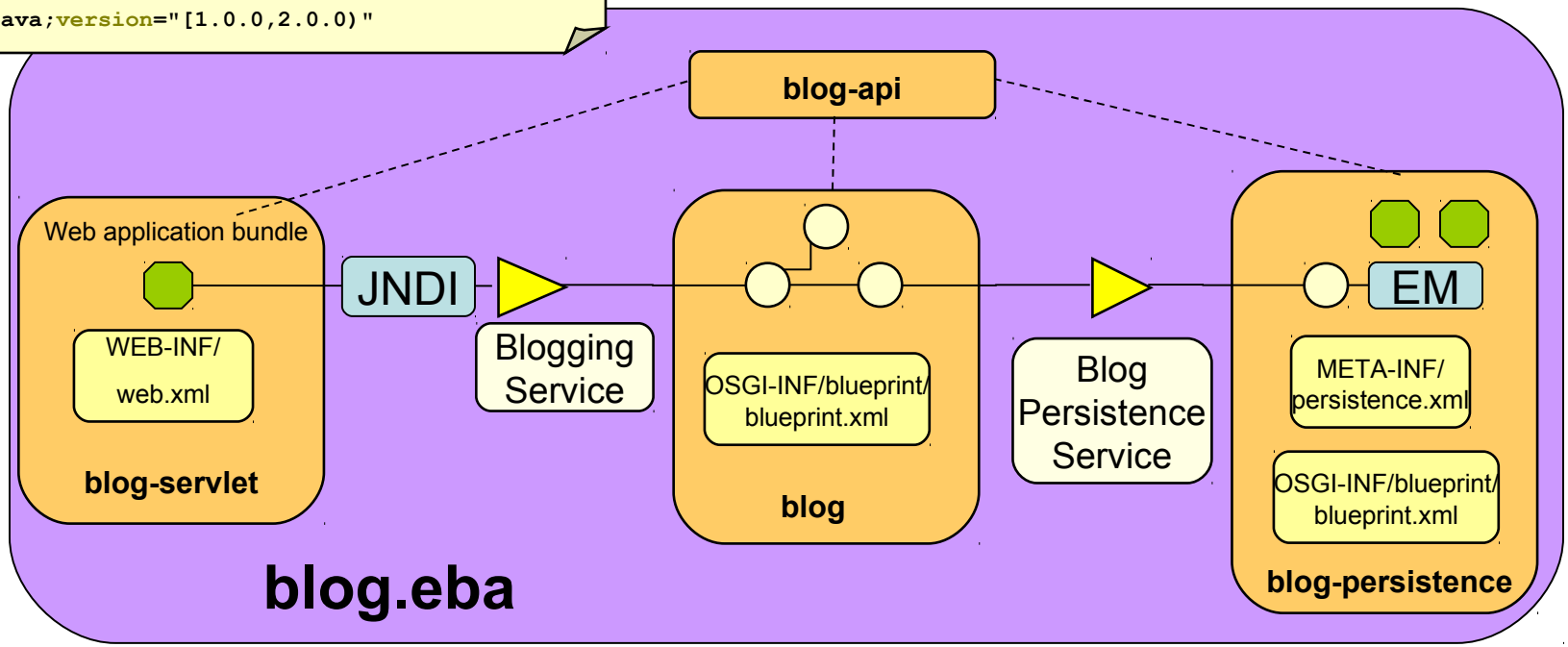
- BP1 - Use Import-Package instead of Require-Bundle
- BP2 - Avoid split packages
- BP3 - Version bundles and packages
- BP4 - Separate API from implementations
- BP5 - Share services not implementations
- BP6 – Make bundles loosely coupled & highly cohesive
- BP7 - Use Blueprint

WebSphere Application Server OSGi Application Specific Best Practices

- BP8 - Use Blueprint to enable service-based provisioning
- BP9 - Use Blueprint to enable SCA integration
- BP10 - Make best use of the component models on offer
- BP11 - take good care of APPLICATION.MF
- BP12 – Use bundle repositories
- BP13 – Create Web Application Bundles(WABs) instead of WARs
- BP14 - Only Use-Bundle when you must
- BP15 - Use persistence bundles to share your persistence units
- BP16 – Let container to do the heavy lift

Quiz – What best practices are used?

```
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)"
```



References

IBM DeveloperWorks Articles

- Best practices for developing and working with OSGi applications
- Developing enterprise OSGi applications for WebSphere Application Server

WebSphere Application Server OSGi feature pack InfoCenter

- WebSphere Application Server Information Center including documentation for the Feature Pack for OSGi Applications a

Get products and technologies

- IBM WebSphere Application Server V7 for Developers
- IBM WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API 2.0
- IBM Rational Development Tools for OSGi Applications
- IBM Rational Application Developer Beta for OSGi-based application tools

Discuss Forum

- OSGi Application feature pack discussion forum

Questions?

Copyright and Trademarks