JCA
(J2EE Connector Architecture)

Allen Long
Email: allen@huihoo.com
http://www.huihoo.com
2004-04

- JCA

J2EE
- Common Client Interface (CCI)

- 　　　.rar
-

JCA

J2EE   EIS   EIS

　　WebLogic   EIS

J2EE Connector Architecture 1.0

JCA

- Enterprise Information Systems (EIS)

--Mainframe Transaction Processing (TPs) - CICS, Tuxedo

--Enterprise Resource Planning (ERP) - SAP ,Oracle

--Database Systems - JDBC, LDAP

- Resource Adapter (RA)   EIS

- Resource Adapter (RA)

--   J2EE components EIS systems

--   J2EE1.3

        Calability (typically via Pooling, Lifecycle management, clustering)

        Distributed Transactions

        Security

Resource Adapter (RA)

--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
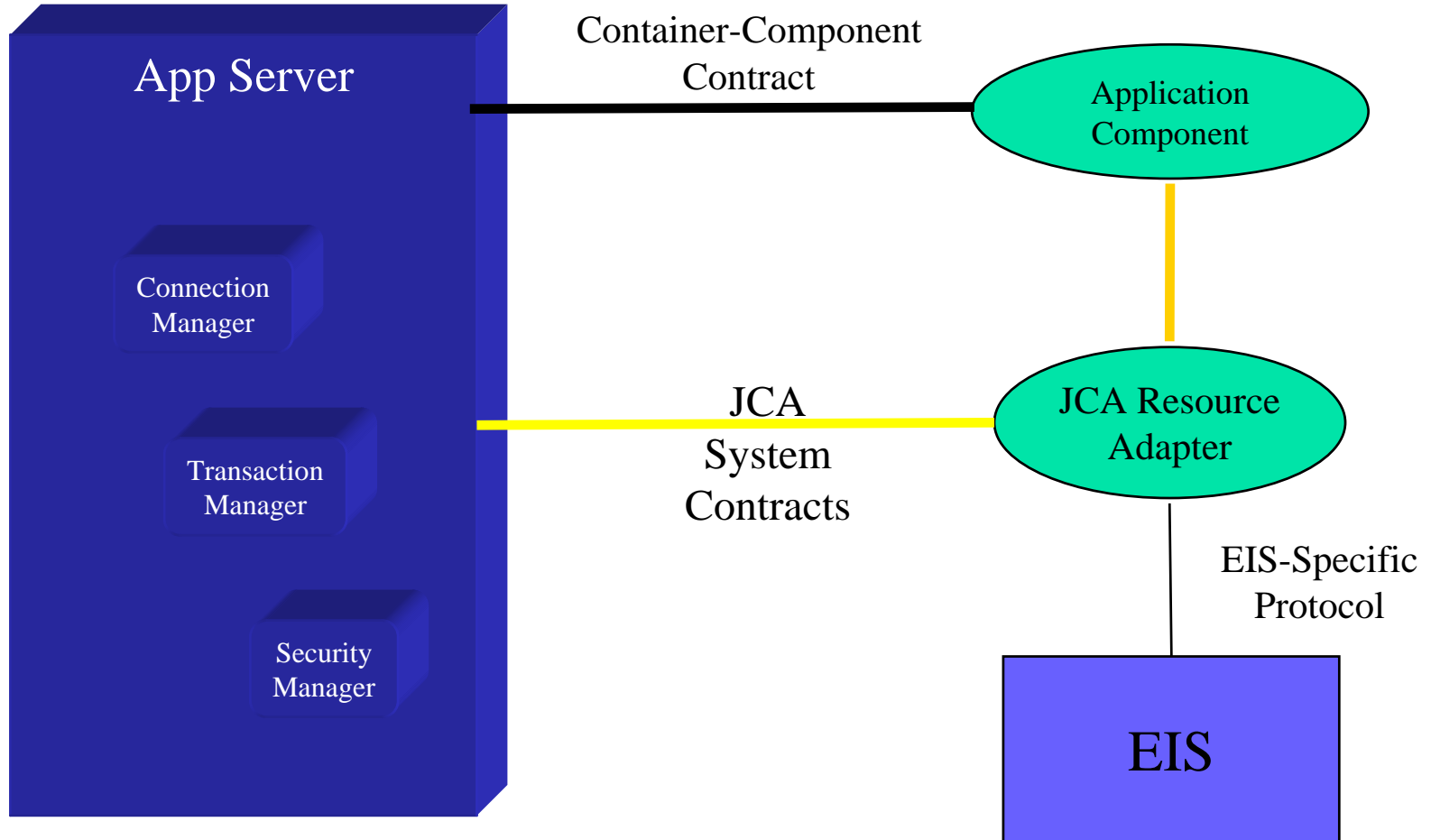
Transaction Management

Security Management

--The EIS protocol contract (typically proprietary)
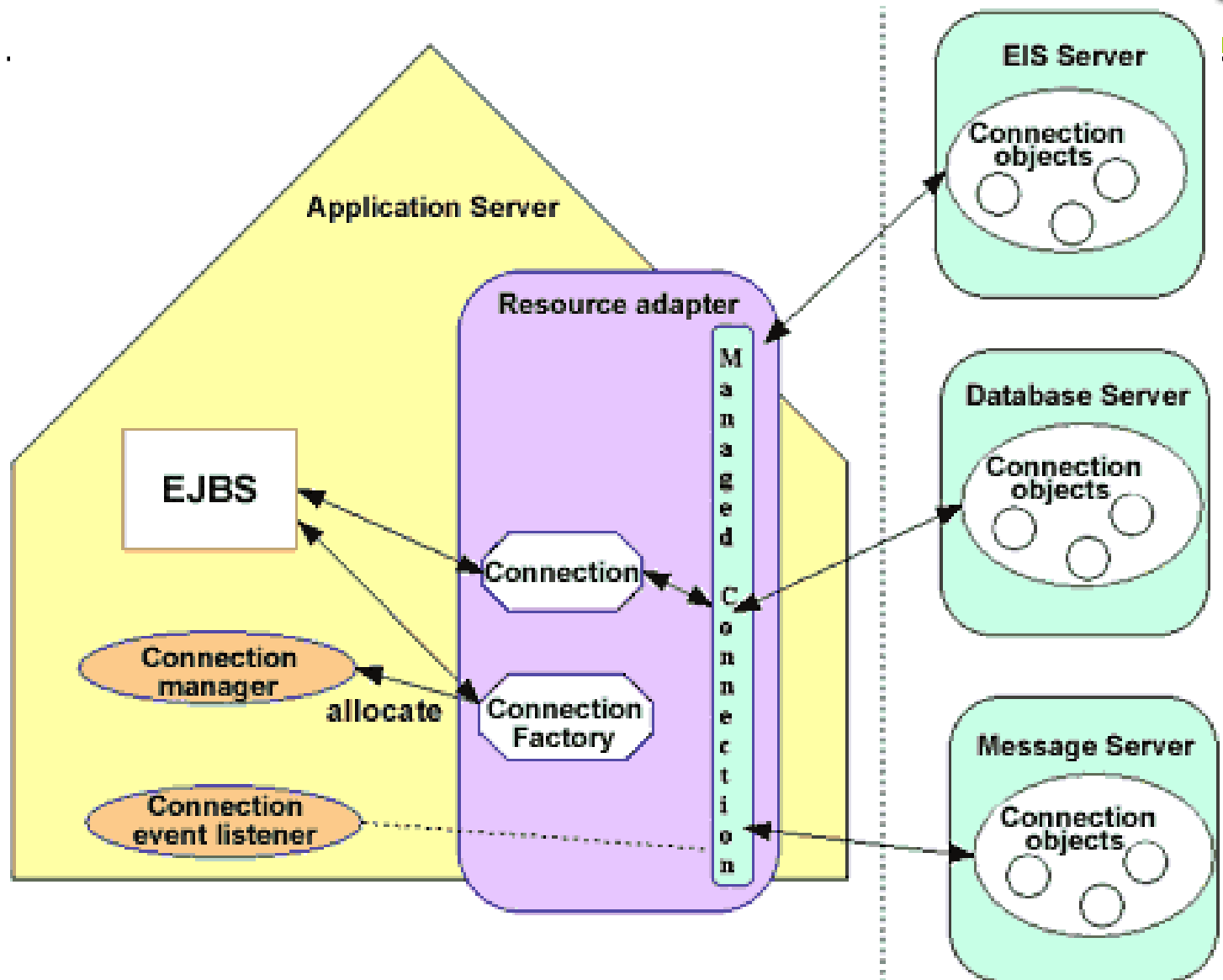
J2EE JAR

JCA

App Server

Container-Component
Contract

Application
Component

Connection
Manager

JCA
System
Contracts

JCA Resource
Adapter

Transaction
Manager

EIS-Specific
Protocol

Security
Manager

EIS

JCA

Resource Adapter (RA)

--　　　　　J2EE　　　　　standalone

　　　　　　, 　　　　　　　　　　　　　　**:**

--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

　　CCI

--

JDBC JCA

JDBC J2EE
JDBC API JDBC 3.0
JDBC J2EE

JCA

The

Connectors Service Provider Interface SPI JDBC

JDBC API

DataSource

ConnectionPoolDataSource

XADataSource

JDBC

JDBC SPI

JDBC

API

JDBC JCA

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 ConnectionSpec

--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

```
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;
}
```
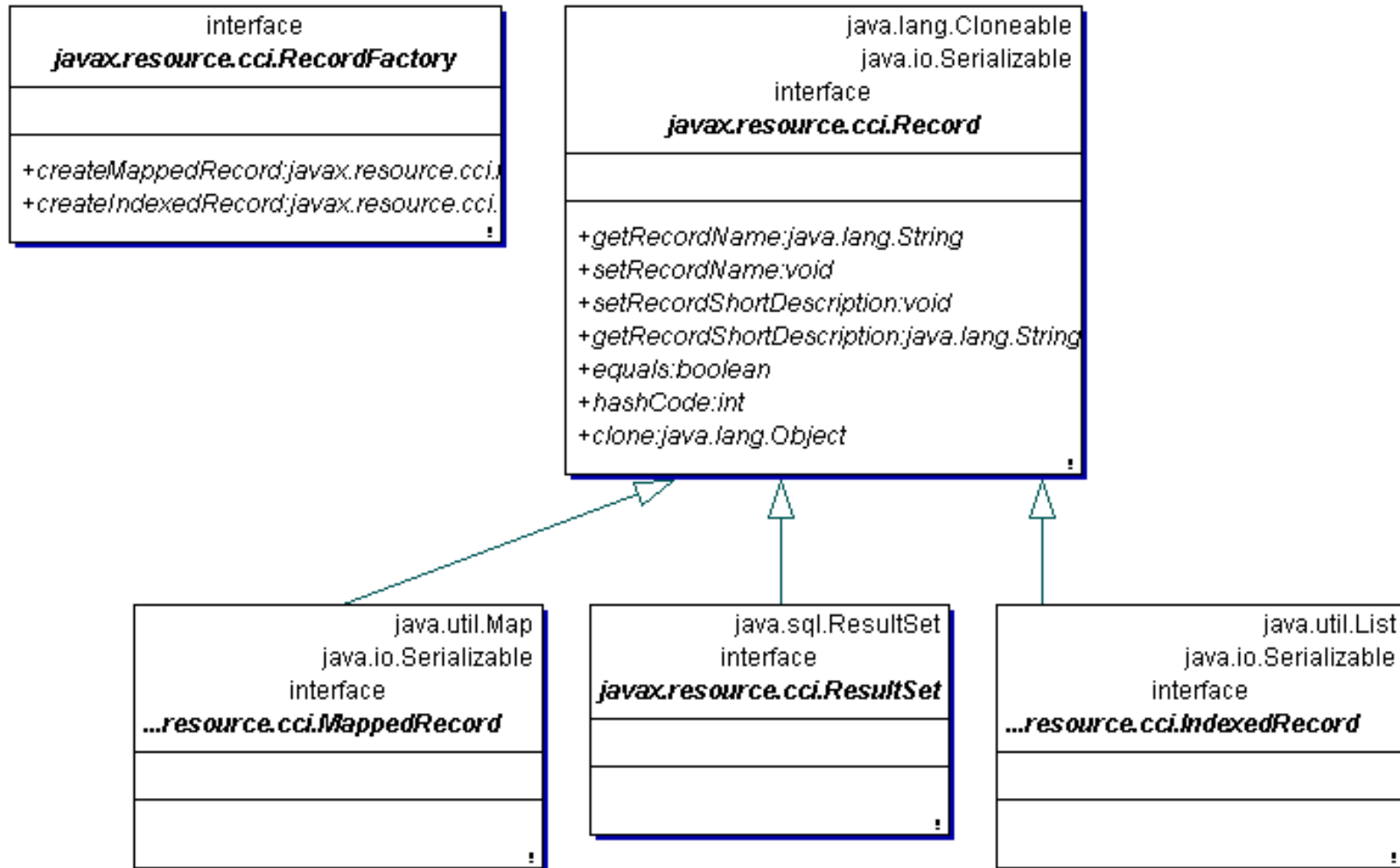
ResultSetType - java.sql.ResultSet (FORWARD, SCROLL_SENSITIVE)

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

-- resource environment reference JNDI (JMS Topics)

Records

interface
javax.resource.cci.RecordFactory

+createMappedRecord:javax.resource.cci.
+createIndexedRecord:javax.resource.cci.

java.lang.Cloneable
java.io.Serializable
interface
javax.resource.cci.Record

+getRecordName:java.lang.String
+setRecordName:void
+setRecordShortDescription:void
+getRecordShortDescription:java.lang.String
+equals:boolean
+hashCode:int
+clone:java.lang.Object

java.util.Map
java.io.Serializable
interface
...resource.cci.MappedRecord

java.sql.ResultSet
interface
javax.resource.cci.ResultSet

java.util.List
java.io.Serializable
interface
...resource.cci.IndexedRecord

- Records RecordFactory .

--The MappedRecord is a Map

--The IndexedRecord is a List

--Either may contain hierarchical structures of Records

- create() recordName

--The factory will utilize this name to reference meta information related to the construction of the record type

- RA

ResultSet

- CCI ResultSet   JDBC ResultSet

--Provides similar support for scrolling, type mapping, updatability

- CCI ResultSet   JDBC

--Blob, Clob, Array, Ref, Distinct, Struct, customized mapping

- Java.sql.ResultSetMetaData is reused(   )

--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

```java
// 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();
```

- RA   connection pooling

--May manage their own pools in a non-managed environment

- (EJB,Servlet,Clinet)   Environment Naming Context (ENC) JNDI   connections   .

- 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));
```
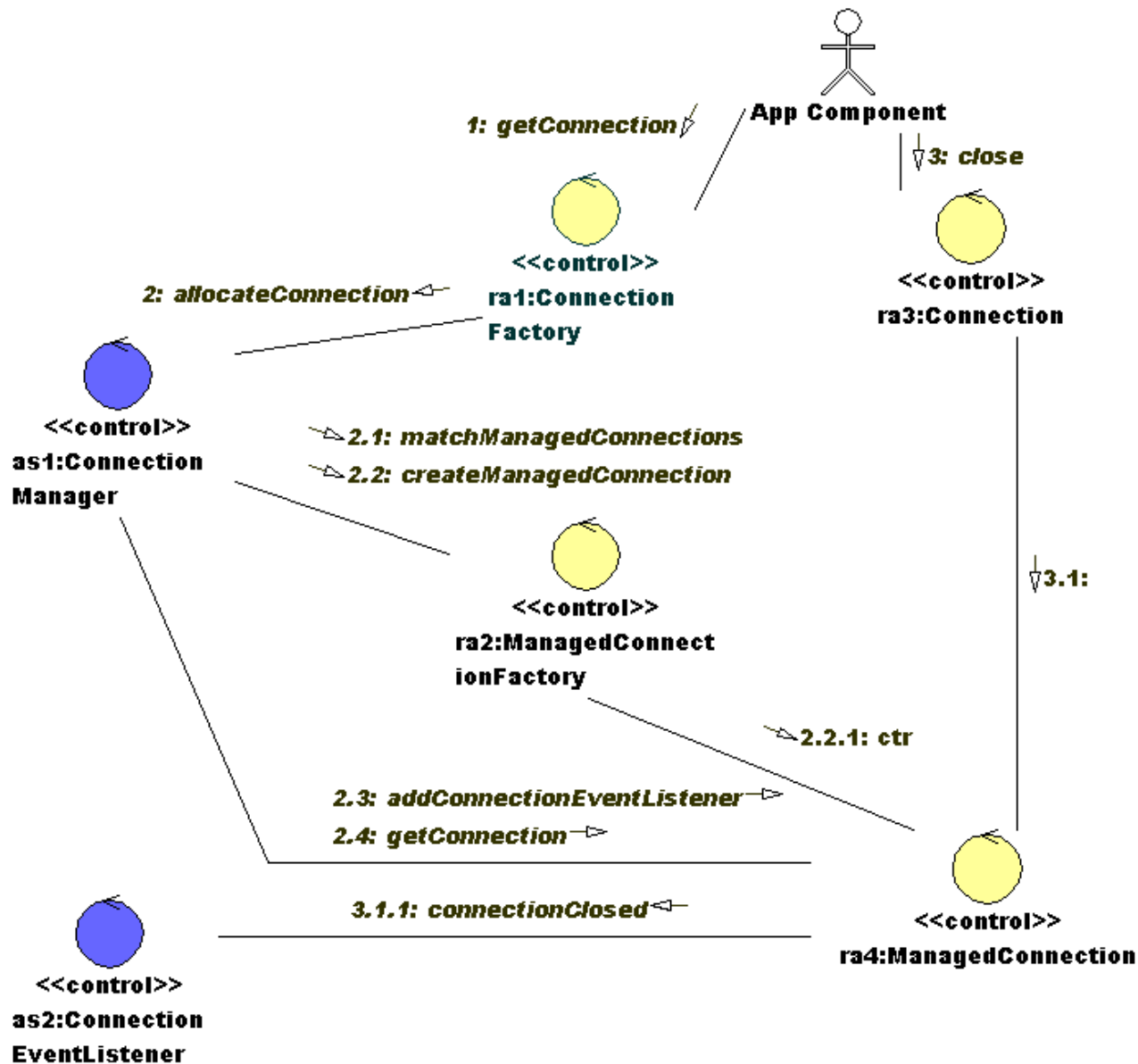
- handles ManagedConnection(physical connection)

-

< res-sharing-scope >

- RA 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)

- ,

- ManagedConnections handles.(associateConnection())

Interaction

1: getConnection

App Component

3: close

2: allocateConnection

<<control>>
ra1:Connection
Factory

<<control>>
ra3:Connection

<<control>>
as1:Connection
Manager

2.1: matchManagedConnections
2.2: createManagedConnection

<<control>>
ra2:ManagedConnect
ionFactory

3.1:

2.2.1: ctr

2.3: addConnectionEventListener
2.4: getConnection

3.1.1: connectionClosed

<<control>>
ra4:ManagedConnection

<<control>>
as2:Connection
EventListener

- J2EE

- resource manager　　　　　　　　　　　　　　　　　　XA　JTA

  　　　　　　　　　　　　　　XA

- resource manager　　　　　　　　　　　　　　　　Local

- 　　　　　　　　　　　　　　　，　　　　　，
  <transaction-support>

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

- **XA**
  - 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)
- **Local**
  - 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

XA Transaction Management

- XA   J2EE   (commit or rollback)

--By the container based on deployment descriptor settings

--Explicitly by component code (Using JTA UserTransaction)

-

-          bean   bean

-

Local

Local   ,

--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);
...
}
```

- EIS sing-on

--Container -> Authentication details defined by the deployer

--Application -> Passed to getConnection() by component code

--Choice is defined in component deployment descriptor <res-auth> element

-

- RM

--Opaque to the connector architecture

-

--PasswordCredential

--GenericCredential

- The type passed to the adapter is defined by its deployment descriptor - <credential-interface>

JCA   JAAS APIs

javax.security.auth.Subject   (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 <u>contents</u> 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

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

- 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

- **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

- 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 ?

- 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

- JCA

  J2EE
- Common Client Interface (CCI)

-

- .rar
-

- JCA JAVA

- JCA JAVA JAVA

- JCA JAVA
  (ERP, CRM, SCM)

- JCA J2EE ?

- http://java.sun.com/j2ee/connector/
 sun   JCA

- http://www.huihoo.com

Allen@huihoo.com
http://www.huihoo.com