rfc-0194-EJB-Integration

Draft

16 Pages

Abstract

The Enterprise JavaBeans architecture (EJB), part of JEE technology, is a managed server-side component architecture for the construction of enterprise applications[4]. While OSGi moves towards enterprise, it has long been interest in integrating EJB into OSGi. This RFP describes the requirements for integrating EJB into the OSGi framework with the intention that the current EJB developers should be able to reuse their current skills and assets to develop and deploy EJBs that run in and exploit the OSGi framework.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the “Distribution”) in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. "OSGi Name Space" shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable,

worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future ("Future Specification"), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 1.

```
Source code is shown in this typeface.
```

0.6 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	10/31/2012	Emily Jiang, IBM, emijiang@uk.ibm.com
0.1	03/10/12	Emily Jiang, rework based on the feedback from Austin f2f on 30th Jan 2013
0.2	03/13/12	Emily Jiang, updated as per discussion on EEG call (13/03/2012)

1 Introduction

The Enterprise JavaBeans architecture (EJB) is an architecture for the development and deployment of component-based business applications[3]. EJB is a part of Java EE that already is well used for developing enterprise applications. However, EJB has limited support for modularity and life-cycle dynamics. With the increase in popularity of OSGi for developing and deploying enterprise applications, the need arises for combining these two technologies. There has long been interest in EJB in the context of OSGi and some application servers such as WebSphere Application Server, GlassFish, OW2 Easybeans and OW2 JOnAS have their own support for EJB in OSGi applications. An OSGi standard way should be defined in order for applications to be portable. This RFP defines the requirements for such a standard.

2 Application Domain

Many enterprise applications have been written using the EJB component model. These applications are often large and require modification from time-to-time, as bugs are fixed or the needs of the business change. Lack of strong modularity in Java EE makes maintenance of large EJB applications difficult. Developers need to be able to build EJB applications in a more modular way. In doing so, they want to re-use their existing EJB assets and skills and only have to learn the new modularity features. Deployers need to be able to deploy these modular EJB applications in ways that leverage the modularity, for example, to manage versioned fixes to an application.

2.1 Terminology + Abbreviations

EJB – Enterprise JavaBean.

EJB Bundle – An OSGi bundle that contains EJBs.

2.2 EJB

The EJB specification is part of Java EE specification. The EJB Specification was originally developed in 1997 and later EJB1.1, EJB2.0 (JSR19), EJB2.1 (JSR153), EJB3.0 (JSR220) and most recent EJB 3.1 (JSR 318) further improved the design. EJBs are deployed and executed in an EJB container, which can be provided by a Java EE Application server.

2.3 EJB Example

There are three types of Enterprise beans: Session Beans, Message Driven Beans and Entity Beans(deprecated). Below is an example of stateless session bean, using EJB 3 annotations.

```
/**
 * Local stateless session Order EJB
 */
@Local
@Stateless
public class OrderEJB implements Order {
    ...
    @Override
    public String process() {
        ...
        return "Order Processed";
    }
}
```

2.4 Modular EJB in Apache Aries

The Apache Aries project (http://aries.apache.org) has created an integration between EJB and OSGi, which introduced the concept of an EJB bundle. An EJB bundle is a bundle that contains EJBs. The EJB bundle is identified by the presence of 'Export-EJB' header in the bundle manifest.

The value of this header specifies the stateless or singleton session beans to be registered in the OSGi service registry. The meaning of the header is as follows:

NONE: process all EJBs but register none of them in the service registry.

A single space: register all EJBs in the service registry.

A comma separated list of class names of the EJBs to be registered in the service registry.

Below is an example of MANIFEST.MF for the Order Bean. The OrderEJB will be registered in the service registry under the interface of `com.acme.order.api.Order`.

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Acme order processing service
Bundle-SymbolicName: com.acme.order.service
Bundle-Version: 1.0.0
Export-EJB: OrderEJB
Import-Package:
 com.acme.order.api;version="[1.0.0,1.1.0)",
 javax.ejb;version="3.1"
```

An EJB is registered in OSGi service registry with the following properties:

ejb.name – The name of the EJB

ejb.type – The type of the EJB, which is either Stateless or Singleton.

Service.exported.interfaces: The remote interfaces of the exported EJB. This enables the EJB to be accessed outside the enterprise application using the OSGi Remote Services specification.

The EJB lookup mechanisms in OSGi continues to support the traditional EJB lookup such as @EJB and JNDI (java:comp/env). With the EJB presence in OSGi service registry, the service look up via (osgi:service) is supported as well.

Traditional EJB look up:

```
/**
 * Order EJB to be injected
 */
@EJB
private Order orderService = null;

or

orderService =
  (Order) new
InitialContext().lookup("java:comp/env/com.acme.order.View/orderService");
```

EJB look up via service registry

```
orderService =
  (Order) new
InitialContext().lookup("osgi:service/com.acme.order.api.Order");
```

2.5 Modular EJB in Glassfish

Glassfish v3 supports EJB bundles. Similar to Apache Aries, an EJB bundle is identified by the presence of the Manifest header of 'Export-EJB' . The value of this header specifies the stateless or singleton session beans to be registered in the OSGi service registry. The implementation does not support remote interfaces.The meaning of the header is as follows.

ALL: all local Stateless and Singleton EJBs exported with local business interfaces.

NONE: no EJBs are exported.

A list of EJB names: the EJBs listed here are exported.

GlassFish supports the injection of OSGi services into an EJB by extending the Context and Dependency Injection (CDI) framework.

@Stateless

public class GF_EJB {

@Inject @OSGiService(Dynamic=true)

Bar bar;

...

}

3 Problem Description

EJB is a standard component architecture for building object oriented and distributed business applications in Java. The EJB container provides system-level services such as persistence, transaction and security while the EJB developers can concentrate on the business logic.

The HTTP Service and Web Application Specifications define how web applications can exploit OSGi, which cover some use cases that integrate Java EE technology. Often, such applications use EJB as the component model for business logic and so the use of EJBs in OSGi helps complete the set of Java EE technologies that enterprise OSGi users are looking for.

4 Requirements

EJB001 – The solution MUST make it possible to use the EJB annotations (or XML equivalent) in an OSGi bundle to expose and consume EJBs.

EJB002 – The solution MUST make it possible to publish EJBs in the OSGi Service Registry.

EJB003 – The solution MUST make it possible to consume OSGi services in an EJB

EJB004 – The solution MUST make it possible to select OSGi services used in EJBs based on OSGi filters.

EJB005 – The solution MUST make it possible to support JNDI look up.

EJB006 – The solution MUST make it possible to write a portable EJB jar that runs both in Java EE as well as in OSGi.

EJB007 – The solution MUST consider the thread-safety issues that can arise when migrating EJB beans from Java EE to OSGi.

EJB008 – The solution MUST consider the issues that can arise in relation to the dynamic bundle lifecycle in OSGi.

EJB009 – The solution MUST make it possible to take advantage of the dynamic service capabilities of OSGi.

EJB010 – The solution MAY introduce additional annotations.

EJB011 – The solution MUST define the behavior in case of incorrect EJB metadata.

EJB013 – The solution MUST define an opt-in mechanism. Bundles not opting in MUST not be considered by the

EJB-OSGi integration layer.

EJB014 – The solution MUST support the simplified APIs of EJB3.x integration in OSGi but MUST NOT prevent a compliant implementation from also supporting earlier EJB APIs.

EJB015 – The solution SHOULD provide a mechanism to specify additional OSGi service registration properties for EJB beans.

EJB016 – All the inter-bundle interactions with EJBs - MUST go through the OSGi Service Registry.

EJB017 – The solution MUST consider the issues that can arise with OSGi service dynamism when these services are injected into a EJB bean.

EJB018 – The solution MUST provide a mechanism to consume multiple matching services/beans in an EJB bean.

EJB019 - The solution MUST support the transaction, security, persistence and lifecycle semantics associated with EJBs.

EJB020 - The solution MUST support the standard EJB injection models.

EJB021 - The solution MUST support service registration of stateless and singleton EJBs and MAY support service registration of other EJB types.

EJB022 – The solution MUST support local and remote EJBs.

4.1 Requirements Derived from RFP 98 [6].

EJB023 – The solution SHOULD enable the standard EJB artifacts to remain installed when a supporting EJB container is dynamically replaced.

EJB024 – The solution MUST NOT impede the ability of an OSGi-compliant EJB container implementation to also be compliant with the Java EE EJB specification.

EJB025 – The solution MAY define the additional aspects that are required for EJB to be properly integrated in an OSGi framework but MUST NOT make any syntactic changes to the Java interfaces defined by the EJB specifications.

EJB026 – The solution MUST NOT require an OSGi Execution Environment greater than that which satisfies only the signatures of EJB specification.

4.2 Requirements derived from RFP 0146[7].

EJB027 – The solution MUST support CDI Integration.

EJB028 – The solution MUST Support the standard CDI injection.

5 Technical Solution

5.1 Architecture overview

Bundle is the deployment unit in OSGi. This RFC takes a design approach where an EJB Jar is deployed as an OSGi bundle in OSGi framework. This bundle is referred to as EJB bundle throughout the specification.

An EJB bundle is an OSGi bundle that contains EJBs. This specification describes the design requirements for an OSGi EJB container that supports EJB applications written to the EJB specification. The EJB container itself is deployed as one or more OSGi bundles.

The design uses the OSGi extender pattern[8]., where the EJB container acts as an extender that is responsible for observing the life cycle of EJB bundles. The EJB extender provides the capbility of osgi.ejb extender namespace with the following manifest header.

Provide-Capability: osgi.extender;osgi.extender=”osgi.ejb”

When an EJB bundle is started, the EJB container processes the configuration files of the application and instantiates and manages the lifecycle of EJBs in the EJB bundle. The EJB Extender manages the EJB bundles.

5.1.1 EJB Bundle

An EJB bundle is defined as a normal bundle that contains EJBs. It is identified by declaring a requirement on the EJB extender. This requirement activates the extender to inspect the EJB bundle's export.ejb directive. The directive of 'export.ejb' specifies which EJBs will be registered in the OSGi Service Registry.

~~It is identified by the opt-in header of Export-EJB. The following table lists the meaning of the Export-EJB header.~~

~~Export-EJB~~export.ejb: ~~(one space)*~~	Process all EJBs and register them in the service registry
export.ejb~~Export-EJB~~: <comma separated ejb names> (e.g. orderEJB, colorEJB)	Process all EJBs but only register the listed EJBs in the service registry
~~: NONEExport-EJB~~	~~Process all EJBs but do not register any EJBs in the service registry~~
Absence	Process all EJBs but do not register any EJBs in the service registry~~Do not process any EJBs in the bundle~~

Require-Capability: osgi.extender; filter:="(osgi.extender=osgi.ejb)";export.ejb:="*"

An EJB Bundle ~~with the header of Export-EJB:~~ registers services with the following service properties. *The service properties are detailed below.*

Key	Data Type	Description
ejb.name	String	The name of the EJB
ejb.type	String	The type of the EJB (e.g. Stateless, Stateful, Singleton)
service.exported.interfaces	String+	The remote interfaces the EJB implements
service.exported.configs~~remote.service.configuration~~	String~~Boolean~~	osgi.ejb means the EJB container does the remoting and its rmi-iiop.~~Specify whether the EJB container does the remoting~~

An EJB bundle is defined as follows:

- An EJB is a valid OSGi bundle and as such must fully describe its dependencies
- An EJB bundle follows the OSGi bundle life cycle.
- An EJB bundle is differentiated from a normal bundle through the ~~presence of Export-EJB manifest header~~requirement on the EJB extender. The ~~header~~export.ejb directive specifies the EJBs to be registered in the OSGi Service Registry. The EJB bundle can contain session beans and MDBs. The processing of entity EJB is not required.
- An EJB bundle supports EJB3.x beans and this specification does not mandate the support of the earlier EJB versions, e.g. EJB2.1.

5.1.2 Fragment bundles

An OSGi fragment is a Jar file with specific manifest headers that enable it to attach to a specified host bundle or specified host bundles in order to function. Fragments are treated as part of the host bundles. Relevant definitions of the fragment are merged with the host bundles definitions before the host is resolved, as long as the information does not conflict. Fragments extend bundles with resources, classes, and permitted headers enabling you to customize your bundles. This specification limits what effects a fragment bundle can have on a EJB bundle in the following way:

- A fragment can contribute additional EJBs to a host bundle.
- A fragment cannot replace the ejb-jar.xml for a host bundle. If there are conflicts, the host configuration wins.
- A fragment can contribute classes to the bundle class space.

~~of the host bundle to determine if a bundle is a EJB bundle.sAn OSGi EJB container must only look at the header~~ A bundle ~~cannot~~might be converted to an EJB bundle by attaching a fragment that specifies ~~Export-EJB~~the requirement of EJB extender manifest header. ~~Without this restriction, a bundle that is not originally authored as an EJB bundle can potentially make its EJBs accessible. This restriction will avoid such unintended behavior.~~

5.2 EJB bundle life cycle

5.2.1 Installing an EJB bundle

An EJB bundle is a valid OSGi bundle whose manifest specifies ~~Export-EJB manifest header~~the requirement of the EJB extender. An EJB bundle can be installed into the framework using the standard BundleContext.installBundle API variants. Once installed, an EJB bundle's life cycle is managed just like any other bundle in the framework.

5.2.2 Starting an EJB bundle

An EJB application is started by starting its EJB bundles. The EJB Extender listeners for the bundle starting and lazy activation life cycle events to initiate this process. A bundle that has a lazy activation policy should not be transitioned to the STARTED state by the EJB Extender until a request is made that requires a class to be loaded.

~~The EJB Extender recognizes an EJB bundle by looking for the presence of Export-EJB manifest header. The extender may not recognizes a bundle as an EJB bundle unless the Export-EJB header is present in its manifest.~~

After recognizing an EJB bundle, the extender initiates the process of deploying the application into an EJB container. It must generate a DEPLOYING event as described in section x. The container must validate the export.ejb directive~~Export-EJB manifest header value~~ to ensure the EJBs can be located in the bundle. Otherwise, a FAILED event must be emitted. After successfully deploying the application, a DEPLOYED event must be generated to indicate that the application is now in service.

5.2.3 Stopping an EJB bundle

An EJB application is stopped by simply stopping the corresponding EJB bundle. In response to a bundle STOPPING event, the extender must initiate the process of undeploying the application from the EJB container and clean up any resources.

- An UNDEPLOYING event is emitted to state that the application will be removed.
- EJB container must clean up any application specific resources.
- Finally, an UNDEPLOYED event is emitted.

Once the bundle is stopped, the OSGi framework will automatically unregistered any services registered by the EJB application.

5.2.4 Uninstalling an EJB bundle

An EJB application can be uninstalled by uninstalling the corresponding EJB bundle. The application bundle will be uninstalled from the OSGi framework, and will be completely removed when the framework is refreshed.

5.3 Integration with EventAdmin service

If the EventAdmin service is registered then the EJB extender bundle must emit the following events:

- org/osgi/service/ejb/DEPLOYING – the EJB extender has spotted an EJB bundle and started the process of deploying the EJB application.
- org/osgi/service/ejb/DEPLOYED – the EJB extender has finished deploying the EJB application, and the application is now running (in service).
- org/osgi/service/ejb/UNDEPLOYING – the EJB extender started removing the EJB application in response to the bundle being stopped.
- org/osgi/service/ejb/UNDEPLOYED – the EJB extender has removed the EJB application. The application is no longer in service.
- org/osgi/service/ejb/FAILED - the EJB extender has failed to deploy the EJB application, this will be emitted after the DEPLOYING event has fired.

For each event the following properties must be published:

- "bundle.symbolicName" (String) the symbolic name of the EJB bundle.
- "bundle.id" (Long) the id of the EJB bundle.
- "bundle" (Bundle) the Bundle object of the EJB bundle.
- "bundle.version" (Version) the version of the EJB bundle.
- "timestamp" (Long) the time when the event occurred
- "extender.bundle" (Bundle) the Bundle object of the EJB Extender bundle
- "extender.bundle.id" (Long) the id of the EJB Extender bundle
- "extender.bundle.symbolicName" (String) the symbolic name of the EJB Extender bundle.
- "extender.bundle.version" (Version) the version of the EJB Extender bundle.

In addition the FAILED event must also have the following property:
"exception" (Throwable) an exception detailing the problem.

5.3 Registering services in Service registry

W~~hen specified in the EJB bundle's manifest header of Export-EJB~~ith the presence of the directive of export.ejb, the EJBs will be registered in the service registry. The entity EJBs and MDBs are not registered in the service registry. If they are listed in the ~~Export-EJB header~~export.ejb directive, they will be ignored.

There are three types of session beans: Stateful, Stateless and Singleton beans.

For registering stateful and singleton beans, an instance of the beans will be registered in the service registry with the service properties specified in 5.1.1.

When registering the Stateful session beans, the current framework cannot support of creating multiple service instances per consuming bundle, which leads to the existence of RFC Service Scope [9]. The RFC Service Scope introduces a third scope: prototype, besides the existing two scopes, *singleton and bundle*. ~~Prototype scope is used by specifying the service.scope proerty with 'prototype' when registering Stateful session beans.~~

A statefule session bean needs to implment the new type of PrototypeServiceFactory and then registers a PrototypeServiceFactory object as the service object, This service will have the prototype scope.

In the consuming bundle, it can use the following new method on the BundleContext.

```
<S> ServiceObjects<S> getServiceObjects(ServiceReference<S> reference)
```

The method of getService() will obtain a new service object for the prototype scope services.

5.4 OSGi EJB Container

The specification defines an OSGi EJB container implementation as one or more OSGi bundles that collectively implement EJB 3.x specifications. The following section describes requirement for an OSGi EJB container. The EJBs can be declared either in ejb-jar.xml under META-INF/WEB-INF or via annotations. A consuming client can be either EJB client or Web client and it can use either annotation (@EJB) or ejb-ref in the ejb-jar.xml to obtain an EJB instance.

The specification requires that an OSGi EJB container provide the integration with

- Transaction
- JPA
- Security
- JNDI (look up via java:global, java:app, java:module)

5.4.1 Resource lookup

For an EJB bundle, there might be a number of EJBs defined. The EJBs can be defined either via annotations or XML in the EJB deployment descriptor, which is called META-INF/ejb-jar.xml or WEB-INF/ejb-jar.xml in Java EE. The EJBs defined in the host and fragments should be searched through.

5.4.2 Resource injection and annotations

The EJB application deployment descriptor may specify the metadata-complete attribute on the ejb-jar element of the ejb-jar.xml. If the attribute value is true, no annotation scan will be performed. Any EJB annotations will be ignored. If the attribute value is false, the container should examine the classes for annotations. This specification must support the EJB 3.x annotations. If JPA, CDI, Security are used in the EJB bundle, the relevant annotations such as @PersistenceContext, @PersistenceUnit for JPA, must be supported.

5.4.3 EJB application class loader

The implementation should not allow the application to override Java SE or Java EE platform classes, such as those in java.* and javax.* namespaces, that either Java SE or Java EE do not allow to be modified.

5.5 Component model interoperability

An EJB bundle may need to inter-operate with other component models such as Blueprint services (RFC124) and Declarative Services. Since EJB container owns the life cycle of an EJB bundle, this eliminate the possibility of an EJB bundle to be simultaneously be a component of another component model.

A typical interaction between the different component models is that an EJB bundle depends on a Blueprint or DS service. Interactions between them are managed by the OSGi service registry. An EJB bundle can perform service lookup via BundleContext or via jndi lookup such as osgi:services or @Resource annotation.

6 Data Transfer Objects

RFC 185 defines Data Transfer Objects as a generic means for management solutions to interact with runtime entities in an OSGi Framework. DTOs provides a common, easily serializable representation of the technology.

For all new functionality added to the OSGi Framework the question should be asked: would this feature benefit from a DTO? The expectation is that in most cases it would.

The DTOs for the design in this RFC should be described here and if there are no DTOs being defined an explanation should be given explaining why this is not applicable in this case.

This section is optional and could also be provided in a separate RFC.

7 Considered Alternatives

For posterity, record the design alternatives that were considered but rejected along with the reason for rejection. This is especially important for external/earlier solutions that were deemed not applicable.

8 Security Considerations

Description of all known vulnerabilities this may either introduce or address as well as scenarios of how the weaknesses could be circumvented.

9 Document Support

9.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

[3]. EJB 3.1 specification, JSR318. http://download.oracle.com/otn-pub/jcp/ejb-3.1-fr-eval-oth-JSpec/ejb-3_1-fr-spec.pdf
[4]. http://en.wikipedia.org/wiki/Enterprise_JavaBean
[5]. Apache Aries project, http://aries.apache.org/
[6]. RFP 98 OSGi Platform and Java EE Integration.
[7]. RFP 0146 CDI Integration
[8]. Kriens, P. The OSGi Extender Model, http://www.osgi.org/blog/2007/02/osgi-extender-model.html
[9]. RFC 195 Service Scopes

9.2 Author's Address

Name	Emily Jiang
Company	IBM
Address	
Voice	
e-mail	emijiang@uk.ibm.com

9.3 Acronyms and Abbreviations

9.4 End of Document