RFC 146 OSGi for Java EE Connector Architecture

Draft

20 Pages

Abstract

This specification describes how Java EE Connector Architecture resource adapters will be supported in an OSGi environment. The resource adapters requires a Java EE Connector Architecture container service. The specification describes a model for resource adapters to access and be accessed by relevant OSGi services.

NOTE: Several sections of the design are still under discussion, and the design described in the document must be considered “work in progress”.

1 Document Information

1.1 Table of Contents

1.2 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 [1].

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

1.3 Revision History

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

Revision	Date	Comments
Initial draft of RFC	March 14, 2009	Paul Parkinson, Oracle, paul.parkinson@oracle.com
2nd Draft	November 5, 2009	JJ Snyder, Oracle, j.j.snyder@oracle.com Reformatted document to model RFC 66 (web container) Rewrote most of the document.
0.9	January 13, 2010	Mike Keith, Oracle michael.keith@oracle.com Cleanup to prepare for submission to group
0.9.1	January 26, 2011	Jesper Pedersen, Red Hat, jesper.pedersen@redhat.com Align text against the Java EE Connector Architecture 1.6 specification
0.9.2	January 31, 2011	Jesper Pedersen, Red Hat, jesper.pedersen@redhat.com Revised 5.2.2 and 5.4 -> 5.7.
0.9.3	February 11, 2011	Jesper Pedersen, Red Hat, jesper.pedersen@redhat.com Revised 5.2, removed 5.6, 5.7.1, 5.7.2, moved 5.3 to 5.7
0.9.4	March 8, 2011	Jesper Pedersen, Red Hat, jesper.pedersen@redhat.com Added „Container requirements (5.3.2), removed „Accessing OSGi environment“ (5.3), comments from David Bosschaert (david@redhat.com)
0.9.5	April 4, 2011	Jesper Pedersen, Red Hat, jesper.pedersen@redhat.com Added „Configuring a resource adapter“ to 5.2

2 Introduction

The OSGi for Java EE Connector Architecture container provides support for resource adapters written to the Java EE Connector Architecture Specification 1.6 [2].

The OSGi for Java EE Connector Architecture container specifies how a resource adapter packaged as a RAR can be installed into an OSGi based runtime, and how it can integrate with OSGi services. It also specifies how a resource adapter is managed by OSGi from a bundle lifecycle perspective.

Additionally, as OSGi evolves in enterprise space and new application models are emerging that leverage OSGi capabilities, it would be necessary for Java EE application components (in particular, resource adapters in the context of this RFC) to interoperate with other component models [3]. Java EE components will need to access and provide OSGi services, as well as interoperate with other components written to Declarative Services and Blueprint Services (RFC 124) [4].

While the specification provides support for OSGi programming model for resource adapters by describing how they can participate in a service base ecosystem, it is not a goal of this specification to provide support for creating resource adapters under a different component model such as Blueprint service.

3 Application Domain

Resource adapters are a critical part of applications built using Java Enterprise Edition (Java EE). The Java EE Connector Architecture model is a popular resource management and access development model that provides a well known API for applications to access various resource managers (such as databases, messaging systems, and legacy EIS and ERP systems) and SPI for resource adapters to exploit container services (such as transaction processing, security, work management, etc.). The Java EE Connector Architecture specification [2] describes how standard resource adapters function in an enterprise environment.

3.1.1 Structure of a Resource Adapter

A resource adapter is a collection of resources that is typically made up of a collection of some of the following:

1. Outbound Connection Factories and related classes
2. Inbound Activation, ActivationSpec and related classes
3. Utility classes
4. Resource files used by the Java classes
5. Descriptive meta information that ties the above elements together in a ra.xml file or provided through annotations

The following table describes the structure of a Java EE Connector Architecture resource adapter archive:

Resource Adapter (RAR) structure

Directory	Contents
/META-INF	Contains the MANIFEST.MF manifest file, as required for jar files. An application RAR can list its dependencies on external libraries needed in the manifest as described in JAR specification. During deployment of the resource adapter, the Connector container must make the correct versions of the extensions available to the application following the rules defined by the *Optional Package Versioning* mechanism (http://java.sun.com/j2se/1.4/docs/guide/extensions/) This directory also contains the optional ra.xml file that ties the resource adapter classes together.

The Java interfaces, implementation, and utility classes required by the resource adapter must be packaged as one or more JAR files as part of the resource adapter archive. A JAR file must use the .jar file extension. These JAR files can reside anywhere inside the RAR file.

Any platform-specific libraries required by the resource adapter must be packaged within the resource adapter archive.

Resource adapters can be packaged and signed into a Resource Adapter Archive format (RAR) file using the standard Java archive tools. The Java EE Connector Architecture specification provides a detailed description of the format of a RAR archive.

3.1.2 Resource Adapter deployment descriptor

The resource adapter deployment descriptor includes the following types of configuration and deployment information:

- Initialization parameters for resource adapters, managed connection factories, admin objects, and configuration
- Inbound and outbound Connection and ConnectionFactory definitions, message listeners, and activation specifications
- Transaction support
- Security role mapping

This information is located in the ra.xml descriptor file or in annotations on the classes them self.

4 Problem Description

The OSGi compendium currently does not include support for the concept of resource adapters. A resource adapter is a system-level software driver that is used by a Java application to connect to an Enterprise Information System (EIS). The resource adapter plugs into a container or framework and provides connectivity between the EIS, the container/framework, and the enterprise application. The resource adapter serves as a protocol adapter that allows any arbitrary EIS communication protocol to be used for connectivity.

The Java EE Connector Architecture [2] defines standard contracts that allow bi-directional connectivity between enterprise applications and EISs. The Java EE Connector Architecture enables an EIS vendor to provide a standard resource adapter for its EIS.

A resource adapter can declare various application attributes declaratively in its ra.xml file or through annotations. It would be logical that resource adapters should be embraced and fully supported in the OSGi environment.

We need a specification which supports the current generation of the Java EE Connector Architecture specification and embraces the concept of deploying resource adapters to an OSGi environment.

5 Requirements

This specification addresses the following requirements:

1. There MUST be a standard programming approach that makes it possible to deploy a standard resource adapter to a Java EE Connector Architecture container programmatically. This approach may be a traditional Java programming interface registered as an object or objects in the OSGi Service Registry, it may be a programming pattern that an OSGi bundle must follow in order to ensure that it is registered as a resource adapter via reflection, or it may be something else, such as a pattern that the bundle must use to register itself on a “whiteboard.”

2. The standard MUST NOT require that a particular configuration API or system, including the OSGi Configuration Administration Service, be supported.

3. The solution MUST support deploying resource adapters as RAR files, as described in the Java Connector Specification.

4. The solution MUST not mandate that resource adapters be packaged or deployed as RAR files, as described in the Java EE Connector Architecture specification, as long as they conform to this specification.

5. The solution MUST provide the ability to access resource connection factories of arbitrary type such as database access (JDBC datasources), messaging systems (JMS connection factories), and other EIS and ERP systems (resource adapter connection factories).

6. The solution MUST allow resource adapters to specify which packages to import from the OSGi Framework.

7. The solution MUST allow resource adapters to export packages and services to the OSGi Framework, as well as allow OSGi-aware resource adapters to access other OSGi services.

8. The solution SHOULD support resource adapter components to interoperate with components written to other models such as OSGi declarative services and Blueprint services. For example, the solution could support injection of component services into Java EE components based on additional meta-data associated with the application components.

9. It SHOULD be possible for a resource adapter bundle to remain installed when its Java EE Connector Architecture container is dynamically replaced.

10. An OSGi-compliant Java EE Connector Architecture container MUST NOT be impeded from also being compliant with the Java EE Connector Architecture specification.

11. An OSGi-compliant Java EE Connector Architecture container MUST support resource adapters implemented to the Java EE Connector Architecture 1.0, 1.5 and 1.6 specifications or any later versions which are backwards compatible with these specifications.

12. The OSGi Java EE Connector Architecture container design MUST NOT require an OSGi Execution Environment greater than that which satisfies the signatures of the Java EE Connector Architecture specification.

13. An OSGi Java EE Connector Architecture container MAY provide additional aspects of the technology that are required for resource adapter support to be properly integrated in an OSGi framework but MUST NOT make any syntactic changes to the Java interfaces defined by the Java EE Connector Architecture specification.

6 Technical Solution

6.1 Architectural overview

Bundles are the deployment and management entities under OSGi. The RFC takes a design approach where a resource adapter is deployed as an OSGi bundle in the framework. There is exactly one resource adapter bundle that corresponds to each deployed resource adapter in the framework.

The specification describes the design requirements for an OSGi Java EE Connector Architecture container that supports resource adapter components written to Java EE Connector Architecture specifications. The Java EE Connector Architecture container itself is deployed as one or more OSGi bundles.

The design uses OSGi extender pattern [6], where the Java EE Connector Architecture container includes an extender that is responsible for observing the life cycle of resource adapter bundles. When a resource adapter bundle is started, the extender processes the configuration of the resource adapter and instantiates and manages lifecycle of the resource adapter. The resource adapter bundles thus become managed bundles of the extender.

6.1.1 Resource Adapter Bundle (RAB)

A Resource Adapter Bundle (RAB) is defined as a normal OSGi bundle that contains the classes, interfaces, jars, etc. necessary for accessing a resource adapter.

A RAB is defined as the follows:

- A RAB is a valid OSGi bundle and as such must fully describe its dependencies.
- A RAB follows the OSGi bundle life-cycle.
- A RAB is differentiated from a normal bundle through the existence of metadata that describes the resource adapter. This metadata is located in the META-INF/ra.xml file or in annotations.

A mandatory manifest header (`META-INF/MANIFEST.MF`) is required to identify a resource adapter bundle

Key	Value	Description
`Resource-Adapter`	`<empty>`	Resource adapter bundle identifier

The RAB follows the packaging rules of a standard Java EE Connector Architecture archive (RAR).

TODO – File extention requirements

6.1.2 Fragment Bundles

Fragments are bundles that can be attached to one or more host bundles by the framework, which can modify the behavior of a host bundle. A fragment can add classes to the bundle class space, the bundle entry space, and

update imports and exports. Fragments are attached by the time a bundle is resolved. This specification limits what effects a fragment bundle can have on a RAB in the following way:

- A fragment cannot provide or replace the resource adapter metadata for a bundle.
- A fragment can provide additional vendor specific deployment descriptors.
- A fragment can contribute classes and resources to the bundle class space.
- A fragment can contribute platform-dependent native libraries to the bundle.

6.2 Resource adapter life cycle

TODO DS/Blueprint

6.2.1 Installing a resource adapter bundle

A RAB is a valid OSGi bundle containing the resource adapter metadata. A RAB can be installed into the framework using the standard `Bundlecontext.installBundle` API variants.

A RAB can be signed, but must comply with the bundle signing rules defined in the OSGi Core Specification v4.2.

Once installed, a RAB bundle's life cycle is managed just like any other bundle in the framework.

6.2.2 Starting a resource adapter bundle

A resource adapter is started by starting its corresponding resource adapter bundle. The Java EE Connector Architecture extender listens for the bundle starting and lazy activation life cycle events to initiate this process.

A resource adapter and the Java EE Connector Architecture extender may start in any order. The extender recognizes a resource adapter bundle by looking for the presence of a resource adapter bundle specific manifest header. The extender does not recognize a bundle as a resource adapter bundle unless the manifest header is present in the bundle.

After recognizing a resource adapter bundle, the extender initiates the process of deploying the resource adapter into the Java EE Connector Architecture container. It must generate a `DEPLOYING` event as described in section 5.3.

The Java EE Connector Architecture container processes deployment information by processing the resource adapter metadata and any vendor specific deployment metadata.

As resource adapters are modularized further into multiple bundles (and not deployed as RAR files only) it is possible that a resource adapter bundle can have import dependencies on other deployed bundles. The container must fully support metadata that specifies connection factories, administered objects, etc. whose classes are obtained via an `Import-Package` statement.

Any validation failures must prevent the resource adapter module from being accessed and must result in a `FAILED` event being emitted.

The Java EE Connector Architecture container performs the necessary initialization of resource adapter components in the bundles, as described in specification [1]. This involves the following:

- Obtain TransactionManager Service described in RFC98
- Create a `BootstrapContext` (which provides Timer, XATerminator, and WorkManager) for the resource adapter.
- Instantiate a configured `javax.resource.spi.ResourceAdapter`.
- Register `org.osgi.jca.ResourceAdapterWrapper` in the service registry. This is a wrapper of the ResourceAdapter without the lifecycle methods.
  - # TODO – What service properties should we register, if any?
- Call `void start(BootstrapContext bootstrapContext)` on the `ResourceAdapter`

The JCA container listens to bundles being installed that need to be processed. The following steps are executed for each applicable bundle:

- Instantiate configured `javax.resource.spi.ManagedConnectionFactory(s).`
- Call `Object createConnectionFactory(ConnectionManager connectionManager)` on the instantiated `ManagedConnectionFactory(s).`
- Register `ConnectionFactory(s)` in the service registry.
  - # TODO – The service is registered under the bundle context of the Resource Adapter bundle.
- Instantiate configured `javax.resource.spi.AdmininisteredObject(s).`
- Register `AdmininisteredObject(s)` in the service registry.

All instances of the `ConnectionFactory(s)` and `AdministeredObject(s)` must be registered in the service registry with an unique identifier in the supplied dictionary instance. The mandatory key is `osgi.jca.service.name`.

An example:

```
MyConnectionFactory cf = ...;

Dictionary d = new Hashtable();

d.put("osgi.jca.service.name", "MyCFInstance");

ctx.registerService(MyConnectionFactory.class, cf, d);
```

Note, that service registration of AdministeredObject(s) must register all specified administered object interfaces under the same key/value pair for each instance.

The Java EE Connector Architecture container is required to complete instantiation of connection factories and administration objects and register them in the service registry before the resource adapter is considered deployed.

After successful deploying the resource adapter, a `DEPLOYED` event must be generated to indicate that the resource adapter is now in service.

6.2.3 Configuring a resource adapter

TODO – Describe lookup of configured resource adapter objects

An instance of a resource adapter bundle can be configured through the OSGi Configuration Admin Service in case that there isn't enough vendor specific deployment information available.

#TODO –

- We need to specify Persistent ID's for Connection Factory(ies) (really ManagedConnectionFactory) and AdminObject(s) in the bundle
- There needs to be a way for the configuration of the Connection Factory(ies) and AdminObject(s) to trigger a deployment of these objects based on the metadata stored in the container. This is currently vendor specific, but one possibility is to leave this area as an implementation detail
- We need to consider that multiple configurations of a Persistent ID can happen – in relation to the ManagedConnectionFactory and AdminObject Persistent ID “templates”
- We need to consider how the configuration should be done – API, metadata file, ... It is along the same lines what vendor do with their vendor specific deployment files
- The dynamic part of the configuration admin service needs to be handled, as f.ex. rebinding in JNDI or updating a config-property isn't always possible. Simplest solution would be to only allow the initial configuration
- There should be a standard template for ManagedServiceFactory in order to configure the standard elements of the ra.xml. This is vendor specific today

A lot of the issues around this area is that this is handled by a vendor specific implementation of the necessary services.

If multiple instances of the same resource adapter bundle is needed the ManagedServiceFactory can be used.

TODO – RAB vs. ConfigService

6.2.4 Stopping a resource adapter bundle

A resource adapter is stopped by simply stopping the corresponding resource adapter bundle. In response to a bundle `STOPPING` event, the extender must initiate the process of undeploying the resource adapter from the Java EE Connector Architecture container. This will involve the following:

- An `UNDEPLOYING` event is emitted to signal that the resource adapter will be removed.
- The Java EE Connector Architecture container calls `stop` on the `ResourceAdapter`.
- Finally, an `UNDEPLOYED` event is emitted.

Once the bundle is stopped the OSGi framework will automatically unregister any services registered by the resource adapter.

6.2.5 Uninstalling a resource adapter bundle

A resource adapter can be uninstalled by uninstalling the corresponding resource adapter bundle. The resource adapter bundle will be uninstalled from the OSGi framework, and will be completely removed when the framework is refreshed.

6.3 OSGi Java EE Connector Architecture container

The specification defines an OSGi Java EE Connector Architecture container implementation as one or more OSGi bundles that collectively implements the Java EE Connector Architecture specification.

The following sections describes the requirements for an OSGi Java EE Connector Architecture container.

6.3.1 Required bundles

The container implementation must supply an OSGi bundle which exports the Java EE Connector Architecture 1.6 specification application programming interface (API).

The export packages for the bundle must be:

```
javax.resource;version=1.6

javax.resource.cci;version=1.6

javax.resource.spi;version=1.6

javax.resource.spi.endpoint;version=1.6

javax.resource.spi.security;version=1.6

javax.resource.spi.work;version=1.6
```

TODO Align against other OSGi/EE specs – like OSGi/Web

6.3.2 Container requirements

An OSGi Java EE Connector Architecture container MUST use implementations of the following components that handles integration with the OSGi services

- Bootstrap context (`javax.resource.spi.BootstrapContext`)
- Connection manager (`javax.resource.spi.ConnectionManager`)

The container MUST also supply an interface for `javax.resource.spi.ResourceAdapter` which is used when resource adapter instance is registered in the service registry.

TODO – Name + methods

6.3.3 Java SE considerations

The Java EE Connector Architecture 1.6 specification requires Java Platform Standard Edition 6 as the minimum execution environment. Consequently, it is the minimum execution environment for running the OSGi Java EE Connector Architecture container.

6.3.4 Java EE considerations

An OSGi Java EE Connector Architecture container implementation will need to consider additional requirements in order to be Java EE compliant.

Java EE Connector Architecture 1.6 together with the Java EE 6 specification describes a comprehensive set of requirements that a Java EE compliant Java EE Connector Architecture container must implement.

In practice, a Java EE Connector Architecture container implementation supports a subset of Java EE services and makes them available to resource adapters, even when it is not fully compliant with Java EE specification.

These include support for

- JAAS
- Environment naming context (e.g. `java:comp/env` access)
- Transactions

TODO – JAAS

This specification highly recommends (but does not require) that an OSGi Java EE Connector Architecture container provide integration with:

- A Transaction manager (RFC 98)
- JNDI (RFC 142). Containers that need to support JNDI environment configuration from properties files must set the JNDI client's bundle's class loader to be the current Thread Context class loader prior to invoking a method on the application component. The context class loader must implement "`org.osgi.framework.BundleReference`" interface as described in the OSGi R4.2 core specification.

6.3.5 Resource adapter class loader

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

6.4 Use of OSGi services

The Java EE Connector Architecture container may use standard OSGi services to implement some of the features of the Java EE Connector Architecture 1.6 specification.

The following OSGi services are RECOMMENDED for implementing parts of the Java EE Connector Architecture 1.6 specification.

- EventAdmin Service – Used to publish events.

- Transaction Manager Services (RFC 98) – Used as the JTA transaction manager for resource adapters that participate in XA transactions.

- JNDI Services (RFC 142) – Used to register connection factories and administered objects (optional).

6.4.1 EventAdmin service

If the EventAdmin service is registered then the Java EE Connector Architecture extender bundle must emit the following events:

- `/org/osgi/service/connector/DEPLOYING` – the extender has spotted a resource adapter bundle and started the process of deploying the resource adapter.

- `/org/osgi/service/connector/DEPLOYED` – the extender has finished deploying the resource adapter and the resource adapter is now running.

- `/org/osgi/service/connector/UNDEPLOYING` – the resource adapter is being undeployed.

- `/org/osgi/service/connector/UNDEPLOYED` – the extender has removed the resource adapter. The resource adapter is no longer in service.

- `/org/osgi/service/connector/FAILED` – the extender has failed to deploy the resource adapter. This will be fired after the `DEPLOYING` event has fired.

For each event the following properties must be published:

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

In addition the `FAILED` event must also have the following property:

- "`exception`" (Throwable) an exception detailing the problem.

6.4.2 Transaction Manager Services (RFC 98)

Resource adapters may support JTA [7] transactions. The Java EE Connector Architecture container must support resource adapters that support JTA transactions. The Transaction services (RFC 98) provide the services that implement the JTA specification.

6.4.3 JNDI Services (RFC 142)

The Java EE Connector Architecture container can to register the connection factories and administration objects in the JNDI namespace. The JNDI services (RFC 142) provide the services that implement the JNDI specification.

7 Security Considerations

7.1 Java EE Connector Architecture container bundle

The Java EE Connector Architecture container should only be implemented by a trusted bundle. This bundle requires the following security permissions.

- `ServicePermission[get]` for the `org.osgi.service.log.LogService` interface. This allows the Java EE Connector Architecture container to log messages.
- `ServicePermission[get]` for the `javax.xml.parsers.SAXParserFactory` or `javax.xml.parsers.DocumentBuilderFactory` interface. The `SAXParserFactory` and `DocumentBuilderFactory` services can be used to process resource adapter descriptors.
- `AdminPermission[resource,class]` to call `Bundle.getEntry`, `Bundle.getEntryPaths` and `Bundle.loadClass` on the resource adapter bundles.

8 Document Support

8.1 References

1. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.
2. Java EE Connector Architecture 1.6, http://www.jcp.org/en/jsr/detail?id=322
3. Java EE 6 specification, http://jcp.org/en/jsr/detail?id=316
4. RFC 124 (Blueprint services)
5. Kriens, P. The OSGi Extender Model, http://www.osgi.org/blog/2007/02/osgi-extender-model.html
6. Uniform Resource Locators, RFC 1738, http://www.ietf.org/rfc/rfc1738.txt
7. JTA Transaction specification

8.2 Author's Address

Name	Jesper Pedersen
Company	Red Hat
Address	314 Littleton Road, Westford, MA, 01886, USA
Voice	+1-978-392-1000
e-mail	jesper.pedersen@redhat.com

Name	David Bosschaert
Company	Red Hat
Address	6700 Cork Airport Business Park Kinsale Road Cork Ireland
Voice	+353 21 230 3400
e-mail	david@redhat.com

8.3 Acronyms and Abbreviations

RAR – Resource Adapter Archive

RAB – Resource Adapter Bundle

8.4 End of Document