

This FAQ addresses frequently asked questions relating to the J2EE Connector Architecture aspects of Oracle Application Server Containers for J2EE 10g (10.1.3) and is broken into the following sections:

[What is J2EE Connector Architecture?](#)

[What are the advantages of J2EE Connector Architecture?](#)

[What is a resource adapter?](#)

[What are the system contracts?](#)

[What is CCI?](#)

[Where can I find more information about J2EE Connector Architecture?](#)

[How is a resource adapter packaged?](#)

[Does the current OC4J release support CCI?](#)

[What is oc4j-ra.xml?](#)

[What are the options for deploying a resource adapter?](#)

[How do I deploy a standalone resource adapter to OC4J?](#)

[How do I deploy an embedded resource adapter to OC4J?](#)

[How do I look up a resource adapter from my application?](#)

[Where can I find an example of deploying a resource adapter on OC4J?](#)

What is J2EE Connector Architecture?

The J2EE Connector architecture defines a standard architecture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EISs). Examples of EISs include ERP, mainframe transaction processing, database systems, messaging systems, and legacy applications not written in the Java programming language. J2EE Connector Architecture is a required part of J2EE 1.4 specification. It addresses the key issues and requirements of EIS

integration by defining a set of scalable, secure, and transactional mechanisms that enable the integration of EISs with application servers and enterprise applications.

What are the advantages of J2EE Connector Architecture?

Prior to the existence of the J2EE Connector Architecture, the Java platform had no standard architecture for integrating heterogeneous EISs. It was up to the individual EIS vendors and application server vendors to determine their own EIS integration solution. As a result, nonstandard and proprietary solutions had to be implemented for each EIS/application server combination. Suppose you have m number of application server vendors and n number of EIS vendors. The effort to integrate all these application servers with all EIS vendors without using the Connector architecture could be expressed by $(m * n)$. The introduction of the J2EE Connector Architecture has significantly reduced the complexity of EIS integration. By adhering to the J2EE Connector Architecture, EIS vendors no longer need to customize their product for each application server. Application server vendors who conform to the J2EE Connector Architecture do not need to add custom code when they add connectivity to a new EIS. Thus the integration effort is reduced to $(m + n)$. In addition to reducing the scope of integration, the Connector architecture also simplifies application development. Because the application servers and resource adapters rely on the system contract to provide the transaction, security, and connection pooling services for EIS integration, the application developer does not have to be concerned with these system-level details. The Connector architecture also helps reduce the scope of Tools integration.

What is a resource adapter?

A resource adapter is a system-level software driver used by an application server or an application client to connect to an EIS. By plugging into an application server, the resource adapter collaborates with the server to provide the underlying mechanisms, the transactions, security, and connection pooling mechanisms. A resource adapter is used within the address space of the application server.

What are the system contracts?

The J2EE Connector Architecture's system-level contracts define a "pluggability" standard between application servers and EISs. By adhering to the terms of these contracts when developing their components, an application server and an EIS know that connecting will be a straightforward operation of plugging in the resource adapter. The J2EE Connector Architecture 1.5 defines the following set of system-level contracts between an application server and EIS:

- Connection management contract - This contract enables an application server to pool connections to an underlying EIS to create a more scalable application environment.
- Transaction management contract - This contract enables an application server's transaction manager to manage transactions across multiple EIS resource managers or to support the local transactions that an EIS resource manager handles internally.
- Security contract - This contract enables secure access to an EIS.
- Lifecycle management contract - This contract allows an application server to manage the startup and shutdown of a resource adapter, including a mechanism for bootstrapping a resource adapter when it is deployed or when the application server starts up, and for notifying a resource adapter when it is undeployed or when the application server shuts down.
- Message inflow contract - This contract allows a resource adapter to deliver messages asynchronously to endpoints (such as message-driven beans) in an application server.
- Transaction inflow contract - This contract allows an imported transaction to be propagated to an application server by a resource adapter.

- Work management contract - This contract allows a resource adapter to perform tasks through the use of units of work submitted to an application server and executed by a work manager.

What is CCI?

CCI stands for Common Client Interface. The CCI defines a standard client API for application components to access EISs. The CCI enables application components and Enterprise Application Integration (EAI) frameworks to drive interactions across heterogeneous EISs using a common client API. The CCI is intended for use by Enterprise Application Integration (EAI) and enterprise tools vendors.

Where can I find more information about J2EE Connector Architecture?

Sun Microsystems Web sites.

<http://java.sun.com/j2ee/connector/>.

How is a resource adapter packaged?

A resource adapter is packaged into a Resource Adapter Archive (RAR) file using the Java Archive (JAR) format. A resource adapter archive file is identified by the file extension .rar. A resource adapter RAR file must contain a correctly formatted deployment descriptor (/META-INF/ra.xml). The deployment descriptor is an XML file containing deployment-specific information about the resource adapter (declarative information about the contract between the resource adapter provider and the deployer). The implementation - the Java classes and interfaces - of a resource adapter is typically packaged in one or more JAR files, and these JAR files are in the resource adapter module. It is also possible that the resource adapter module may contain platform-specific native libraries. Below is an example of a resource adapter module:

```
/META-INF/ra.xml  
/howto.html  
/images/icon.jpg  
/ra.jar  
/cci.jar  
/win.dll  
/solaris.so
```

The file ra.xml is the deployment descriptor. The files ra.jar and cci.jar contain the Java interfaces and implementation classes for the resource adapter. Last, win.dll and solaris.so represent native libraries that the adapter uses.

Does the current OC4J release support CCI?

Yes OC4J does support the CCI API. However, you need to use a resource adapter that supports CCI. For example, you can deploy and use Sun's CCI Blackbox resource adapter with your application (cciblackbox-tx.rar).

What is oc4j-ra.xml?

oc4j-ra.xml is the OC4J specific deployment descriptor for a resource adapter. It contains deployment configurations for deploying resource adapters to OC4J, which includes EIS connection information as specified in the deployment descriptor of the resource adapter, JNDI

name to be used, connection pooling parameters, and resource principal mapping mechanism and configurations. Below is an example of oc4j-ra.xml (assuming the resource adapter is using JDBC to connect to a database):

```
<?xml version="1.0"?>
<oc4j-connector-factories
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.oracle.com/technology/oracleas/schema/oc4j-connector-factories-10_0.xsd"
  schema-major-version="10" schema-minor-version="0">

  <connector-factory connector-name="myconnector"
    location="eis/MyEIS">
    <description>My RAR</description>
    <config-property name="connectionURL"
      value="jdbc:oracle:thin:@//db_host:1521/svcname" />
  </connector-factory>
</oc4j-connector-factories>
```

What are the options for deploying a resource adapter?

There are two options for deploying a resource adapter:

- Standalone deployment - The resource adapter is deployed as a standalone module.
- Embedded deployment - The resource adapter is first assembled into a J2EE application and then it is deployed as part of that J2EE application.

How do I deploy a standalone resource adapter to OC4J?

Using the Application Server Control Console is the recommended way for deploying a standalone resource adapter to OC4J. In Application Server Control Console, from the OC4J Home page, select the Applications Tab, view "Standalone Resource Adapters", and then select the resource adapter of interest. See the OC4J Resource Adapter Administrator's Guide 10g (10.1.3) for details.

How do I deploy an embedded resource adapter to OC4J?

Assuming your OC4J home is `$J2EE_HOME`, the resource adapter is packaged in `embedded_ra.rar`. You need to assemble the resource adapter (`embedded_ra.rar`) into an application module (EAR file) along with other application components, add the following to the application deployment descriptor of your application module (`/META-INF/application.xml`):

```
...
<application>
  ...
  <module>
    <connector>embedded_ra.rar</connector>
  </module>
  ...
</application>
```

Then you can package the application module into an EAR file. You can deploy the EAR file to OC4J in the same way that you deploy any other J2EE application EAR archives.

Using the Application Server Control Console is also the recommended way for deploying a embedded resource adapter to OC4J. In Application Server Control Console, from the OC4J Home page, select the Applications Tab, view "Applications", select the desired application that holding the resource adapter; then from the resulting Application Home page, under "Modules", select the resource adapter module of interest. See the OC4J Resource Adapter Administrator's Guide 10g (10.1.3) for details.

After deployment, OC4J will create a directory `$J2EE_HOME/application-deployments/your_application_name/embedded_ra` and generate a file `oc4j-ra.xml`, you can modify this file to provide OC4J specific configurations (EIS connection, JNDI name, etc.). Please see an example of `oc4j-ra.xml` in the "[What is oc4j-ra.xml?](#)" section. You will need to restart OC4J for the changes that you made to `oc4j-ra.xml` to take effect.

How do I look up a resource adapter from my application?

You can look up a resource adapter from your application using the JNDI name defined in `oc4j-ra.xml`. Using the `oc4j-ra.xml` example from the "[What is oc4j-ra.xml?](#)" section, and assuming it is configured for a JDBC datasource connecting to a database, you can lookup the datasource provided by the resource adapter as follows:

```
Context ic = new InitialContext();
DataSource ds = (DataSource)ic.lookup("java:comp/env/eis/MyEIS");
```

Or you can lookup a logical JNDI name to make your application code more portable. Assuming the logical JNDI name is `eis/EIS` and the physical JNDI name defined in `oc4j-ra.xml` is `eis/MyEIS`, then you need to add the following in your OC4J specific deployment descriptor (`orion-web.xml` for web modules and `orion-ejb-jar.xml` for EJB modules):

```
<resource-ref-mapping name="eis/EIS" location="eis/MyEIS"/>
```

Then you can lookup the logical JNDI name of the resource adapter in your application code:

```
Context ic = new InitialContext();
DataSource ds = (DataSource)ic.lookup(java:comp/env/eis/EIS);
```

Where can I find an example of configuring and using a resource adapter on OC4J?

The following example on Oracle Technology Network shows you how to configure and use a resource adapter on OC4J:

http://www.oracle.com/technology/tech/java/oc4j/1013/how_to/how-to-jca-intro/doc/how-to-jca-intro.html

ORACLE FUSION MIDDLEWARE

Oracle Application Server Containers for J2EE 10g (10.1.3) FAQ

October 2005

Author: Frances Zhao

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Copyright © 2005, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.