



JCA

(J2EE Connector Architecture)

Allen Long

Email: allen@huihoo.com

<http://www.huihoo.com>

2004-04



JAVA

内容安排



- JCA介绍
 - 目的
 - J2EE集成
- Common Client Interface (CCI)
 - 对象模型
 - 用例
- 系统合约
 - 连接池
 - 事务
 - 安全
- 部署 .rar
- 未来方向





J2EE连接器构架使EIS供应商为它们的EIS产品提供标准的资源适配器。资源适配器作为应用服务器（如WebLogic服务器）的插件提供了EIS与应用服务器集成的基础设施。

理想的做法是内置一个可用于任何资源类型和所有连接管理功能（包括合用）的通用连接接口。这就是J2EE Connector Architecture 1.0 规范的目标之一



JCA目的



- 为连接后台的Enterprise Information Systems (EIS)提供标准接口

- Mainframe Transaction Processing (TPs) - CICS, Tuxedo

- Enterprise Resource Planning (ERP) - SAP ,Oracle

- Database Systems - JDBC, LDAP

- Resource Adapter (RA)是一个系统级驱动程序用于连接EIS

- Resource Adapter (RA)可以被应用服务器管理

- 首要的目的是平滑的集成J2EE components和EIS systems

- 通过J2EE1.3兼容性应用服务器提供以下

- Calability (typically via Pooling, Lifecycle management, clustering)

- Distributed Transactions

- Security



合约



每个Resource Adapter (RA)支持以下三种合约

--The Application contract (may support CCI)

A JCA Resource Adapter is not required to support any specific application contract (e.g.future versions of JDBC)

--The JCA System contracts

Connection Management

Transaction Management

Security Management

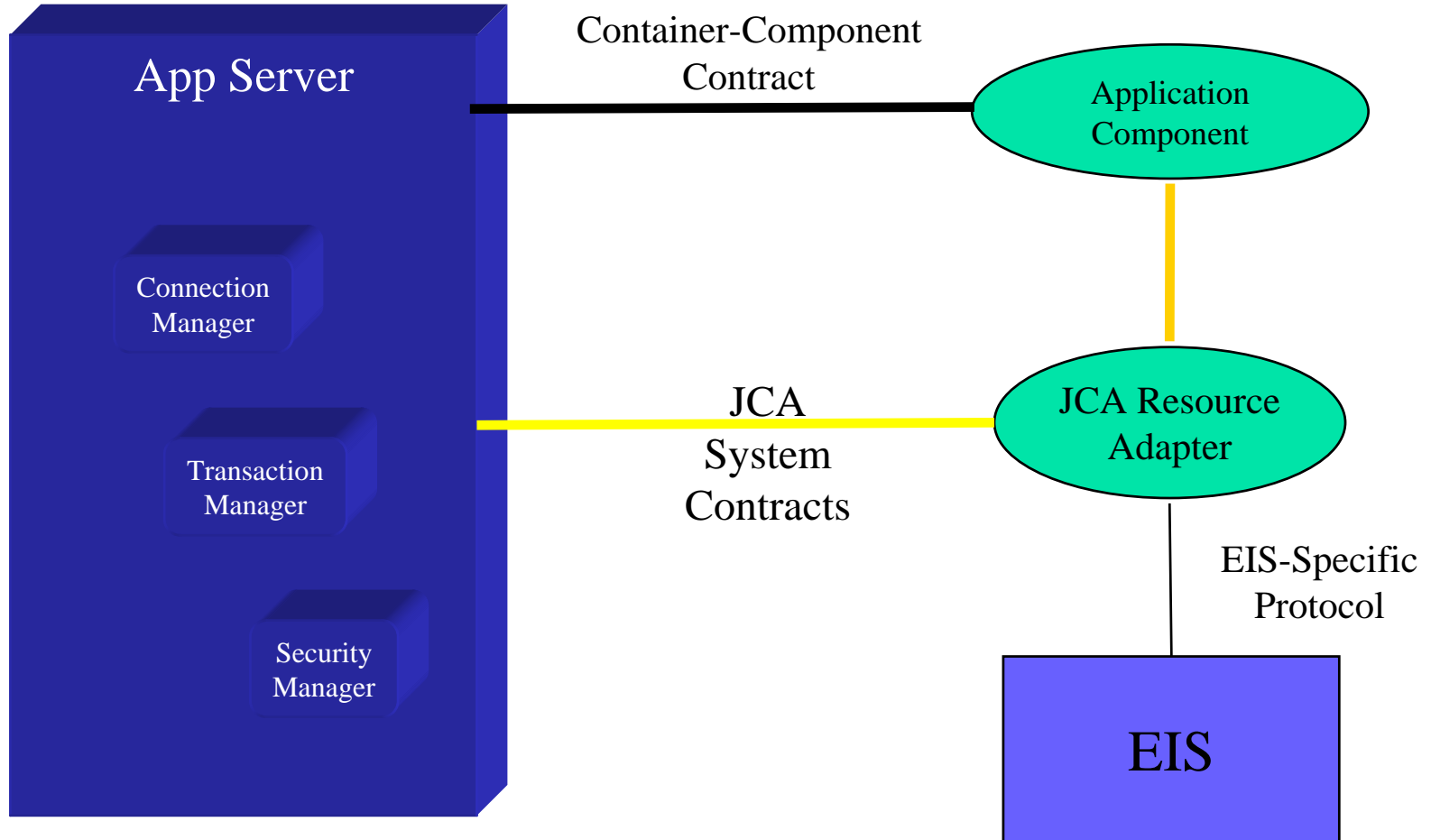
--The EIS protocol contract (typically proprietary)

每个J2EE JAR可以看作是一个合约



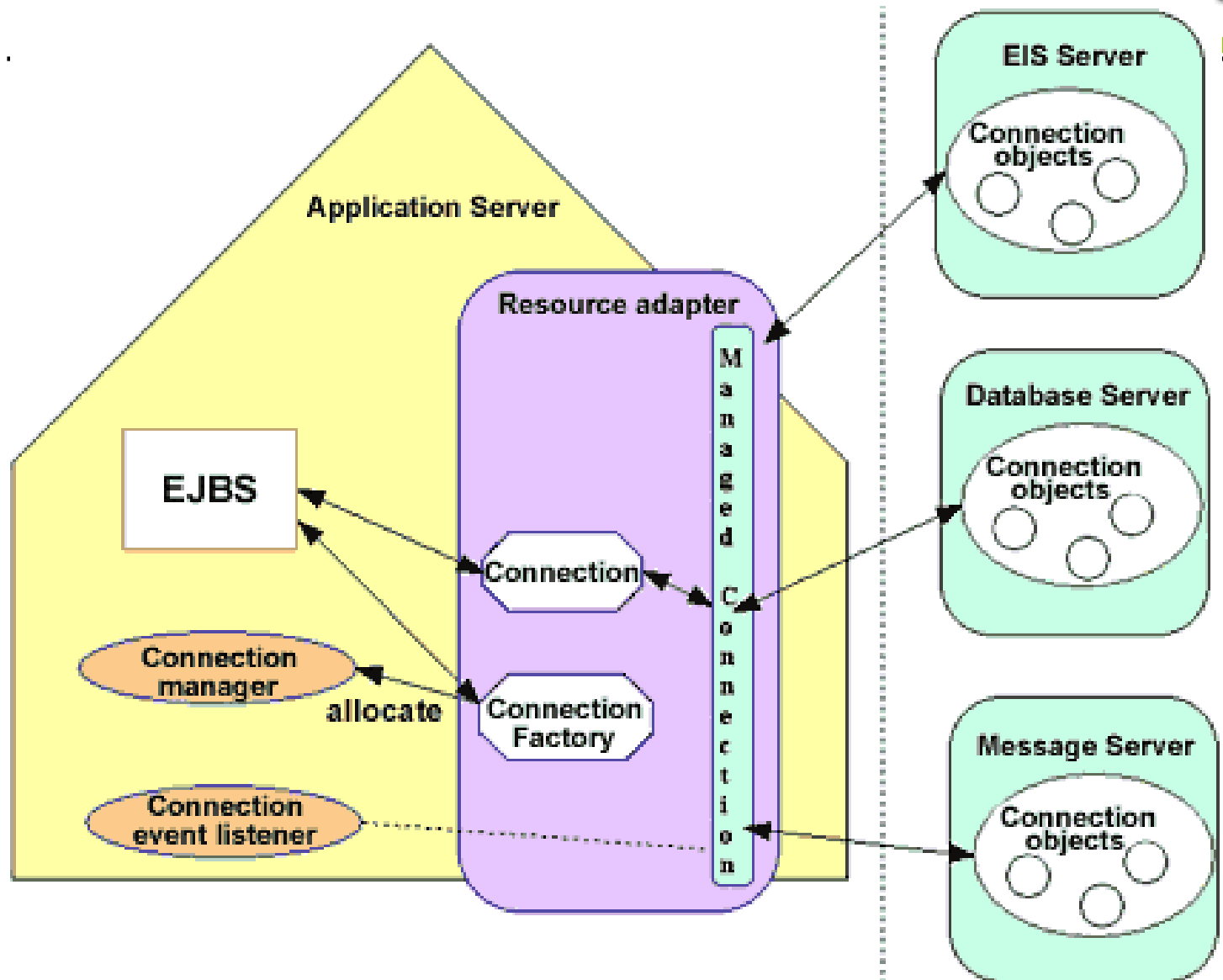
JAVA

一个高层次的JCA架构



JAVA

JCA架构





JAVA

应用合约适应于通过Resource Adapter (RA)进行用户间的相互操作

--不管来自J2EE组件或是standalone

理想状态下,合约扩展了以下一些细节:

--Called the “Common Client Interface” or CCI

--Interfaces provide standardization for:

- obtaining connections

- submitting synchronous requests for record-based data

- querying metadata

- controlling local transactions

支持CCI并不是必须的

--供应商可能提供全部的属性接口

JDBC和JCA关系



大多数应用程序开发人员不需要知道 JDBC 和 J2EE 连接器体系结构之间的关系，就可以很好地使用 JDBC API。但是，由于 JDBC 3.0 规范已经考虑到这项新的体系结构，这使得开发人员能更好地理解 JDBC 在哪里适合 J2EE 标准，以及这个规范的发展方向是什么。

JCA指定了一组协议，允许企业的信息系统以一种可插入的方式连接到应用服务器上。这种体系结构定义了负责与外部系统连接的资源适配器。连接器服务提供者接口（The Connectors Service Provider Interface, SPI）恰好和 JDBC 接口提供的服务紧密配合。

JDBC API 实现了连接器体系结构定义的两个协议中的两个。

第一个是将应用程序组件与后端系统相连接的**连接管理**，它是由 DataSource 和 ConnectionPoolDataSource 接口来实现的。

第二个是支持对资源的事务性访问的**事务管理**，它是由 XADataSource 来处理的。

第三个是支持后端系统的安全访问的**安全性管理**，在这点上，JDBC 规范并没有任何对应点。尽管有最后那个不足，JDBC 接口仍能映射到连接器 SPI 上。

如果一个驱动程序厂商将其 JDBC 驱动程序映射到连接器系统协议上，它就可以将其驱动程序部署为资源适配器，并立刻享受可插性、封装和在应用服务器中部署的好处。这样，一个标准的 API 就可以在不同种类的企业信息系统中，供企业开发人员使用。

JDBC和JCA关系



Function	JDBC (java.sql)	CCI (javax.resource.cci)
Create Connection	javax.sql.DataSource	ConnectionFactory + ConnectionSpec
Create Commands	Connection	Connection
Format Commands	Statement	Interaction + InteractionSpec
Obtain Results	ResultSet	RecordFactory & Records



"Spec" classes



CCI包含了ConnectionSpec用于传输任意的信息

--An empty interface

--Implementation defines properties following JavaBean conventions (get/set)

--Standard Properties: UserName, Password

Interaction Spec

--connection.getInteraction() takes an InteractionSpec argument

--Also a JavaBean with standard properties

FunctionName

ExecutionTimeout

FetchSize

InteractionVerb - synchronous actions

 SYNC_SEND

 SYNC_SEND_RECEIVE (default)

 SYNC_RECEIVE

ResultSetType - java.sql.ResultSet (FORWARD, SCROLL_SENSITIVE)

ResultSetConcurrency - java.sql.ResultSet (READ_ONLY, UPDATABLE)

--可能作为一个resource environment reference绑定到 JNDI上 (就象 JMS Topics一样)

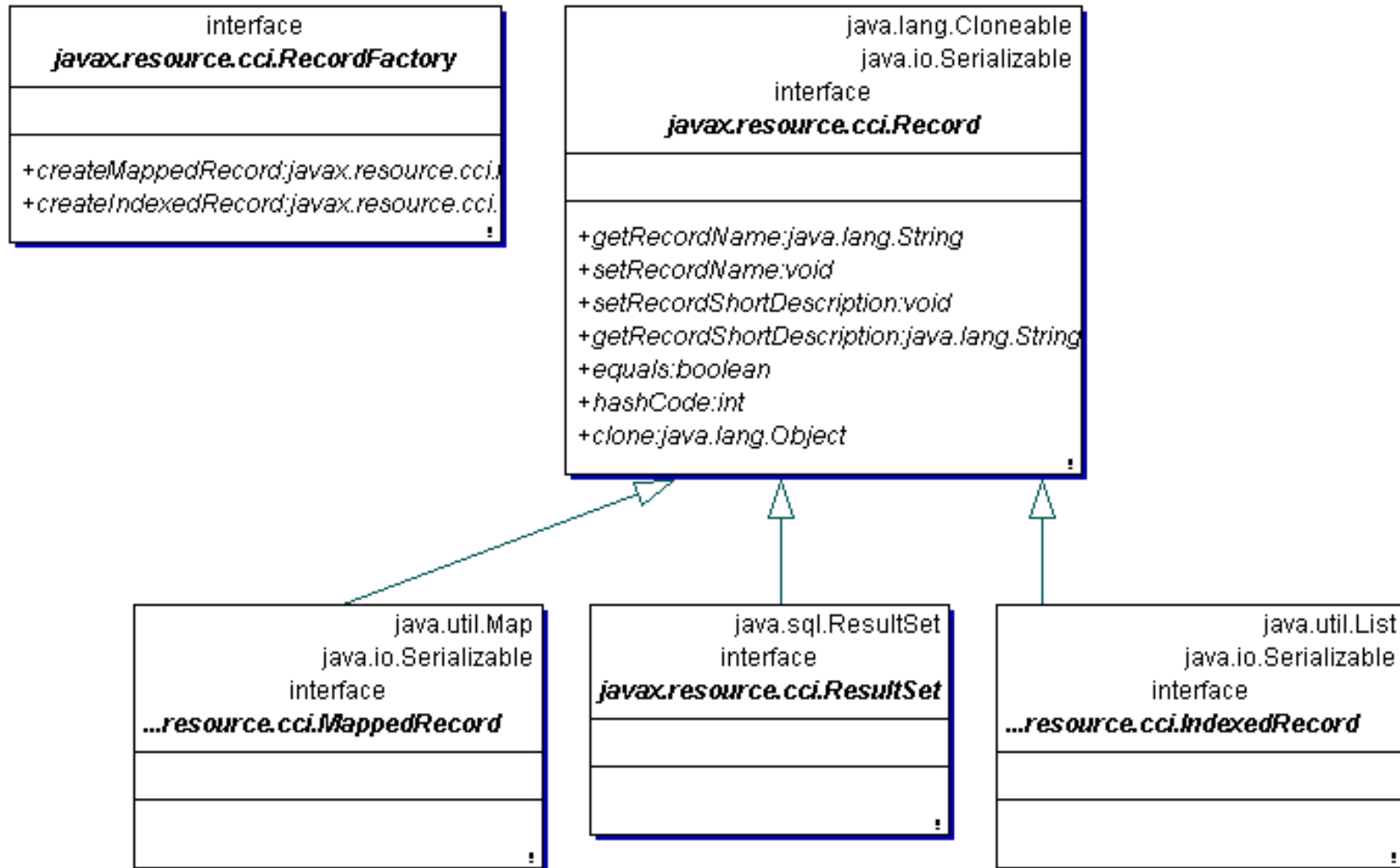


```
public interface
javax.resource.cci.InteractionSpec
extends Serializable {
    public static final int SYNC_SEND =
0;
    public static final int
SYNC_SEND_RECEIVE = 1;
    public static final int SYNC_RECEIVE
= 2;
}
```

Records



Interaction使用Records对象调用和接收响应的数据



Records



JAVA

- Records可能是通过RecordFactory接口实现创建的。
 - The MappedRecord is a Map
 - The IndexedRecord is a List
 - Either may contain hierarchical structures of Records
- 工厂方法create()将带一个recordName的参数
 - The factory will utilize this name to reference meta information related to the construction of the record type
- RA可能包含附加的方法来创建客户记录

ResultSet



JAVA

- CCI ResultSet 扩展了JDBC ResultSet
--Provides similar support for scrolling, type mapping, updatability
- CCI ResultSet可能不支持大多高级JDBC类型
--Blob, Clob, Array, Ref, Distinct, Struct, customized mapping
- Java.sql.ResultSetMetaData is reused(重用)
--Query column name, type, etc.
- ResultSetInfo provides information on the supported ResultSet functionality
--e.g. visibility of updates and inserts

CCI Example - CICS Connector



```
// STANDALONE
//create and set values for a managed connection factory for EPI
EPIManagedConnectionFactory mcf =new
    EPIManagedConnectionFactory();
cf.setConnectionURL("tcp://gunner.almaden.ibm.com");
cf.setServerName("SCSCPAA6");
ConnectionFactory
    cxf=(ConnectionFactory)mcf.createConnectionFactory();

// or J2EE (using Environment Naming Context)
InitialContext ctx = new InitialContext();
ConnectionFactory cxf =
    (ConnectionFactory)ctx.lookup("java:comp/env/jca/myConnector
");

//create a connection object
Connection connection =cxf.getConnection();

//create an interaction with CICS to start transaction EPIP
Interaction interaction =connection.createInteraction();
```

May pass
ConnectionSpec to
define identity,
context, etc.



CCI Example - CICS Connector



```
// Implements InteractionSpec
EPIInteractionSpec iSpec =new EPIInteractionSpec();
iSpec.setFunctionName("EPIP"); // well-known property
iSpec.setAID(AIDKey.enter); // vendor-specific property
iSpec.setInteractionVerb(InteractionSpec.SYNC_SEND_RECEIVE); // default

// Create a record to store the response from CICS - implements Record
// Can create records in this way or through RecordFactory implementations
// obtained via ConnectionFactory.getRecordFactory()
EPIScreenRecord screen =new EPIScreenRecordImpl();

// execute w/ InteractionSpec, InputRecord, OutputRecord
boolean rc =interaction.execute(iSpec,null,screen);

//close the interaction and connection
interaction.close();
connection.close();
```



连接管理



- 连接管理合约允许RA利用应用服务器支持connection pooling
 - May manage their own pools in a non-managed environment
- 应用组件(EJB, Servlet, Client)通过Environment Naming Context (ENC)在JNDI中找到connections工厂.
- 组件请求一个connection handle - possibly passing additional information

```
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.ConnectionFactory;
import javax.naming.InitialContext;
...
InitialContext ctx = new InitialContext();
ConnectionFactory fcty =
(ConnectionFactory)ctx.lookup("java:comp/env/eis/myEIS");
Connection conn = fcty.getConnection(new MyEISConnectionSpec(data));
```



JAVA

连接管理



- 多连接handles可能访问同样的ManagedConnection(physical connection)

- 应用服务器将检验这些连接以确定其中的一部分能满足这些请求

< res-sharing-scope >

- 应用服务器在RA上带以下参数调用matchManagedConnection()

- Information passed to getConnection() (via a ConnectionSpec if CCI)

- Security information for the current principal (javax.security.auth.Subject)

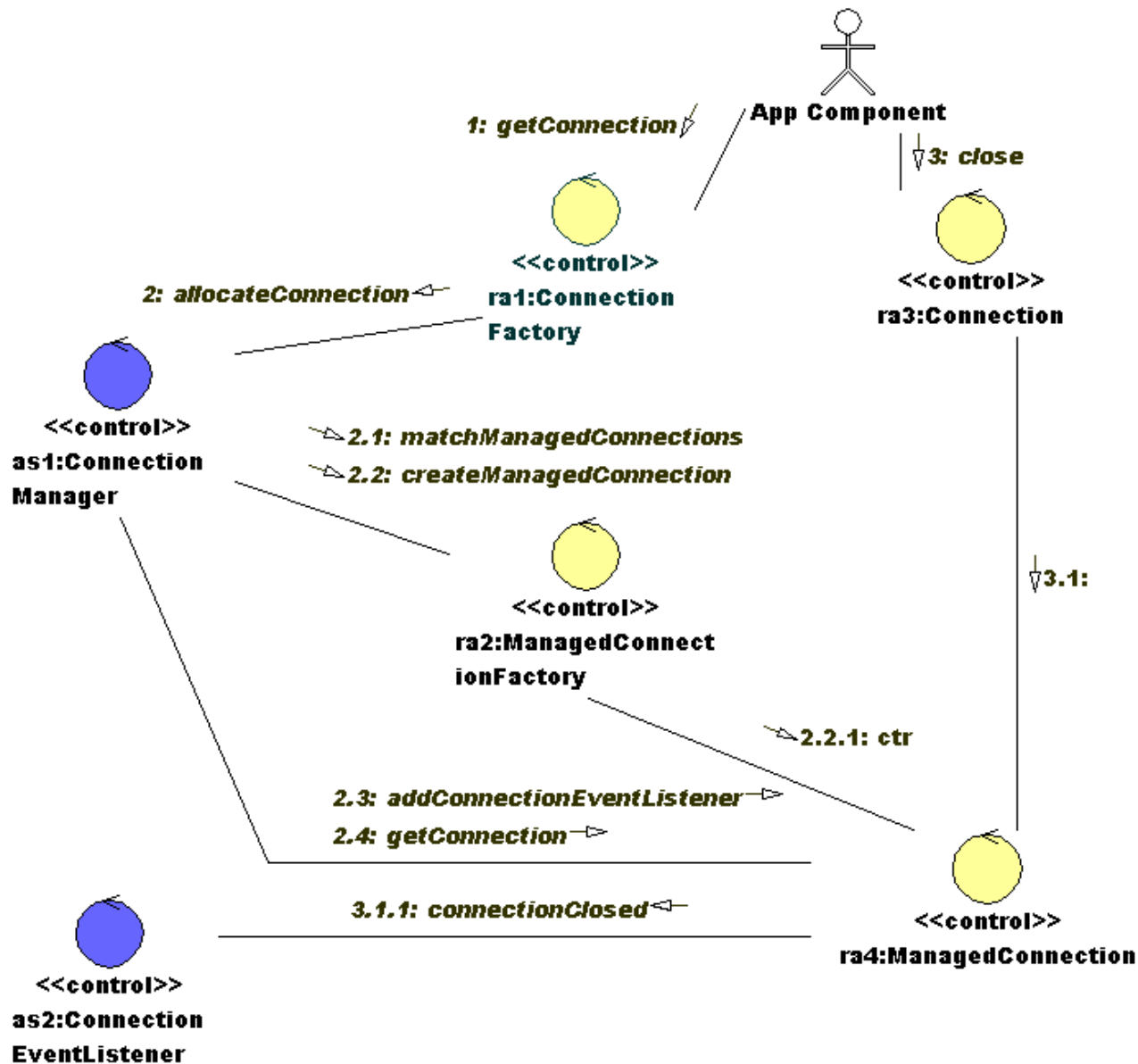
- A subset of active connections (based on app server logic)

- 如果没有匹配的连接,应用服务器将创建新的连接

- 在超时时应用服务器可能在一个ManagedConnections上关连了很多的连接handles.(见associateConnection())



连接管理Interaction



事务管理合约



- J2EE区分两种类型事务和合约
- resource manager的外部事务控制器是一个XA或JTA事务资源管理器可能只支持XA事务的一段式提交
- resource manager的内部事务控制器是一个Local事务
- 一个适配器可能不支持,或支持两种,或支持一种事务
<transaction-support>

```
public interface javax.resource.spi.ManagedConnection {  
    ...  
    javax.transaction.xa.XAResource getXAResource() throws  
    ResourceException;  
    javax.resource.spi.LocalTransaction getLocalTransaction() throws  
    ResourceException;  
    ...  
}
```





- 如果资源管理器不支持两段式提交的XA事务
 - The same thread of execution (involving one or more app components) may only utilize a single non-XA RA
 - In general, use of non-XA and XA capable RAs may not be mixed in the same thread of execution
 - Local transactions may not span container boundaries (J2EE 1.3 spec)
- 通过使用Local事务可以获得较高性能
 - But how does server know a priori what resource managers will be involved in a transaction ?
 - Some advantage may be achieved through 1-PC optimization
 - 2PC “last resource optimization” may be provided by some app servers
- 任意数量的非事务资源管理器可能混合了这些事务



JAVA

XA Transaction Management



- XA事务管理器在J2EE中有两种方法启动和停止(commit or rollback)
 - By the container based on deployment descriptor settings
 - Explicitly by component code (Using JTA UserTransaction)
- 对 workflow、任务和管理活动等建模（如订房、购物车等）
- 协调多个实体bean，控制实体bean之间的交互
- 将业务应用逻辑从客户端转移到服务器端



Local事务管理



Local事务管理既可以是容器管理的,也可以是组件管理的

--Components manage local transactions using a RA-specific API (typically implementing `javax.resource.cci.LocalTransaction`)

--The `LocalTransaction` instance is obtained from the connection contains only `start()`, `commit()`, `rollback()`

--In situations of component-managed local transactions the app server is notified of transaction events via the `ConnectionEventListener`



```
public interface javax.resource.spi.ConnectionEventListener {  
    ...  
    void localTransactionStarted(ConnectionEvent event);  
    void localTransactionCommitted(ConnectionEvent event);  
    void localTransactionRolledback(ConnectionEvent event);  
    ...  
}
```



安全合约



- 应用组件提供了两种EIS sing-on方式
 - Container -> Authentication details defined by the deployer
 - Application -> Passed to getConnection() by component code
 - Choice is defined in component deployment descriptor <res-auth> element
- 适配器在它们的描述中常提供默认的设置
- 应用级的认证信息直接传递到RM
 - Opaque to the connector architecture
- 容器级的认证信息是下面两种方式中一种
 - PasswordCredential
 - GenericCredential
- The type passed to the adapter is defined by its deployment descriptor - <credential-interface>

容器认证(Authentication)



- 。 JCA通过JAAS APIs传递认证信息
- 。 应用服务器创建一个javax.security.auth.Subject对象(它包含a single principal and an instance of the correct Credential type)
- 。 The Subject instance is passed to the ManagedConnectionFactory matchManagedConnection() or createManagedConnection() methods by the app server
- 。 The app server determines the contents of the Credential based on:
 - The principal identification in effect at the time of getConnection()
 - »May be different from the authenticated user (e.g. EJB 2.0 runAs)
 - <authentication-mechanism-type> element(s)
 - deployment settings for mapping of principal identifications



Resource Principal Identification



- The resource principal identity for the adapter-EIS communication may be established by the deployer in various ways:
 - Configured Identity (static)
 - »Identity is defined in descriptor
 - Principal Mapping
 - »Each resource has a mapping of component principal -> resource principal
 - Caller Impersonation
 - »Resource principal acts on behalf of the caller principal
 - »Principal delegation specific to security mechanism (e.g. Kerberos)
 - Credentials Mapping
 - »The principal identity remains the same but credentials are mapped to a different authentication domain



缓冲管理



- 。 Current JCA architecture includes minimal contracts for a Caching Manager
- 。 The caching manager registers interest in transaction events
 - Similar to the EJB SessionSynchronization interface
 - beforeCompletion() - just prior to commit()
 - afterCompletion() - notifies whether transaction committed or rolledback
- 。 The caching manager services CCI queries and resyncs with the EIS when appropriate based on the above callbacks
- 。 Connector-based JDO implementations are Caching Managers



打包和部署



- Resource adapters are packaged in jar files similar to application components (suffix is .rar)
- The Resource Adapter Archive contains all java and native files required to execute
- Also contains the RAR deployment descriptor - ra.xml
- Each application server typically provides a schema for an additional proprietary deployment descriptor
 - JNDI bind location
 - Details of security mapping



JAVA

部署描述符中的内容



• Class file identification

- <connection-interface> and <connection-impl-class>
- <connectionfactory-interface> and <connectionfactory-impl-class>
- <managedconnectionfactory-class>

• Security Settings

- <authentication-mechanism-type>* - BasicPassword, Kerbv5, etc.
- <credential-interface> - PasswordCredential or GenericCredential
- <reauthentication-support> - Can ManagedConnections change principal ?
- <security-permission> - special security permissions (standard “grant” .policy syntax)
 - »use privileged code blocks to exercise

• Misc

- <license-required>
- <config-property> - misc configuration values



JAVA

未来的方向



• Specification

- Pluggability of JMS Providers
- Thread Management Contract
 - »Needed to allow asynchronous communication with EIS
 - »Finally have the ability to create threads (!?)
- CCI support requirement
- CCI XML support
 - »Metadata - Query Interactions and Record types available
 - »Submit Interactions as XML

• Industry

- Entity Bean CMP mapping to EIS ?
- Greater variety and ease of use of EIS
- Ability of new technologies to integrate with J2EE
 - »JDO ?



JAVA

供应商



- Insevo
 - CICS, IMS, JD Edwards, Baan, SAP, Siebel, PeopleSoft
 - Support CCI and XML-based interface/schema repository
 - Bi-directional communication
- Resource Adapters, Inc
 - SAP, PeopleSoft, Siebel, Oracle
 - Also support XML interaction and Bi-directional communication
- SolarMetric, PrismTech, FastObjects, Versant
 - JDO Implementations w/JCA support
- Sonic
 - Can plug RAs into SonicXQ as services

内容回顾



- JCA介绍
 - 目的
 - J2EE集成
- Common Client Interface (CCI)
 - 对象模型
 - 用例
- 系统合约
 - 连接池
 - 事务
 - 安全
- 部署 .rar
- 未来方向



总结



- JCA使JAVA真正步入应用集成时代
- JCA使JAVA程序能连接到非JAVA程序和应用软件包
- JCA是JAVA顺利进入已有大型企业应用系统(ERP, CRM, SCM)的关键
- JCA是J2EE在企业级应用站稳脚的最后机会?



参考资料



- <http://java.sun.com/j2ee/connector/>
sun公司的JCA站点
- <http://www.huihoo.com>
国内一个关于中间件的专业站点



结束



谢谢大家！

Allen@huihoo.com

<http://www.huihoo.com>

