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 Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Document Information</td>
<td>2</td>
</tr>
<tr>
<td>0.1 Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>0.2 Terminology and Document Conventions</td>
<td>3</td>
</tr>
<tr>
<td>0.3 Revision History</td>
<td>3</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2 Application Domain</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Terminology + Abbreviations</td>
<td>4</td>
</tr>
<tr>
<td>2.2 EJB</td>
<td>4</td>
</tr>
<tr>
<td>2.3 EJB Example</td>
<td>4</td>
</tr>
<tr>
<td>2.4 Modular EJB in Apache Aries</td>
<td>5</td>
</tr>
<tr>
<td>2.5 Modular EJB in Glassfish</td>
<td>6</td>
</tr>
<tr>
<td>3 Problem Description</td>
<td>6</td>
</tr>
<tr>
<td>4 Use Cases</td>
<td>7</td>
</tr>
<tr>
<td>4.1 Use of EJB as a component model in OSGi</td>
<td>7</td>
</tr>
<tr>
<td>4.2 Register EJBs as services in OSGi Service registry</td>
<td>7</td>
</tr>
<tr>
<td>4.3 EJBs using OSGi services</td>
<td>7</td>
</tr>
<tr>
<td>4.4 Transaction integration with EJB in OSGi</td>
<td>7</td>
</tr>
<tr>
<td>4.5 JPA integration with EJB in OSGi</td>
<td>7</td>
</tr>
<tr>
<td>4.6 Security integration with EJB in OSGi</td>
<td>8</td>
</tr>
<tr>
<td>4.7 Life-cycle Use Cases</td>
<td>8</td>
</tr>
<tr>
<td>5 Requirements</td>
<td>8</td>
</tr>
<tr>
<td>5.1 Requirements Derived from RFP 98 [6]</td>
<td>9</td>
</tr>
<tr>
<td>5.2 Requirements derived from RFP 0146[7]</td>
<td>9</td>
</tr>
<tr>
<td>6 Document Support</td>
<td>10</td>
</tr>
<tr>
<td>6.1 References</td>
<td>10</td>
</tr>
<tr>
<td>6.2 Author’s Address</td>
<td>10</td>
</tr>
<tr>
<td>6.3 End of Document</td>
<td>10</td>
</tr>
</tbody>
</table>
0.2 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 6.1.

Source code is shown in this typeface.

0.3 Revision History

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>20/05/12</td>
<td>Emily Jiang (IBM) – Initial version</td>
</tr>
<tr>
<td>0.2</td>
<td>19/06/12</td>
<td>Emily Jiang feedback from NYC F2F discussion</td>
</tr>
<tr>
<td>0.3</td>
<td>26/09/12</td>
<td>Emily Jiang updated based on the feedback from Basel F2F discussion and public bug comments</td>
</tr>
</tbody>
</table>

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 EJB3 annotations.

```java
/**
 * 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](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
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:

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

or

orderService =
```
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.

```java
@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 Use Cases

4.1 Use of EJB as a component model in OSGi

A software developer who is familiar with EJB may want to use OSGi to improve their application modularity. Rather than learning new component models in OSGi, they should be able to use their existing EJB skills to develop EJBs and only need to learn the new OSGi skill required to add that modularity (e.g. augment the MANIFEST.MF to turn them into EJB bundles).

4.2 Register EJBs as services in OSGi Service registry

Company ACME has deployed many enterprise applications, most of which are developed using EJBs. The company has decided to use OSGi and wants to be able to call the EJBs from non-EJB bundles. To do this, they wish to register the EJBs as services in the OSGi Service Registry.

4.3 EJBs using OSGi services

Company ACME has deployed some EJBs in OSGi and wants to access OSGi services provided by others OSGi bundles. They want to be able to do this in a way which is familiar to the EJB developer.

4.4 Transaction integration with EJB in OSGi

Company ACME has deployed some EJB applications, some of which use container managed transactions while others use bean managed transactions. The company wants to use OSGi to modularize these EJB applications and still have their transactions support work as before.

4.5 JPA integration with EJB in OSGi

Company ACME has deployed some EJB applications, some of which interact with JPA. The company wants to use OSGI to modularize these EJB applications and still have their JPA usage work as before. The integration should be able to inject an Entity Manager or Entity Manager Factory to EJBs.
4.6 Security integration with EJB in OSGi

Company ACME has deployed some EJB applications, some of which have application security specified. The company wants to use OSGi to modularize these EJB applications and still have their security usage work as before.

4.7 Life-cycle Use Cases

Company ACME has deployed some EJB applications, some of which need to be updated. The company wants to use OSGi to manage these EJB application updates without redeploying the entire applications again.

5 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.

5.1 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.

5.2 Requirements derived from RFP 0146[7].

EJB027 – The solution MUST support CDI Integration.

EJB028 – The solution MUST Support the standard CDI injection.
6 Document Support

6.1 References

[6]. RFP 98 OSGi Platform and Java EE Integration.
[7]. RFP 0146 CDI Integration

6.2 Author's Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Emily Jiang</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>IBM</td>
</tr>
<tr>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>Voice</td>
<td></td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:emijiang@uk.ibm.com">emijiang@uk.ibm.com</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Graham Charters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>IBM</td>
</tr>
<tr>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>Voice</td>
<td></td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:charters@uk.ibm.com">charters@uk.ibm.com</a></td>
</tr>
</tbody>
</table>

6.3 End of Document