Tables

FIGURE 12-1

Code Samples

CODE EXAMPLE 5n13Tw3/F2 1 T96.96 0 96.9149.1 66 416.22 Tm.0037 Tm.230 TwSaSamp Resourdap5)7(er)1001-19es

1.1 Overview

The Java EE Connector architecture defines a standard architecture for connecting

- **Lifecycle management contract.** A contract between an application server and a resource adapter that allows an application server to manage the lifecycle of a resource adapter. This contract provides a mechanism for the application server to bootstrap a resource adapter instance during its deployment or application server startup, and to notify the resource adapter instance during its undeployment or during an orderly shutdown of the application server.
- **Work management contract.** A contract between an application server and a resource adapter that allows a resource adapter to do work (monitor network endpoints, call applic0 Tw[(10ract)-5.3(.)om.6(licone, caletc.) bhutubmitt)4.33

The connector architecture defines a standard SPI (Service Provider Interface) for integrating the transaction, security, and connection management facilities of an application server with those of a transa

1.11 Connector Architecture Version 1.0 Contributor Details

1.11.1 Expert Group Details (JSR-16)

Refer to the URL http://www.jcp.org/en/jsr/detail?id=16 for details on JSR-16. The following are part of the expert group and have made invaluable contributions to the Connector architecture specification:

1.11.2 Acknowledgements

2.1.2 Connector Architecture

2.1.5 Managed Environment

A managed environment defines an operational environment for a Java EE-based, multi-tier, web-enabled application that accesses EISs. The application consists of one or more application components—EJBs, JSPs, servlets—which are deployed on

If there are *m* application servers and *n* EISs, the connector architecture reduces the scope of the integration problem from an *m* x *n* problem to an *m* + *n* problem.

FIGURE 2-1 System Level Pluggability Between Application Servers and EISs

2.2.2 Common Client Interface

An enterprise tools vendor provides tools that lead to a simple application programming model for EIS access, thereby reducing the effort required in EIS integration. An EAI vendor provides a framework that supports integration across multiple EISs. Both types of vendors need to integrate across heterogeneous EISs.

Each EIS typically has a client API that is specific to the EIS. Examples of EIS client APIs are RFC for SAP R/3 and ECI for CICS.

An enterprise tools vendor adapts different client APIs for target EISs to a common client API. The adapted API is typically specific to a tools vendor and supports an

The connector architecture provides a solution for the *m*

CHAPTER

pooling if done by the resource adapter internally) by using the low-level APIs exposed by the resource adapter. This model is similar to the way a two-tier JDBC application client accesses a database system in a non-managed environment.

3.5 Standalone Container Environment

CHAPTER 4

4.1.2 Application Server Vendor

The application server vendor provides an implementation of a Java EE-compliant application server that provides support for component based enterprise

- **Application composition tool**

TPSoft Inc. is another enterprise system vendor that provides a transaction processing (TP) system. TPSoft has also developed a standard resource adapter for its TP system. The resource adapter library supports CCI as part of its implementation.

AppServer Inc. is a system vendor that has an application server product which supports the development and deployme

After deploying and successfully testing th

5.2 Goals

- Provide a mechanism for an application server to manage the lifecycle of a resource adapter instance.

5.3 Lifecycle Management Model

FIGURE 5-1 Lifecycle Management Contract (Interfaces)

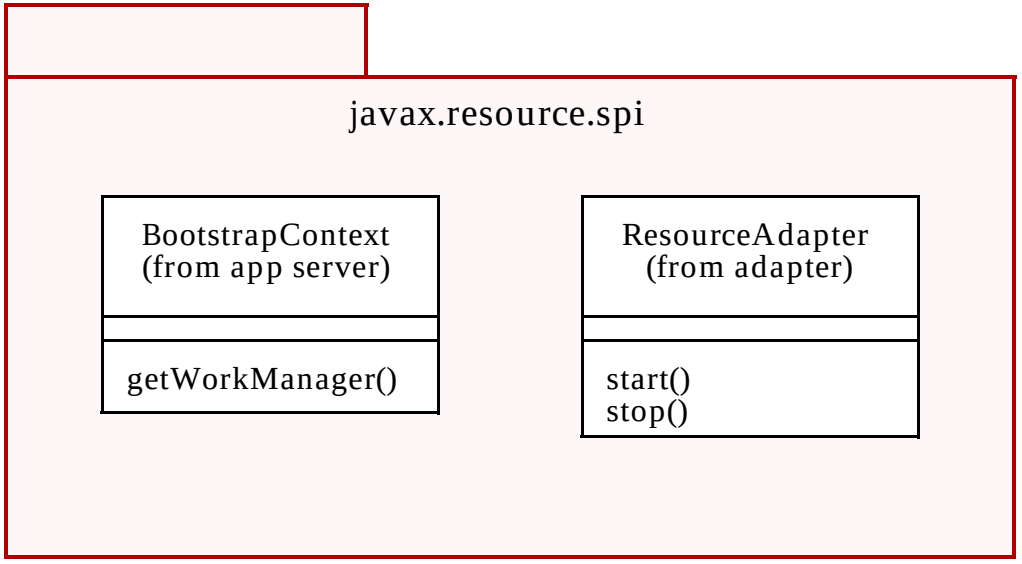


FIGURE 5-2 Lifecycle Management (Object Diagram)

instance. The application server must create at least one functional resource adapter instance per resource adapter deployment. An application server may create more

A resource adapter instance may provide bi-directional connectivity to multiple EIS

The

5.3.7.1 `ResourceAdapter` JavaBean Instance Configuration

- Create a `ResourceAdapter` JavaBean instance. This will initialize the instance with the defaults specified by way of the JavaBean mechanism.
- Apply the `ResourceAdapter` class configuration properties specified in the resource adapter deployment descriptor, on the `ResourceAdapter` instance. This may override some of the default values specified through the JavaBean

- **Mandatory attributes.** To require the deployer to provide a value for a configuration property. The resource adapter implementation may use the `@NotNull` constraint for this use case.

In the Java EE 6 envirAment, as sificap(ir)to ctioed4(EE 6 i)tir

dynamically reconfigurable through the `config-property-supports-dynamic-updates` attribute in the deployment descri

`ResourceAdapter` and

FIGURE 5-5 Lifecycle Management Model (Sequence Diagram)

The application component does a lookup of a connection factory in the Java Naming and Directory Interface™ (JNDI) name space. It uses the connection factory to get a connection to the underlying EIS. The connection factory instance delegates the connection creation request to the `ConnectionManager` instance.

The `ConnectionManager` enables the application server to provide different quality-of-services in9.48 398.04 553(v(n)-1.1.5(40(r)]29.(d.1(pli25.2(c)0ne)10.1587 -114.0118 Tc

FIGURE 6-2 Class Diagram: Connection Management Architecture

6.5.3.1 Interface

Note – The connector architecture allows one or more `ManagedConnection` instances to be multiplexed over a single physical pipe to an EIS. However, for simplicity, this specification describes a `ManagedConnection` instance as being mapped 1-1 to a phyer t6 pihed 1.5(t6)-6n.JET6488 T139.0107 Tc.0487 34[(T)-3he c3.39e

6.5.4.1 Interface

The connection management contract defines the following interface for a `ManagedConnection`

The `removeConnectionEventListener` method removes a registered `ConnectionEventListener` instance from a `ManagedConnection` instance. Since an application server may modify the list of event listeners at a time when the `ManagedConnection` may be iterating through its

6.5.4.3 Connection Matching Contract

The application server invokes the `ManagedConnectionFactory.matchManagedConnections` method (implemented

The method `getUserName` returns the user name known to the underlying EIS instance for an active connection. The name corresponds to the resource principal under whose security context the connection to the EIS instance has been established.

T49eer5 0 0 1A4(useds 98 7ci)5(puseds)]adc.06(apter TD.05 -1.2must5 -1.2p(pusedsovi

- Optionally, an exception indicating a connection related error. Refer to Section 22.2, “System Exceptions” on page 22-2 for details on the system exception. Note that the exception is used for the `CONNECTION_ERROR_OCCURRED` notification.

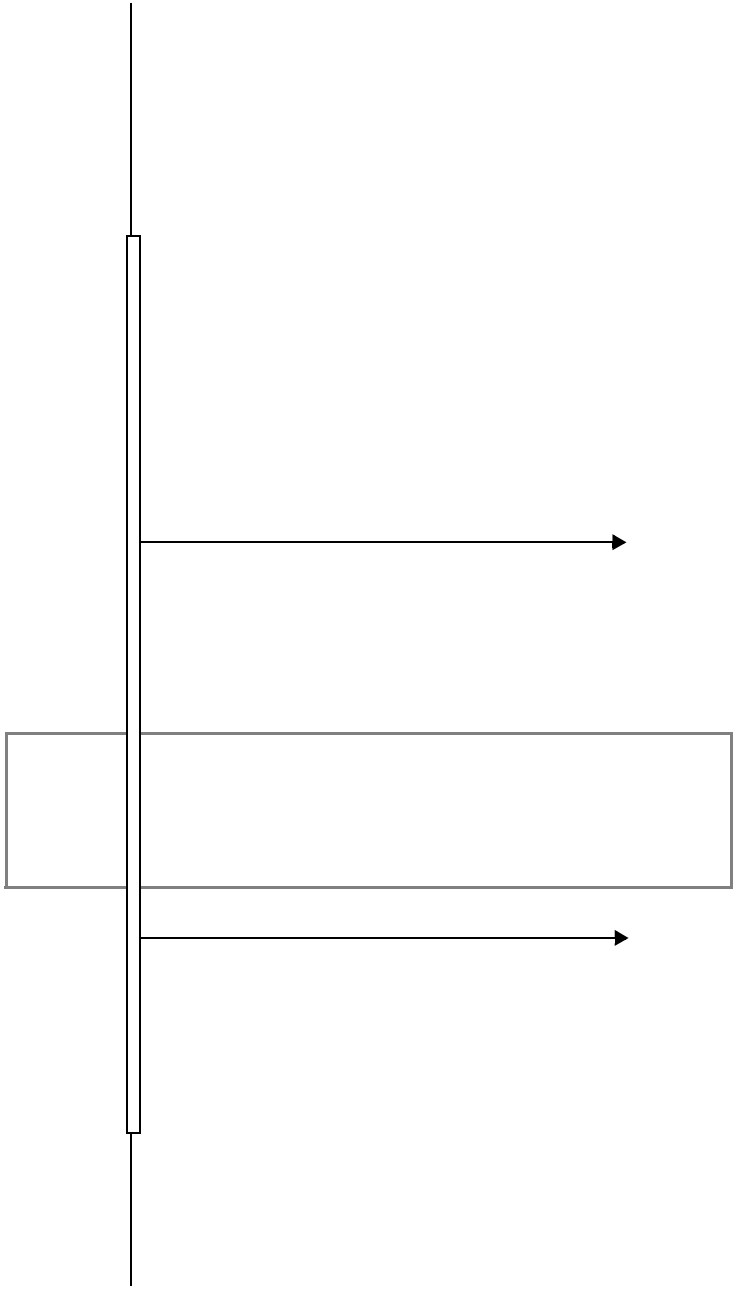
This class defines the following types of event notifications:

- `CONNECTION_CLOSED`
- `LOCAL_TRANSACTION_STARTED`
-

FIGURE 6-4

- The `ConnectionManager` instance (provided by the application server) handles the `allocateConnection` request by interacting with the application server specific connection pool manager. The interaction between a `ConnectionManager` instance and pool manager is internal and specific to an application server.
- The application server finds a candidate set of `ManagedConnection` instances from its connection pool. The candidate set includes all

FIGURE 6-6 OID: Connection Pool Management with Connection Matching



The following code extract shows how an

6.9.2 Scenario: Connection Creation in Non-Managed

- The `ConnectionManager`

CHAPTER 7T

7.2.2 Local Transaction Management

To avoid the overhead of using an XA transaction in a single resource manager scenario, the application server may optimize this scenario by using a local

The following figure illustrates this concept:

FIGURE 7-5 ManagedConnection Interface for Transaction Management

The transaction manager uses the `XAResource` interface to associate and dissociate a transaction with the underlying EIS resource manager instance and to perform a two-phase commit protocol. The transaction manager does not directly use the `ManagedConnection` interface. The next section describes the `XAResource` interface in more detail.

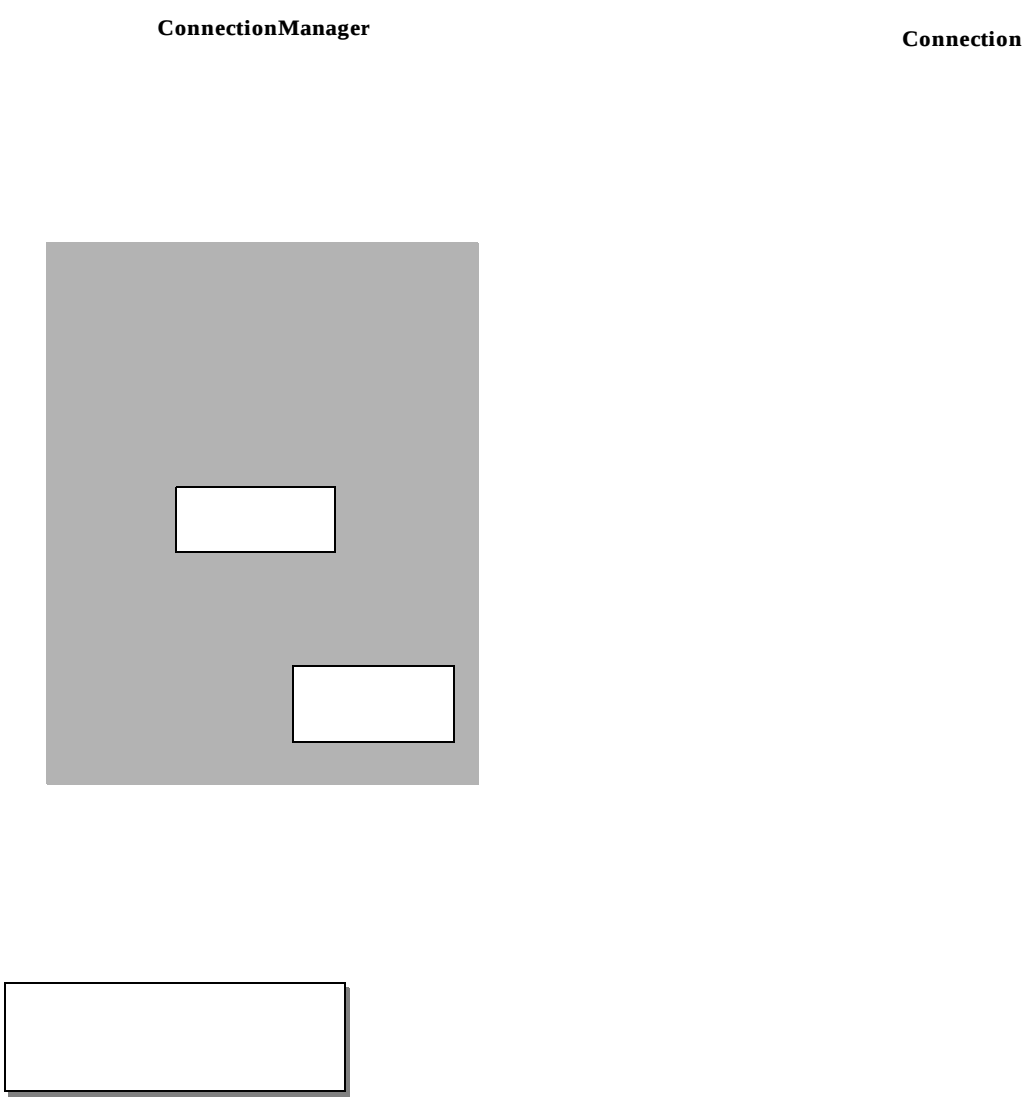
The application server uses the `LocalTransaction` interface to manage local transactions.

7.3.3 Interface: `LocalTransaction`

The following code extract shows the `javax.resource.spi.LocalTransaction` interface:

A resource adapter implements the `LocalTransaction` interface to provide support for local transactions that are performed on the underlying resource manager. An application server uses the `LocalTransaction` interface to manage local

FIGURE 7-6 Object Diagram: Transaction Management



7.6.1 Scenarios Supported

The following table specifies various transaction management scenarios and mentions whether these scenarios are within the scope of the connector architecture.

Description	Supported / NotSupported

7.6.2 Resource Adapter Requirements

The connector architecture does not require that all resource adapters must support JTA `XAResource` based transaction contract.

7.6.2.5 Unilateral Roll-back

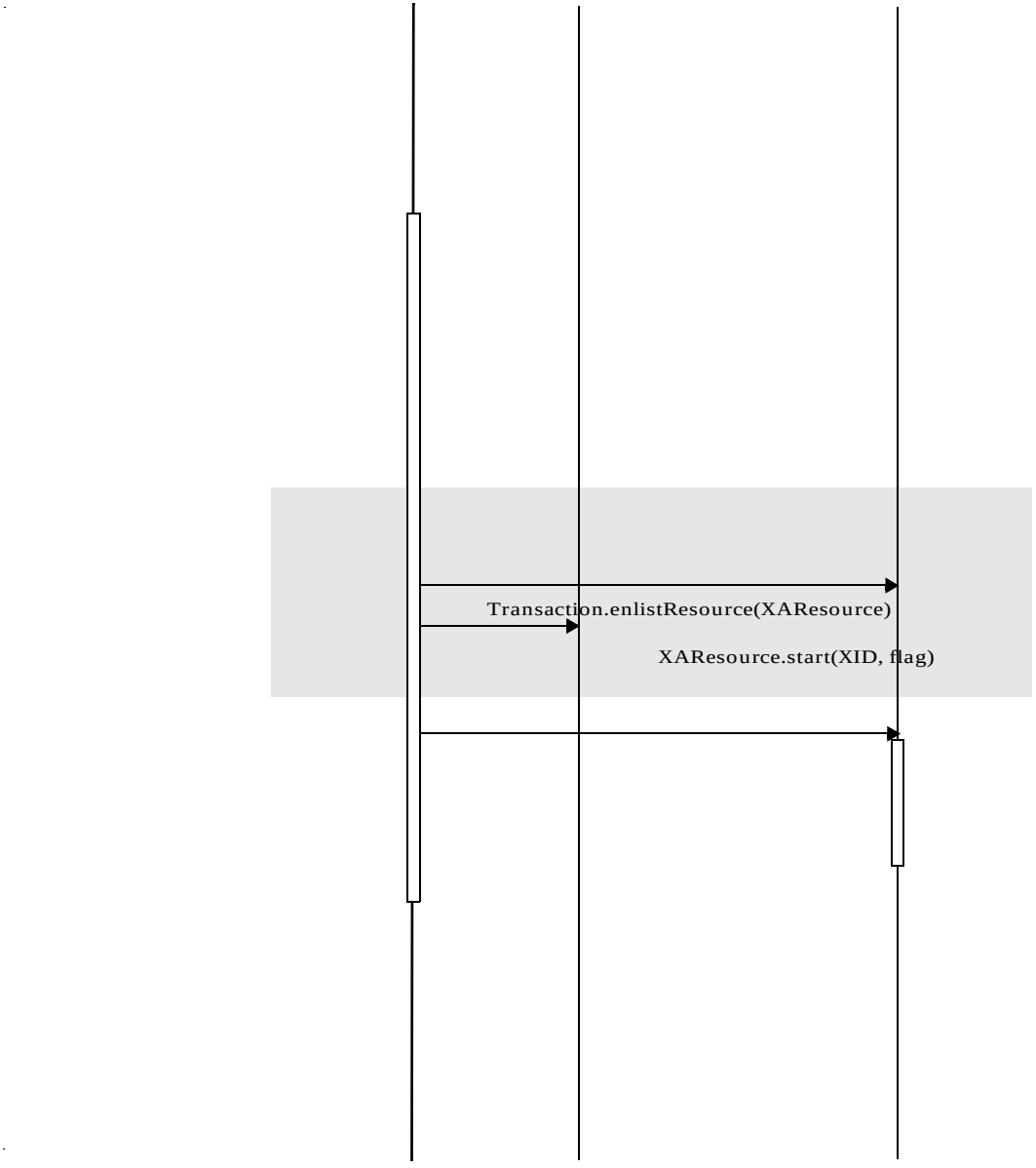
- The RM need not wait for global transaction completion to report an error. The RM can return a rollback-only flag as a result of any `XAResource.start` (fo()Tj/F5 1 Tf9.

7.6.4 Scenario: Transactional Setup for a ManagedConnection

The following object interactions are involved in the scenario shown in FIGURE 7-7.

1. The runtime scenario begins with a clie

FIGURE 7-7 OID: Transactional Setup For Newly Created ManagedConnection Instances



7.6.5 Scenario: Connection Close and JTA Transactional Cleanup

For each `ManagedConnection` instance in the pool, the application server registers a `ConnectionEventListener` instance to receive specific events on the connection. The connection event callback mechanism enables the application server to manage

7.6.6 OID: Transaction Completion

FIGURE 7-9

In these scenarios, an application component starts a local transaction using an application-level transaction demarcation interface, for example, `javax.resource.cci.LocalTransaction` as defined in the CCI, supported by the resource adapter. The resource adapter, in its implementation of the transaction demarcation interface, sends event notifications related to the local transaction, namely, local transaction begin, commit, and rollback. The application server is

7.8.3 Transaction Interleaving

The application server uses the connection event listener mechanism, specified through the interfaces

Later, A invokes B under the same transaction scope. B also attempts to acquire a

The connector architecture has an additional requirement that is applicable to resource adapters that support local transactions. Note that both `LocalTransaction`

7.10.3 Scenario: Local Transaction

This scenario illustrates the use of the connection sharing mechanism to implement requirement for a local transaction to span components.

In this scenario, two EJB components get connections to the same EIS resource manager within a single transaction. Both EJB components use the same local transaction capable resource adapter.

A local transaction is associated with a single physical connection. Both EJB components in this scenario share the same physical connection under the local

15. The container returns the physical connection to the pool for handling subsequent connection requests.

FIGURE 7-11 Connection Sharing Across Component Instances

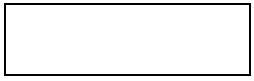
7.11 Connection Association

According to the connection management contract, a connection handle is created from a `ManagedConnection` instance using the `ManagedConnection.getConnection` method. A connection handle maintains an association with the underlying `ManagedConnection` instance.

7.11.1 Scenario

In the scenario shown in the following figure, session bean A acts as a client of entity bean C and makes invocations on methods of entity bean C. Another session bean B

FIGURE 7-14 Connection Acquisition Processing



- Keep the security architecture technology neutral and enable the specified security contract to be supported

8.5.1 Authentication Mechanism

define authorization policies. Programmatic and declarative security arorizo.8.0489 Tw[(and decl)u

8.6 Roles and Responsibilities

The output of the Deployer's work is a security policy descriptor specific to the operational environment. The format of the security policy descriptor is specific to an application server.

The Deployer performs the following depl

8.6.4 EIS Vendor

EIS provides a security infrastructure and environment that supports the security requirements of the client applications. An EIS can have its own security domain with a specific set of security

CHAPTER 9

Security Contract

This chapter specifies the security contract between the application server and the EIS. It also specifies the responsibilities of the Resource Adapter Provider and the Application Server Vendor for supporting the security contract.

9.1.2 Subject

The `PasswordCredential` class must implement the `equals` and `hashCode` h

FIGURE 9-1 Security Contract

Depending on whether the application server or application component is configured to be responsible for managing EIS sign-on (refer to Section 8.6.1, “Application Component Provider” on page 8-9), the resource adapter calls the

9.1.8.1 Contract for the Application Server

The application server may provide specific security services, such as principal mapping and delegation, and single sign-on, before using the security contract with the resource adapter. For example, the application server can map the caller

The application server has the following options for invoking the `createManagedConnection`

9.1.9 ManagedConnection

- The application server must use the method

There are several advantages in allowing an application server to manage threads instead of a resource adapter:

■

The application server may decide to reclaim active threads based on load conditions. It calls the

CODE EXAMPLE 10-1 javax.resource.spi.work

- `run` method: The `WorkManager` dispatches a thread that calls the `run` method to

10.3.3.2 `Work` Accepted

The submitted `Work` instance has been accepted for further processing. The accepted `Work` instance may either start executio

- The application server must use a value of -1 (`WorkManager.UNKNOWN`) to indicate an unknown `Work`

FIGURE 10-6 Work Submission - Callback Mechanism (Sequence Diagram)

10.3.6 Resource Adapter Thread Usage Recommendations

- Resource adapters are strongly recommended to use the work management contract to do work and interact with the application server only from within a `Work`

10.3.8 Illustration: Using a `Work` Instance to Listen on Multiple Network Endpoints

J2SE Version 1.4 provides the `java.nio` package that includes a multiplexed, non-blocking I/O facility. Using the `java.nio` package it is possible for a single thread, such as a `Work` instance, to listen on multiple network endpoints. javath6.4F2 1 Tf9.24 0 0 9.24 150.524.

the resource adapter during `Work` submission, the application server must send event notifications to the `WorkListener`. (see Section 10.3.4 “WorkListener Interface and WorkEvent Class”).

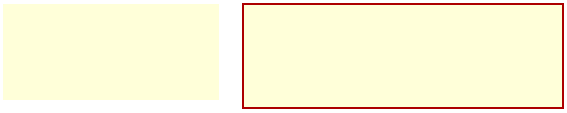
A `DistributableWork` instance may also use the mechanisms described in Generic

not implement the custom `WorkContext`. See Section 11.4.2 “Checking Support for a WorkContext Type” for a discussion about how resource adapters can check with the `WorkContext`s supported by the application server.

11.3.2 Requirements

■

FIGURE 11-3 WorkContext establishment during Work submission(Sequence Diagram)



11.4.1 Indicating Support for a `WorkContext` Type

A resource adapter provider can declare that it requires a list of `WorkContext` types to be supported by the application server through the `required-work-context`

The resource adapter may use an instance of the standard `HintsContext`

FIGURE 11-4 Generic Work Context Lifecycle listener callback (Sequence Diagram)

```
    }
}

public class MyWork implements Work, WorkContextProvider {

    void release(){ ..}

    List<WorkContext> getWorkContexts() {
           TransactionContext txIn
               = new TransactionContext(xid);
           List<WorkContext> icList = new ArrayList<WorkContext>();
           icList.add(txIn);
           // Add additional WorkContexts
           return icList;
    }

    void run(){
```

application servers which host business logic written as EJBs (session, entity and message-driven beans). Further, there are web service interactions that occur as part of the overall corporate workflow-3

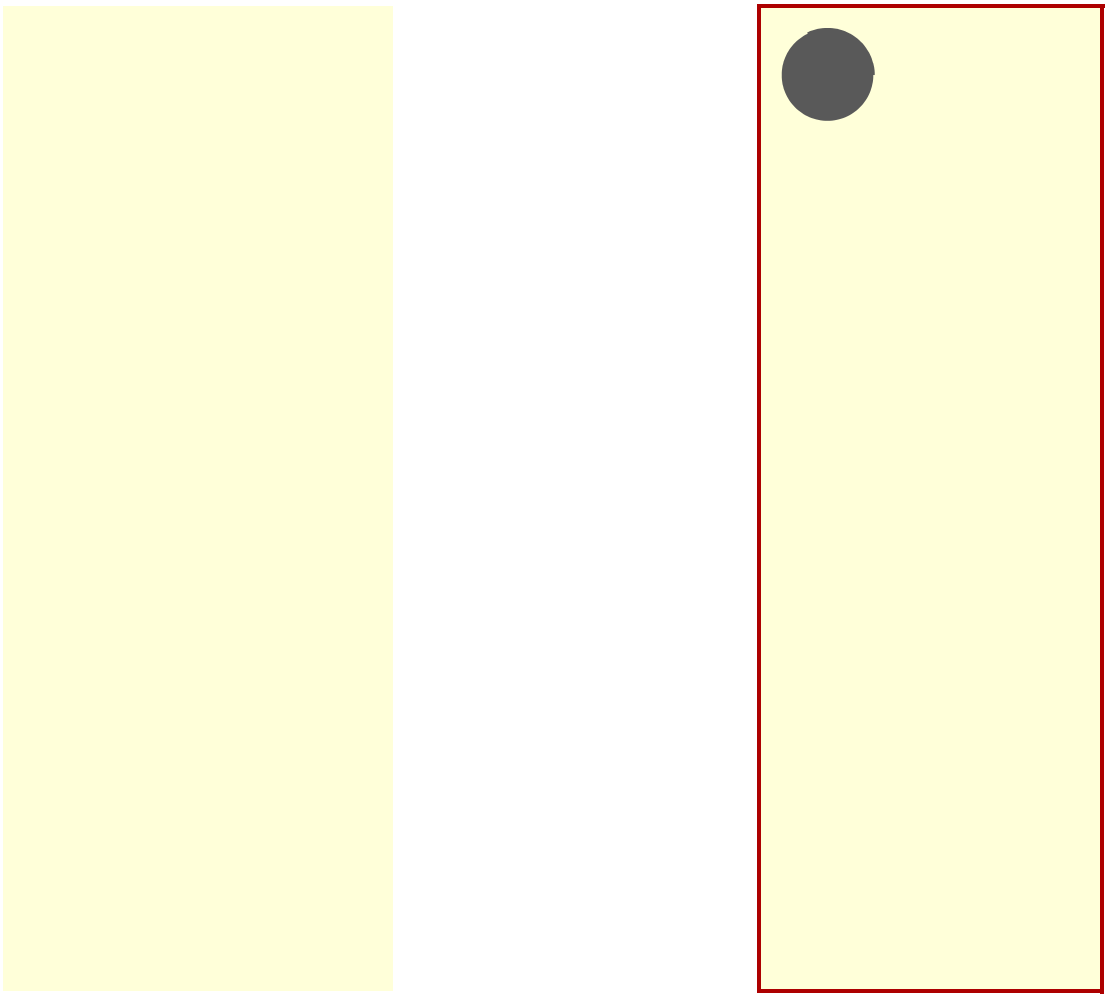
CHAPTER

- The message delivery always involves a message routing infrastructure, which

Thus, there is a need for a standard, generic contract between an application server and a message provider which allows a message provider to deliver messages to message endpoints (message-driven beans) residing in the application server independent of the specific messaging style, messaging semantics, and messaging infrastructure used to deliver messages. Such a contract also serves as the standard message provider pluggability contract which allows a wide range of message

13.3 Message Inflow Model

FIGURE 13-2 Message Inflow Contract (Object Diagram)



The `ResourceAdapter` interface supports methods used for endpoint activations and deactivations. The

The resource adapter is expected to detect the endpoint message listener type, either

13.4.2.1 List of Supported Message Listener Types

The resource adapter provides a list of endpoint message listener types it supports. Each type is represented as a name of the Java type of the message listener interface.

13.4.2.2 `ActivationSpec` JavaBean

The resource adapter provides the Java class name of an `ActivationSpec` JavaBean, one for each supported message

accessor method corresponding to the configuration property with the `@NotNull` constraint (or the corresponding XML validation descriptor equivalent), to indicate that the configuartion property must be specified during activation specifiow

An administered object instance, that implements `ResourceAdapterAssociation` interface must ensure that its Java class and the interface type implements `javax.resource.Referenceable` and `java.io.Serializable`

The deployer also configures a set of administered objects, if necessary. The resource

server chooses to prevent message delivery during endpoint activation, it may block the `createEndpoint` method call until the activation is completed or throw an `UnavailableException`.

13.4.6 Endpoint Deployment Steps

The sequence of steps involved in endpoint deployment involving the various actors

FIGURE 13-6 Endpoint Deployment (Sequence Diagram)

13.5.6 Transacted Delivery (Using Container-Managed Transaction)

Depending on the endpoint preferences, the application server brackets the message delivery to an endpoint instance with a Java Transaction API (JTA) transaction.

- This ensures that all the work done by the endpoint instance is enlisted as part of the transaction.
- This provides atomic message delivery / message consumption; that is, if the transaction were to be aborted by the application server due to an exceptional condition, all the work done by the endpoint instance is aborted, and the delivery is undone. If this does not occur, the transaction is committed, all the work done by the endpoinstTc.0484rsUsing 0d D, aiae(/ m)]TJ.2081(6(t)1.8 / m)]Tis undo.6(ition,t)2.3(i

application server fully controls the transaction boundaries and the resource adapter is merely a participant (the `XAResource` Resource Manager (RM)). See FIGURE 13-7.

- **Option B, `beforeDelivery`/**

The `beforeDelivery` and `afterDelivery`

enclosing transaction successfully commits, the messages are deemed to have been

FIGURE 13-7 Transacted Message Delivery: Option A(Sequence Diagram)

FIGURE 13-8 Transacted Message Delivery: Option B (Sequence Diagram)

```
<config-property>

      <config&noperty>
         <configopert<config-propert

               <config&nopert
                  <configopert<y>.2267 (o2.44y>)]TJ1.70 g0 05 Ti7nonfi Ta Twy1.2
```

```
<connectionfactory-interface>
   javax.jms.TopicConnectionFactory
</connectionfactory-interface>
<connectionfactory-impl-class>
```

```
<messageadapter>
   <messagelistener>
      <messagelistener-type>
         javax.jms.Messagagner><me/ssagaggisteneer>
```

class `com.kangaroo.MyEISAdapterImpl` since it supports the `com.kangaroo.MessageListener` type.

Then the deployer creates an instance of `com.kangaroo.MyEISActivationSpecImpl` and populates it with values. The `ActivationSpec`

- The contract may be leveraged to make the application components do work as part of a transaction initiated by a legacy EIS.

15.4.4 Requirements

- An application server must implement the transaction inflow contract. That is, it

-

CHAPTER

FIGURE 16-3 Security Context Establishment During Work Submission(Sequence Diagram)

3. Add instances of the necessary `Callback`s (Section 16.4.2 “Callbacks for Information from the Application Server” describes when a particular `Callback` is required to be employed by the resource adapter), usually a subset of the ones listed above, to an array and invokes the `handle()` method in the container’s `CallbackHandler` implementation by passing the array of `Callback` instances.

4. On sucessful return from the `CallbackHandler.handle()`

16.4.2 Callbacks for Information from the Application Server

As part of step 3 described in the section above, the following `Callback`s may be employed by a resource adapter. The descriptions of the `Callback`s below have been taken from the "Java Authentication Service Provider Interface for Containers Specification, version 1.4" on page E-2 specification. For detailed information, refer to the "Java Authentication Service Provider Interface for Containers Specification, version 1.4" on page E-2 and the Java API documentation of the `Callback`s defined in the `javax.security.auth.message.callback` package of that specification:

- A resource adapter may use the `CallerPrincipalCallback` to set the container's representation of the caller principal. The `CallbackHandler` must

FIGURE 16-5

16.4.4 Case 2: Identity Translated Between Security Domains

16.4.5 Establising a `Principal` as the Caller Identity

Prior to returning to the container, `setupSecurityContext` must use the container provided `CallbackHandler`

The `WorkManager` must detect that the handler

corresponding to the established security identity, and `MessageDrivenContext.isCallerInRole()`

```
 PasswordValidationCallback pwdCallback =
new PasswordValidationCallback(
    executionSubject, username, pwd);
```

-

An application development tool can also compose or generate an application component that uses the generated Java classes for EIS access.

- Data representation-related interfaces that are used to represent data structures involved in an interaction with an EIS instance:
  - `javax.resource.cci.Record`
  - `javax.resource.cci.MappedRecord`
  -

17.6.2.4 Implementation

The `InteractionSpec` interface must be implemented as a JavaBean to support tools. The properties on the

17.6.3 LocalTransaction

The javax.resource.cci.LocalTransaction defines a transaction demarcation

the `Record` and `RecordFactory`

17.10.1.2

MappedRecord

The methods `createMappedRecord` and

The following code extract shows the `Streamable`

The following code extract shows the

- The use of the JDBC `ResultSet` interface enables a tool or EAI vendor to

A result set that uses read-only concurrency does not allow updates of its content, while an updatable result set allows update

A component uses the `rowUpdated`, `rowInserted`, and `rowDeleted` methods of the `ResultSet` interface to determine whether a row has been affected by a visible

CHAPTER **18**

Metadata Annotations

This chapter defines a simplified API for development of resource adapters. The goal tt18

18.3.1 `metadata-complete` Deployment Descriptor Element

A new attribute, `metadata-complete`, is introduced in the Connector 1.6 deployment descriptor (`ra.xml`). The `metadata-complete` attribute defines whether the deployment descriptor for the resource adapter module is complete, or whether the class files available to the module and packaged with the resource adapter should be examined for annotati

All the metadata annotations described in this chapter are in the `javax.resource.spi`

18.5.1 Discovery of Configuration Properties

Configuration tools provided by the container must introspect the JavaBeans listed

18.7.1 Example

An `ActivationSpec`

When a resource adapter RAR packaged within a Java EE application EAR needs to be referenced, the resource adapter name may be prefixed with a “#” character to portably refer to the embedded resource adapter within the EAR. As an example, a Servlet bundled in an enterprise archive EAR, may access the embedded resource adapter foo.rar in the EAR, by using the name “*#foo*”.

These resource definition annotations must only be defined in modules that have access to the resource adapter as per the rules defined in Section 20.2.0.4, “Requirements” on page 20-5.

These resource definition annotations must be supported in all products that support the deployment process as defined by the Java EE Platform Specific1(tio)Tj/F]TJP8 anF]TJPd

```
@Retention(RetentionPolicy.RUNTIME)
public @interface ConnectionFactoryDefinition {
    String name();
    String description() default "";

    String resourceAdapter();
    String interfaceName();

    TransactionSupport.TransactionSupportLevel transactionSupport()
        default
        TransactionSupport.TransactionSupportLevel.NoTransaction;
    int maxPoolSize() default -1;
    int minPoolSize() default -1;
    String[] properties() default {};
```

The connection factory may be configured by setting the annotation elements for the commonly used connection factory properties as indicated above. Additional properties required by the `ManagedConnectionFactory`, that is associated with the connection factory being defined, are specified through the `properties` element. Properties, if specified, that do not match configuration property names of the `ManagedConnectionFactory` or cannot be mapped to vendor-specific properties may be ignored.

18.9.1.1 Example

A XA-capable connection factory resource may be defined in a Servlet as follows:

CODE EXAMPLE 18-13 ConnectionFactoryDefinition Annotation Definition Example

```
@ConnectionFactoryDefinition(name="java:comp/eis/MyEISCF",
     interfaceName="com.eis.ConnectionFactory",
     resourceAdapter="MyEISRA",
     transactionSupport=
        TransactionSupport.TransactionSupportLevel.XATransaction)
```

Once defined, a conn116 Tc.
c.0754 Tw[(pr)1736e)5.1sa ourdhaa tevisaple-sa the

Once defined, the connector connection factory resources may be referenced by a component, that has the standalone

Once defined, the Queue resource may be referenced by a component, that has the embedded `MyJMSRA` resource adapter visible to it as per the rules defined in Section 20.2.0.4, “Requirements” on page 20-5, using the `resource-ref` deployment descriptor element or the `Resource`

18.9.4.1 Example

Multiple administered object definiti

CHAPTER 19

API Requirements

This chapter specifies the API requirements for the resource adapter and application server implementations.

19.1 Requirements of the Application Server

- The application server must support the de

19.4 Equality Constraints

This section specifies the equality constraints on object implementations of the various types defined by this specification.

19.4.1 Equality based on Java Object Identity

The candidate objects are implementations of

CHAPTER 20

Packaging Requirements

This chapter specifies requirements for packaging and deploying a resource adapter.

FIGURE 20-1 Packaging and Deployment Lifecycle of a Resource adapter

A resource adapter module corresponds to a Java EE module in terms of the Java EE composition hierarchy. Refer to the Java EE Platform specification (see “Java

The following figure shows the composition model of a resource adapter module with other Java EE modules.

FIGURE 20-2 Deployment of a Resource Adapter Module

The stand-alone deployment of a resource adapter module into an application server is typically done to support scenarios in which multiple Java EE

20.2.0.2 RAR Contents

- Required `WorkContext` classes: A resource adapter may optionally provide a

`LocalTransaction`: The resource adapter supports resource manager local transactions by implementing the `LocalTransaction` interface. The local transaction management contract is specified in Section 7.7, “Local Transaction Management Contract” on page 7-28.g the C

J

- ActivationSpec class: The resource adapter provider must specify the Java class name of the activation specification class. The implementation of this class must be a JavaBean. An `ActivationSpec` specifies an activation specification per message listener type. The `ActivationSpec` is configured by a message endpoint deployer during application deployment.
- Required ActivationSpec properties: The resource adapter provider may optionally specify a set of required properties for an

20.5.1.1 Requirements

The `ResourceAdapter` implementation must be a JavaBean.

20.5.2 `ManagedConnectionFactory`

The class that implements the

20.6 JNDI Configuration and Lookup

This section specifies requirements for the configuration of the JNDI environment for a resource adapter.

In both managed and non-managed application scenarios, an application component or application client must look up a conn[(T)-4. Sactorylon-1.3(istn)5.ce aon-1.3(ith)5(e)ompo

This section specifies the responsibilities of the roles involved in the JNDI configuration of a resource adapter.

20.6.1.1 Deployer

The deployer is responsible for configuring connection factory instances in the JNDI environment. The deployer should manage the JNDI namespace such that the same

20.6.2 Scenario:

The mapping from a `Reference` instance to multiple configured property sets

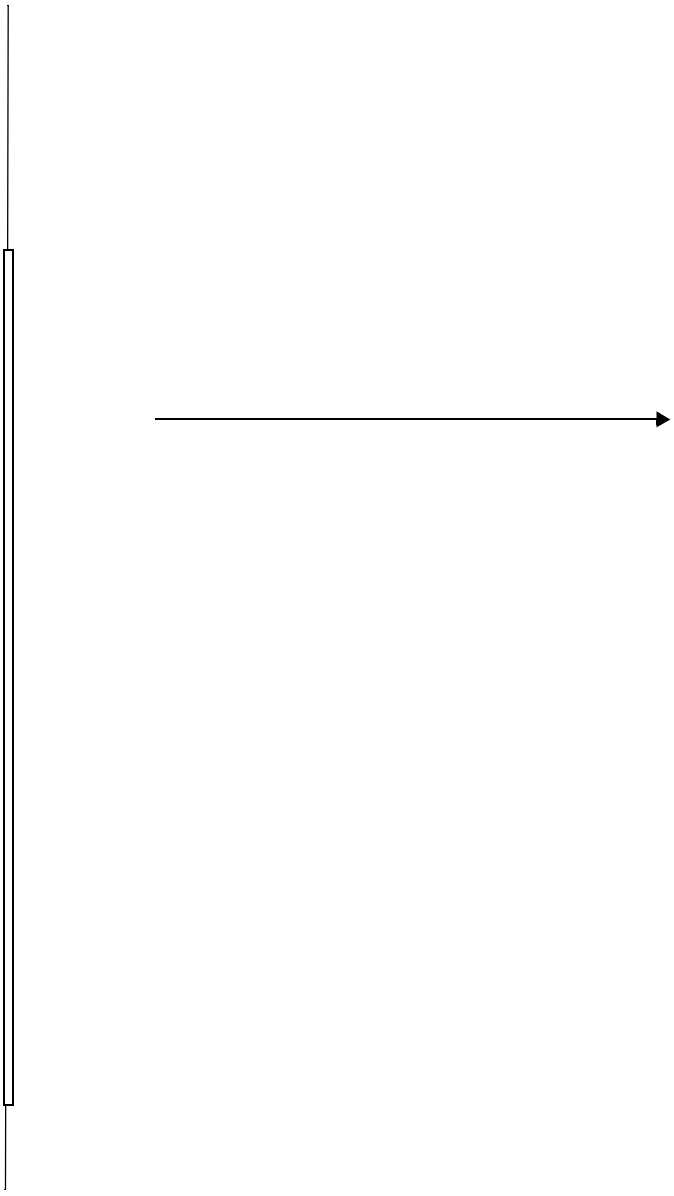
20.6.3.2 Deployment

The following deployment code

Configuration" on page 5-14 while configuring a `ManagedConnectionFactory` JavaBean instance. The application server may use an existing instance of the `ManagedConnectionFactory` implementation class, if available.

3. The application server calls setter methods on the `ManagedConnectionFactory`

FIGURE 20-3 OID:Lookup of a ConnectionFactory Instance from JNDI



"META-INF/ra.xml" in the connector’s rar file. All Connector deployment descriptors must indicate the connector resource adapter schema by using the Java EE namespace:

http://xmlns.jcp.org/xml/ns/javaee

and by indicating the version of the schema by
using the version element as shown belowmespace:

```
</xsd:sequence>
```

```
          Any additional security mechanisms are outside the
          scope of the Connector architecture specification.

          ]]>
        </xsd:documentation>
      </xsd:annotation>

        </xsd:element>
        <xsd:element name="credential-interface"
         type="javaee:credential-interfaceType"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:ID"/>
    </xsd:complexType>

<!-- ********************************************************** -->

    <xsd:complexType name="config-property-nameType">
      <xsd:annotation>
        <xsd:documentation>
      <![CDATA[

        The config-property-nameType contains the name of a
        configuration property.
```

```
  ]]>
  </xsd:documentation>
</xsd:annotation>
```

```
          ]]>
        </xsd:documentation>
      </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:ID"/>
    </xsd:complexType>

<!-- ********************************************** -->

    <xsd:complexType name="connectorType">
      <xsd:annotation>
```

```
          “false”, the deployment tool must examine the class files of the
          application for annotations, as specified by this specification. If
          the deployment descriptor is not included or is included but not
          marked metadata-complete, the deployment tool will process
          annotations

          Application servers must assume that metadata-complete is true for
          resource adapter modules with deployment descriptor version
          lower than 1.6.
        </xsd:documentation>
    </xsd:annotation>
    </xsd:attribute>

    <xsd:attribute name="id" type="xsd:ID"/>
  </xsd:complexType>

<!-- ************************************************** -->
```

```
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>
```

```
    resource adapter module. This type specifies whether a
    license is required to deploy and use this resource adapter,
    and an optional description of the licensing terms
    (examples: duration of license, number of connection
    restrictions). It is used by the license element.

      </xsd:documentation>
    </xsd:annotation>

    <xsd:sequence>
      <xsd:element name="description"
       type="javaee:descriptionType"
       minOccurs="0"
       maxOccurs="unbounded"/>
      <xsd:element name="license-required"
       type="javaee:true-falseType">
    <xsd:annotation>
      <xsd:documentation>

        The element license-required specifies whether a
        license is required to deploy and use the
        resource adapter. This element must be one of
        the following, "true" or "false".

      </xsd:documentation>
    </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"/>
  </xsd:complexType>

<!-- ********************************************** -->

  <xsd:complexType name="messageadapterType">
    <xsd:annotation>
      <xsd:documentation>

    The messageadapterType specifies information about the
    messaging capabilities of the resource adapter. This
    contains information specific to the implementation of the
    resource adapter library as specified through the
    messagelistener element.

      </xsd:documentation>
    </xsd:annotation>

    <xsd:sequence>
```

```
  </xsd:complexType>

<!-- ********************************************** -->

  <xsd:complexType name="outbound-resourceadapterType">
    <xsd:annotation>
      <xsd:documentation>

    The outbound-resourceadapterType specifies information about
    an outbound resource adapter. The information includes fully
```

```
        <xsd:field    xpath="javaee:adminobject-class"/>
      </xsd:unique>

        <xsd:element name="security-permission"
         type="javaee:security-permissionType"
         minOccurs="0"
</sd:IompleType name="security-permission"Ty6.7(pae")]TJ0 -1.226ionTypdx:ID"/>
         maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:ID"/>
    </xsd:complexType>

  <!-- ************************************************. --
```

CHAPTER 21

Runtime Environment

This chapter focuses on the Java portion of a resource adapter that executes within a Java compatible runtime environment. A Java runtime environment is provided by an application server and its containers.

The chapter specifies the Java APIs that a Java EE

documentation. Refer to `http://docs.oracle.com/javase/6/docs/technotes/guides/security/permissions.html`.

TABLE 21-1 Default Security Permission Set

Security Permission	Default Policy	Notes
`java.security.AllPermission`	deny	Extreme care should be taken before granting this permission to a resource adapter. This

21.3 Requirements

21.4 Privileged Code

A resource adapter runs in its own protection domain as identified by its code

During resource adapter deployment, the application server processes this `security-permission-spec` and grants the necessary permissions to the `wombat.rar` code base. This is an implementation-specific mechanism and not prescribed by the specification. As an example, the application server may append these permissions to the `java.policy` file or some implementation-specific policy file, and this may involve manual intervention.

The following JavaBeans of a resource adapter archive have their lifecycle managed by the application server (see Chapter 5, “Lifecycle Management"):

- `ResourceAdapter`
-

CHAPTER 22

Exceptions

This chapter specifies standard exceptions that identify error conditions which may occur as part of the connector architecture.

-

23.1.1.1 Standalone Resource Adapter Visibility to an Application

The Java EE Connector Architecture 1.6 specification defines the classloading requirements for an application serv

APPENDIX **A**

Previous Version Deployment Descriptors

This appendix contains Document Type

```
<!DOCTYPE connector PUBLIC"-//Sun Microsystems, Inc.//DTD
Connector 1.0//EN" "http://java.sun.com/dtd/connector_1_0.dtd">

-->

<!--
Ths(/d.04 cgAwing d.04 /coventid.04 ons appld.04 y to all d.04 J2 EEdepd.04 coy
elements unleso indficted othet

Ielementsohcte


 I ed.04 cements d.04 whose vct "enumerted typd.04 e", the d.04 value iso


 I ed.04 cements d.04 (hct sped.04 cify a pt)6.7a(hname ht)6.7o a filethe same
relatived.04efd.04ilenamesd.04e(d.04i.e.,thoso
relatived.04etothoofthetfile’sd.06(namespaec)6.7e.>
d.04 i.e., thto specid.04 fy>
```

```
optional description specifies any resource adapter specific
requirement
for the support of security contract and authentication mechanism.

Note that BasicPassword mechanism type should support the
javax.resource.spi.security.PasswordCredential interface. The
Kerbv5
mechanism type should support the
javax.resource.spi.security.Generic-
Credential interface.

Used in: resourceadapter
-->
<!ELEMENT authentication-mechanism (
description?, authentication-mechanism-type, credential-
interface)>

<!--
The element authentication-mechanism-type specifies type of an
authentication
mechanism.

The example values are:
<authentication-mechanism-type>BasicPassword
                  </authentication-mechanism-type>
   <authentication-mechanism-type>Kerbv5
                  </authentication-mechanism-type>

Any additional security mechanisms are outside the scope of the
Connector architecture specification.

Used in: authentication-mechanism
-->
<!ELEMENT authentication-mechanism-type (#PCDATA)>

<!--
The element config-property contains a declaration of a single
configuration property for a ManagedConnectionFactory instance.

Each ManagedConnectionFactory instance creates connections to a
specific EIS instance based on the properties configured on the
```

```
The element connectionfactory-impl-class specifies the fully-
qualified
name of the ConnectionFactory class that implements resource
adapter
specific ConnectionFacionr(fae.r)]TJ0 2.44267 TD[Used io:s reouece(adapm)6.7(rr
```

```
Used in: license
-->
<!ELEMENT license-required (#PCDATA)>

<!--
The element managedconnectionfactory-class specifies the
name of the Java class that implements the
javax.resource.spi.Managed-
```

```
<!ELEMENT security-permission-spec (#PCDATA)>

<!--
The small-icon element contains the name of a file containing a
small
(16 x 16) icon image. The file name is a relative path within the
component's jar file.
```

```
The element vendor-name specifies the name of resource adapter
```

A.2 J2EE Connector Architecture 1.5 Resource Adapter XML XSD

This section specifies the XML Schema defi

```
            </activationspec-class>

            ]]>
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="required-config-property"
      type="j2ee:required-config-propertyType"
      minOccurs="0"
      maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="id" type="xsd:ID"/>
</xsd:complexType>

<!-- ********************************************************** -->

<xsd:complexType name="adminobjectType">
   <xsd:annotation>
      <xsd:documentation>

      <![CDATA[
```

```
            ]]>
          </xsd:documentation>
        </xsd:annotation>

        </xsd:element>
        <xsd:element name="adminobject-class"
        type="j2ee:fully-qualified-classType">
          <xsd:annotation>
            <xsd:documentation>
              <![CDATA[

              The element adminobject-class specifies the fully
              qualified Java class name of an administered object.

              Example:
              <adminobject-class>com.wombat.DestinationImpl
              </adminobject-class>

              ]]>
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>

        <xsd:element name="config-property"
        type="j2ee:config-propertyType"
        minOccurs="0"
        maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:ID"/>
    </xsd:complexType>

    <!-- ********************************************** -->

    <xsd:complexType name="authentication-mechanismType">
        <xsd:annotation>
          <xsd:documentation>

            The authentication-mechanismType specifies an
            authentication
            mechanism supported by the resource adapter. Note that
            this
            support is for the resource adapter and not for the
```

```
  Note that BasicPassword mechanism type should support
 the
  javax.resource.spi.security.PasswordCredential
 interface.
  The Kerbv5 mechanism type should support the
  org.ietf.jgss.GSSCredential interface or the deprecated
  javax.resource.spi.security.GenericCredential
 interface.

  </xsd:documentation>
</xsd:annotation>

<xsd:sequence>
<xsd:element name="description"
type="j2ee:descriptionType"
minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="authentication-mechanism-type"
type="j2ee:xsdStringType">
   <xsd:annotation>
     <xsd:documentation>
     <![CDATA[

     The element authentication-mechanism-type specifies
     type of an authentication mechanism.

     The example values are:

     <authentication-mechanism-type>BasicPassword
```

The config-propertyType contains a declaration of a

```
      ]]>
      </xsd:documentation>
  </xsd:annotation>

</xsd:element>

<xsd:element name="connection-interface"
type="j2ee:fully-qualified-classType">
```

```
            <xsd:annotation>
                <xsd:documentation>

                The element resourceadapter-version specifies a string-
                based version
                of the resource adapter from the resource adapter
                provider.

                </xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="license"adapter
g
```

application server to find out the Credential interface it should use as part of the security contract.

The possible values are:

CODE EXAMPLE A-2 Connector Architecture 1.5 Resource Adapter XSD

```
                unique in the messageadapter. Several
                messagelisteners
                can not use the same messagelistener-type.

                </xsd:documentation>
              </xsd:annotation>
              <xsd:selector xpath="j2ee:messagelistener"/>
              <xsd:field    xpath="j2ee:messagelistener-type"/>
           </xsd:unique>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"/>
 </xsd:complexType>

<!-- ********************************************** -->


 <xsd:complexType name="licenseType">
    <xsd:annotation>
       <xsd:documentation>

       The licenseType specifies licensing requirements for the
       resource adapter module. This type specifies whether a
       license is required to deploy and use this resource adapter,
       and an optional description of the licensing terms
       (examples: duration of license, number of connection
       restrictions). It is used by the license element.

       </xsd:documentation>
    </xsd:annotation>

    <xsd:sequence>
       <xsd:element name="description"
       type="j2ee:descriptionType"
       minOccurs="0"
       maxOccurs="unbounded"/>
       <xsd:element name="license-required"
       type="j2ee:true-falseType">
           <xsd:annotation>
              <xsd:documentation>

              The element license-required specifies whether a
              license is required to deploy and use the
              resource adapter. This element must be one of
              the following, "true" or "false".

              </xsd:documentation>
```

```
    </xsd:element>
    <xsd:element name="credential-interface"
     type="javaee:credential-interfaceType"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID"/>
</xsd:complexType>
```

```
    </xsd:simpleContent>
  </xsd:complexType>

<!-- ************************************************** -->

  <xsd:complexType name="config-property-typeType">
    <xsd:annotation>
      <xsd:documentation>
    <![CDATA[
```

```
<xsd:annotation>

    <![CDATA[

      The connection-impl-classType specifies the fully
      qualified name of the Connection class that
      implements resource adapter specific Connection
      interface.  It is used by the connection-impl-class
      elements.

      Example:

  <connection-impl-class>com.wombat.ConnectionImpl
  </connection-impl-class>

      ]]>

</xsd:annotation>
  </xsd:element>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:ID"/>
```

```
      Specification, version 6.
     </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:group ref="javaee:descriptionGroup"/>
  <xsd:element name="vendor-name"
   type="javaee:xsdStringType" minOccurs=”0”>
<xsd:annotation>
  <xsd:documentation>

    The element vendor-name specifies the name of
    resource adapter provider vendor.

   If there is no vendor-name specified, the application server
   server must consider the default “” (empty string) as
   the name of the resource adapter provider vendor.

  </xsd:documentation>
</xsd:annotation>
  </xsd:element>
  <xsd:element name="eis-type"
   type="javaee:xsdStringType" minOccurs=”0”>
<xsd:annotation>
  <xsd:documentation>

    The element eis-type contains information about the
    type of the EIS. For example, the type of an EIS can
    be product name of EIS independent of any version
    info.

    This helps in identifying EIS instances that can be
    used with this resource adapter.

 If there is no eis-type specified, the application
 server must consider the default "" (empty string) as
 the type of the EIS.

  </xsd:documentation>
</xsd:annotation>
  </xsd:element>
```

```
     of the resource adapter from the resource adapter
     provider.

     If there is no resourceadapter-version specified, the
    application
     server must consider the default "" (empty string) as
     the version of the resource adapter.

   </xsd:documentation>
</xsd:annotation>

   </xsd:element>
   <xsd:element name="license"
    type="javaee:licenseType"
    minOccurs="0"/>
   <xsd:element name="resourceadapter"
    type="javaee:resourceadapterType"/>

   <xsd:element name="required-work-context"
    type="javaee:fully-qualified-classType”
    minOccurs="0”
    maxOccurs="unbounded”>
  <xsd:annotation>
   <xsd:documentation>

     The element required-work-context specifies the
      class name that implements WorkContexte tiex tace,TJ0 -1.2267 TD[(0068

       </xsd:documentation>
```

deployment tool. This attribute does not have a default, and the resource adapter developer/deployer is required to specify it. This attribute allows the deployment tool to choose which schema to validate the descriptor against.

```
The inbound-resourceadapterType specifies information
about an inbound resource adapter. This contains information
specific to the implementation of the resource adapter
library as specified through the messageadapter element.

  </xsd:documentation>
</xsd:annotation>

<xsd:sequence>
  <xsd:element name="messageadapter"
   type="javaee:messageadapterType"
   minOccurs="0">
<xsd:unique name="messagelistener-type-uniqueness">
  <xsd:annotation>
    <xsd:documentation>

      The messagelistener-type element content must be
      unique in the messageadapter. Several messagelisteners
      can not use the same messagelistener-type.

    </xsd:documentation>
  </xsd:annotation>
  <xsd:selector xpath="javaee:messagelistener"/>
  <xsd:field    xpath="javaee:messagelistener-type"/>
</xsd:unique>
  </xsd:element>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:ID"/>
```

```
    <xsd:element name="description"
     type="javaee:descriptionType"
     minOccurs="0"
     maxOccurs="unbounded"/>
    <xsd:element name="license-required"
     type="javaee:true-falseType">
  <xsd:annotation>
    <xsd:documentation>
```

```
          can not use the same adminobject-interface and
adminobject-class.

        </xsd:documentation>
      </xsd:annotation>
      <xsd:selector xpath="javaee:adminobject"/>
      <xsd:field    xpath="javaee:adminobject-interface"/>
      <xsd:field    xpath="javaee:adminobject-class"/>
    </xsd:unique>

      <xsd:element name="security-permission"
       type="javaee:security-permissionType"
       minOccurs="0"
       maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"/>
  </xsd:complexType>

<!-- ********************************************** -->

  <xsd:complexType name="security-permissionType">
    <xsd:annotation>
      <xsd:documentation>

    The security-permissionType specifies a security
    permission that is required by the resource adapter code.

    The security permission listed in the deployment descriptor
    are ones [(   e re ones [(   e re onple)6.are67 -2fe)6.are6(r)from  e re oo
```

APPENDIX **B**

Caching Manager

This chapter describes how the connector architecture supports caching.

This section serves as a brief introduction to the caching support in the connector architecture. A future version of the connector architecture will address this issue in detail.

B.1 Overview

The connector architecture provides a standard way of extending an application manager for plugging in caching managers. A caching manager may be provided by a third-party vendor or a resource adapter provider.

The architecture of this application is illustrated in the following diagram.

FIGURE C-1 Illustrative Architecture of an eStore Application

C.1.1 Scenario

A customer, using a web browser, initiates an e-commerce transaction with the

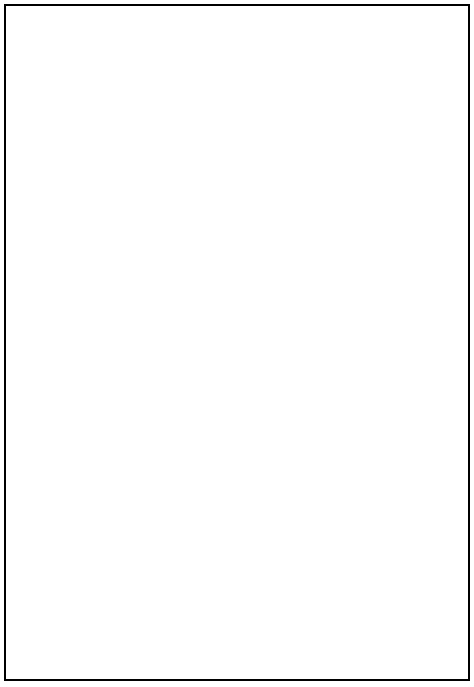
The ERP system administrator configures all legal employees as valid user accounts in the ERP system. He also must set up various roles (Manager, HRManager, and Employee), default passwords, and access privileges. This security information is kept synchronized with the enterprise-wide directory service, which is used by

application provides integrated purchasing and inventory management functions) and the financial accounting applications (the applications supported by the legacy system from vendor Y).

Company B is a huge decentralized enterprise; its business units and departments are geographically distributed. In this scenario, different IS departments manage

APPENDIX

FIGURE D-1 Security Architecture.



D.3.1 JAAS Modules

The connector architecture recommends (but does not mandate) that an application server support *platform-wide* JAAS modules (also called authentication modules) for authentication mechanisms that are common across multiple EISs. The implementation of these JAAS modules is typically specific to an application server. However, these modules may be developed to

The format for the above configuration is specific to an application server implementation.

D.5 Scenarios

3. The application server calls `LoginContext.login` method. On a successful return from the principal mapping module, the application server gets a `Subject` instance that has the mapped resource principal with a valid `PasswordCredential`.

FIGURE D-2 Resource Adapter-Managed Authentication

4. The application server invokes the method `ManagedConnectionFactory53.2 Tm.0039`

D.5.2 Scenario: Kerberos and Principal Delegation

The scenario in the following figure involves the following steps:

6. If the resource adapter and EIS support GSS-API

D.5.3 Scenario: GSS-API

If an EIS supports the GSS mechanism, a resource adapter may (but is not required to) use GSS-API to set up a secure association with the EIS instance. The section “Generic Security Service API: GSS-API” on page D-6 gives a brief overview of GSS-API.

FIGURE D-4 GSS-API use by Resource Adapter

A formal specification of .7(atio0)-.4(R6.42 Tm.95(e5468 108 276.42 Tm.0080.0544 Tw[4.7(t) Tcpter)]T

D.5.4 Scenario: Kerberos Authentication After Principal Mapping

The scenario depicted in the following figure involves the following steps:

FIGURE D-5 Kerberos Authentication After Principal Mapping

1. The application server is configured to use the principal mapping module and Kerberos module. The two authentication modules are stacked together with the principal mapping module first.
2. The application server creates a `LoginContext` instance by passing in the `Subject` instance for the caller principal and a `CallbackHandler` instance. Next, the application server calls the `login`

APPENDIX E

F.2 Version 1.0 - Public Draft 1

- Removed definition of "Connector" from 2.1. The term Connector is now used broadly refer to the Connector architecture, while resource adapter refers to the system library.
- Added requirement for ConnectionEventListener to 6.9.2: Application Server
- Added connection handle property to

APPENDIX G

APPENDIX

APPENDIX I