

This FAQ addresses frequently asked questions relating to the Oracle Application Server Containers for J2EE 10g (10.1.3) BeanConnect for CICS 2.0:

[What is BeanConnect for CICS 2.0?](#)

[What are the components of BeanConnect for CICS 2.0?](#)

[What is J2EE Connector Architecture?](#)

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

[What is a resource adapter?](#)

[What are the system contracts?](#)

[What system contracts do BeanConnect for CICS 2.0 support?](#)

[What is CCI?](#)

[Does BeanConnect for CICS 2.0 support CCI?](#)

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

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

[What are the options for diagnostics and troubleshooting?](#)

[How do I deploy BeanConnect Resource Adapter to OracleAS or OC4J instances?](#)

[Is there tool support for developing with BeanConnect for CICS 2.0?](#)

What is BeanConnect for CICS 2.0?

Oracle Application Server BeanConnect for CICS 2.0 is a full JCA 1.5 compliant (JCA 1.5 is part of J2EE 1.4) Resource Adapter that supports connectivity to IBM CICS as EIS systems.

What are the components of BeanConnect for CICS 2.0?

BeanConnect for CICS 2.0 consists of the following components:

- The BeanConnect Resource Adapter makes the JCA interface (JCA V1.5) available to the user. It is embedded (deployed) in the J2EE application server and runs as part of the J2EE application server within the same address space.
- The BeanConnect Proxy represents the transactional connection between the resource adapter within the application server on the one hand and the CICS application on the other hand. It includes a proxy container based on the transaction monitor openUTM and a software gateway to support the protocol LU6.2.
- The BeanConnect Management Console is the Java GUI for configuring and administering the BeanConnect Proxy.

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 system contracts do BeanConnect for CICS 2.0 support?

BeanConnect for CICS 2.0 is fully compliant with JCA 1.5, and implements all of the system contracts above.

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.

Does BeanConnect for CICS 2.0 support CCI?

Yes. BeanConnect for CICS 2.0 not only supports the CCI but also offers an additional interface that clearly decreases the programming effort. See the Oracle Application Server BeanConnect for CICS 2.0 Guide (Interfaces and Programming Chapter) for details.

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.

What are the options for diagnostics and troubleshooting?

The BeanConnect Management Console provides extensive support for diagnostics and troubleshooting options for BeanConnect Resource Adapter and Proxy. Below are some examples.

Log4J logging functions:

- Log4J configuration changes (Proxy and Resource Adapter)
- Online record sampling and filtering (also remote)
- Offline record analysis based on logging in XML file

Trace control functions:

- Network Trace
- Transaction flow trace

Handling functions for diagnostic files:

- File Browser with remote browsing
- File transfer to Management System
- Display and analysis of selected files

How do I deploy BeanConnect Resource Adapter to OracleAS or OC4J instances?

The recommended way is to deploy using Oracle Application Server Console, which provides simple and easy-to-follow guide through the entire deployment and configuration process. See the Oracle Application Server Resource Adapter Administrator's Guide 10g (10.1.3) for details.

Is there tool support for developing with BeanConnect for CICS 2.0?

Yes. The award-winning Oracle JDeveloper is the recommended development tool, with comprehensive and integrated support for J2EE and Web Services including JCA based development. Tutorial and training specific to BeanConnect for CICS 2.0 are available.

ORACLE FUSION MIDDLEWARE

Oracle Application Server 10g (10.1.3) BeanConnect for CICS 2.0 FAQ

May 2006

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 © 2006, 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.