Oracle® Tuxedo JCA Adapter

Inflow Transaction Users Guide

12*c* Release 1 (12.1.1)

June 2012

ORACLE

Oracle Tuxedo JCA Adapter Users Guide, 12*c* Release 1 (12.1.1)

Oracle Tuxedo JCA Adapter Inflow Transaction Guide

Oracle Tuxedo JCA Adapter Inflow Transaction Guide

This document describes the configuration, deployment, and touches some programming aspects of the inflow transaction through connector-based MDB (also called *JMS-based MDB*) feature. It allows inflow (inbound) transactions from Oracle Tuxedo to a Java application server. It contains the following topics:

- Packaging and Contents
- Inflow Transaction Overview
- Inflow Transaction Configuration
- Inflow Transaction Adapter Deployment
- Oracle Tuxedo Application Domain
- Oracle Tuxedo Transactional Client Source Code

For more information see, the Tuxedo JCA Adapter Programming Guide and the Tuxedo JCA Adapter Users Guide.

Packaging and Contents

The Inflow Transaction feature is delivered as an RAR file; you must un-jar the file and modify the Oracle Tuxedo JCA Adapter (Tuxedo JCA Adapter) configuration before it can be installed on an application server.

Supported Application Servers and Oracle Tuxedo Versions

Table 1 lists supported Oracle Tuxedo and application server versions.

Table 1 Supported Versions

Name	Version
Application Server	WebLogic Server versions 11gR1 & 12c, and WebSphere 7.0 & 8.0
Oracle Tuxedo	11gR1PS1 and later

RAR File Name

The RAR file name is `com.oracle.tuxedo.TuxedoAdapter.rar`. After you modify the Tuxedo JCA Adapter configuration, it can be archived (using any name) be used to configure Tuxedo JCA Adapter to the application server.

RAR File Contents

Table 2 lists the RAR file contents.

Table 2 RAR File Content

File Name	Description
`adapter.properties`	*Tuxedo JCA Adapter* message catalogue .
`adapter_ja.properties`	Japanese version of message catalogue of *Tuxedo JCA Adapter*.
`com.bea.core.i18n_[version].jar`	I18N library
`com.bea.core.jatmi_[version].jar`	JATMI library
`com.oracle.tuxedo.adapter_[version].jar`	Tuxedo JCA Adapter
`dmconfig.xml`	Sample *dmconfig* file for *Tuxedo JCA Adapter* configuration.
`javax.ejb_[version].jar`	EJB library

Table 2 RAR File Content

`javax.transaction_[version].jar`	JTA library
`tja.xsd`	*Tuxedo JCA Adapter* schema file for *Oracle* .
`META-INF/MANIFEST.MF`	Manifest file
`META-INF/client-side.ra.xml`	Sample client-side only resource adapter deployment descriptor
`META-INF/ra.xml`	Sample resource adapter deployment descriptor for connection factory based configuration
`META-INF/sample.weblogic-ra.xml`	Sample weblogic-ra.xml file for **WebLogic Server** for connection factory based configuration
`META-INF/server.ra.xml`	Sample resource adapter deployment descriptor for *dmconfig* based configuration
`META-INF/weblogic-ra.xml`	Sample weblogic-ra.xml file for **WebLogic Server**

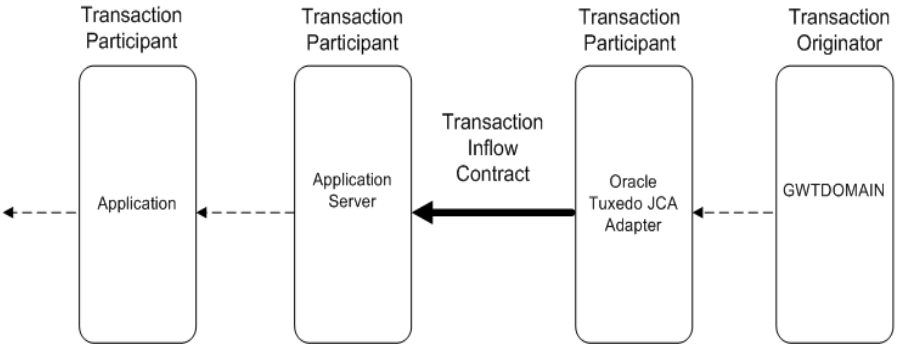
Inflow Transaction Overview

The Tuxedo JCA Adapter supports Oracle Tuxedo TDOMAIN protocol including its transaction as shown in Figure 1. When the Tuxedo JCA Adapter receives an inbound request from Oracle Tuxedo, it checks whether there is an associated Oracle Tuxedo transaction context or not. If it does have it, then the Tuxedo JCA Adapter creates a `javax.transaction.xa.Xid` based on the Oracle Tuxedo transaction context.

The Tuxedo JCA Adapter supplies this XID to an *ExecutionContex*t and submits the *Work* instance along with the *ExecutionContext* to the application server *WorkManager* for execution. By propagating an imported transaction to a Java application server this way, the application server and subsequent participants can work as part of the imported transaction.

Note: JCA 1.5 compliant Application Server uses ExecutionContext but for JCA 1.6 compliant Application Server uses TransactionContext.

Figure 1 Oracle Tuxedo TDOMAIN Protocol



Inflow Transaction Configuration

Configuration can be separated into two parts.

- Tuxedo JCA Adapter Configuration
- Application Server Resource Adapter Configuration

Tuxedo JCA Adapter Configuration

An “Exported” service is a Java resource that can be accessed by an Oracle Tuxedo client; in this particular case it is the *connector based MDB*. You must configure the “Export” element in the `dmconfig` file for an Oracle Tuxedo client to access resources located in the Java application server.

A single “Export” element in the `dmconfig` file refers to an exported resource to the Oracle Tuxedo client. Listing 1 shows two exported services (`Tolower` and `Echo`), to an Oracle Tuxedo client. The `RemoteName` is the service name the Oracle Tuxedo GWTDOMAIN gateway uses to invoke the service; the `name` attribute is the service name of the resource. The `Type` must be MDB for inflow transaction, and the `Source` is the JNDI binding of the MDB.

Listing 1 Exported Services Example

```
…
<Export name=”Tolower”>
  <RemoteName>TolowerMDB</RemoteName>
```

```
  <SessionName>session_1</SessionName>
  <Type>MDB</Type>
  <Source>eis/Tolower</Source>
</Export>
<Export name="Echo">
  <RemoteName>EchoMDB</RemoteName>
  <SessionName>session_1</SessionName>
  <Type>MDB</Type>
  <Source>eis/Echo</Source>
</Export>
…
```

Multiple exported services using single MDB is also supported. The purpose is to give greater freedom to the adapter application developer. You can configure them using the same JNDI name specified in the *dmconfig* file `Source` element; however, since there is only one interface implemented by the application for that MDB, the application must do the dispatching itself. Listing 2 shows an example of multiple exported services using single MDB.

Listing 2 Multiple Exported Services Using Single MDB Example

```
…
<Export name="INFO_SERVICE">
  <RemoteName>INFO</RemoteName>
  <SessionName>session_1</SessionName>
  <Type>MDB</Type>
  <Source>eis/services</Source>
</Export>
<Export name="ACCOUNT_SERVICE">
  <RemoteName>ACCOUNT</RemoteName>
```

```
  <SessionName>session_1</SessionName>
  <Type>MDB</Type>
  <Source>eis/services</Source>
</Export>
…
```

This example exports two services `INFO` and `ACCOUNT` to an Oracle Tuxedo client using the same MDB that binds to JNDI name `eis/services`. In this case you must create and deploy one MDB that dispatches using the service name passed to the MDB. Listing 3 shows an example MDB Code Fragment doing its own dispatching.

Listing 3 MDB Code Fragment

```
…
public Reply service(TPServiceInformation mydata)
    throws TuxedoReplyException
{
  String serviceName = mydata.getServiceName();
  if (serviceName.equals(“ACCOUNT_SERVICE”)) {
    doAccount1(mydata);
  }
   else if (serviceName.equals(“INFO_SERVICE”)) {
    doInfo(mydata);
  }
  else {
    /* throws an exception */
  }
}
```

Listing 4 shows a complete Tuxedo JCA Adapter configuration file example.

Listing 4 Tuxedo JCA Adapter Configuration File Example

```
<?xml version="1.0" encoding="UTF-8"?><TuxedoConnector>
  <LocalAccessPoint name="JDOM">
    <AccessPointId>JDOM_ID</AccessPointId>
    <NetworkAddress>//localhost:10801</NetworkAddress>
  </LocalAccessPoint>
  <RemoteAccessPoint name="TDOM1">
    <AccessPointId>TDOM1_ID</AccessPointId>
    <NetworkAddress>//localhost:12478</NetworkAddress>
  </RemoteAccessPoint>
  <SessionProfile name="profile_1">
    <BlockTime>60000</BlockTime>
    <ConnectionPolicy>ON_STARTUP</ConnectionPolicy>
  </SessionProfile>
  <Session name="session_1">
    <LocalAccessPointName>JDOM</LocalAccessPointName>
    <RemoteAccessPointName>TDOM1</RemoteAccessPointName>
    <ProfileName>profile_1</ProfileName>
  </Session>
  <Export name="Tolower">
    <RemoteName>TolowerMDB</RemoteName>
    <SessionName>session_1</SessionName>
    <Type>MDB</Type>
```

```
    <Source>eis/tolower</Source>
  </Export>
  <Export name="Echo">
    <RemoteName>EchoMDB</RemoteName>
    <SessionName>session_1</SessionName>
    <Type>MDB</Type>
    <Source>eis/echo</Source>
  </Export>
  <Export name="INFO_SERVICE">
    <RemoteName>INFO</RemoteName>
    <SessionName>session_1</SessionName>
    <Type>MDB</Type>
    <Source>eis/services</Source>
  </Export>
  <Export name="ACCOUNT_SERVICE">
    <RemoteName>ACCOUNT</RemoteName>
    <SessionName>session_1</SessionName>
    <Type>MDB</Type>
    <Source>eis/services</Source>
  </Export>
</TuxedoConnector>
```

Multiple Exported Services Using Multiple MDB Support

Multiple exported service using multiple is also supported. You can configure them using different JNDI name specified in "source" element in the dmconfig; and the Tuxedo JCA Adapter will do the dispatching base on the value in "source" element. Listing 5 shows a `dmconfig` file configuration example.

Listing 5 Multiple Exported Services Using Multiple MDB Support dmconfig File

```
…
<Export name="SERVICE_1">
  <RemoteName>TJA_SERVICE_1</RemoteName>
  <Type>MDB</Type>
  <Source>eis/service</Source>
</Export>
</Export name="SERVICE_2">
  <RemoteName>TJA_SERVICE_2</RemoteName>
  <Type>MDB</Type>
  <Source>eis/service</Source>
</Export>
<Export name="INFO">
  <RemoteName>TJA_INFO</RemoteName>
  <Type>MDB</Type>
  <Source>eis/info</Source>
</Export>
</TuxedoConnector>
```

With the above example, you must to create and deploy two MDBs:

- one for `SERVICE_1` and `SERVICE_2` with its JNDI bindsing to "`eis/service`",
- the other for `INFO` with its JNDI binding to "`eis/info`".

The 1st MDB must dispatch `SERVICE_1` and `SERVICE_2` internally.

Resource Adapter Deployment Descriptor Configuration

You must configure the Resource Adapter Deployment Descriptor (`ra.xml`). The name, `ra.xml`, cannot be changed. Every RAR file must contain one `ra.xml` file. For inflow

transactions using MDB to work, you must configure the `inbound-resourceadapter` element. This element is used to describe the interface and activation specification specific to the Tuxedo JCA Adapter.

The `inbound-resourceadapter` element is fixed. The `source` property is the only property that you can configure. If configured, the JCA container requires the `source` property to be specified in the EJB descriptor (`ejb-jar.xml`), file.

Listing 6 shows an `ra.xml` file example. You can use the ra.xml file distributed with the Tuxedo JCA Adapter as a base and customize it as needed.

Listing 6 ra.xml File Example

```
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/connector_1_5.xsd"
    version="1.5">
  <display-name>Tuxedo JCA Adapter</display-name>
  <vendor-name>Oracle</vendor-name>
  <eis-type>Tuxedo</eis-type>
  <resourceadapter-version>12c(12.1.1)</resourceadapter-version>
  <license>
    <description>Tuxedo SALT license</description>
    <license-required>false</license-required>
  </license>
  <resourceadapter>

<resourceadapter-class>com.oracle.tuxedo.adapter.TuxedoResourceAdapter</re
sourceadapter-class>
    <!--
```

```
      The following is the list of properties name can be configured as
adapter-wise configuration.

      traceLevel  - java.lang.String  - a numerical value

      xaAffinity  - java.lang.String  - transaction affinity to a remote
domain, "true" or "false", default to true

      keyFileName - java.lang.String  - encryption key file name

      throwFailureReplyException - java.lang.Boolean - default to ture

     appManagedLocalTxTimeout   - java.lang.Integer - Application managed
transaction or AUTOTRAN timeout

                                                    defaults to 300 seconds

     fieldTable16Class - java.lang.String - a comma-separated list of fully
qualified FML classes

     fieldTable32class - java.lang.String - a comma-separated list of fully
qualified FML32 classes

     viewFile16Class   - java.lang.String - a comma-separated list of fully
qualified VIEW classes

     viewFile32Class   - java.lang.String - a comma-separated list of fully
qualified VIEW32 classes

      tpusrFile         - java.lang.String - path name to the TPUSR file

     remoteMBEncoding  - java.lang.String - remote Tuxedo encoding name for
multi-byte language

     mBEncodingMapFile - java.lang.String - path name to Multi-byte encoding
name mapping

      autoTran          - java.lang.Boolean- enable adapter-wise AUTOTRAN,
default to false

    -->

    <outbound-resourceadapter>

      <connection-definition>


<managedconnectionfactory-class>com.oracle.tuxedo.adapter.spi.TuxedoManage
dConnectionFactory</managedconnectionfactory-class>
```

```
        <!--
          The following is the list of properties that you can use
          to configure thee connection pool or connection factory.
          You must either configure localAccessPointSpec or
          connectionFactoryName if transaction is used.
        These property described here is serving as template, user should not
          configure them here, instead user should configure them either
through WebSphere console
          or weblogic-ra.xml side file.
        -->
        <config-property>
          <description>factory-wise AUTOTRAN setting, default to false,
overrides adapter-wise setting</description>
          <config-property-name>autoTran</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
         <description>factory-wise Failure Reply Exception setting, default
to true, overrides adapter-wise setting</description>

<config-property-name>throwFailureReplyException</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise application managed transaction or
AUTOTRAN time out, overrides adapter-wise setting</description>

<config-property-name>appManagedLocalTxTimeout</config-property-name>
          <config-property-type>java.lang.Integer</config-property-type>
```

```
        </config-property>
        <config-property>
         <description>connection factory or pool name, this is required if
XA or local application managed
                        transaction is required</description>
        <config-property-name>connectionFactoryName</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>application password in either clear text or cipher
text using com.oracle.tuxedo.tools.EncryptPassword tool</description>
          <config-property-name>applicationPassword</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>local access point specification of the format
//hostname:port/domainId=DOMAINID</description>
         <config-property-name>localAccessPointSpec</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise SSL to configure whether mutual
authentication is required, default to false</description>

<config-property-name>mutualAuthenticationRequired</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
```

```
          <description>factory-wise SSL for configuring identity key store
file name, must be configured if SSL is desired</description>

<config-property-name>identityKeyStoreFileName</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise SSL setting for private key alias used
in the key store, must be configured if SSL is desired</description>
          <config-property-name>privateKeyAlias</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise trusted key store file name, must be
configured if SSL is desired</description>

<config-property-name>trustedKeyStoreFileName</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise password for identityKeyStore in clear
text</description>

<config-property-name>identityKeyStorePassPhrase</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise password for privateKeyAlias in clear
text</description>
```

```
<config-property-name>privateKeyAliasPassPhrase</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise password for trustedKeyStore in clear
text</description>

<config-property-name>trustedKeyStorePassPhrase</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise RemoteAccessPoint specification of the
format //hostname:port/domainId=DOMAINID</description>
        <config-property-name>remoteAccessPointSpec</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
        <description>factory-wise allow anonymous access to Tuxedo, default
to false</description>
          <config-property-name>rapAllowAnonymous</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
        <description>factory-wise application key value for anonymous user,
default to -1</description>

<config-property-name>rapDefaultApplicationKey</config-property-name>
          <config-property-type>java.lang.Integer</config-property-type>
```

```
        </config-property>
        <config-property>
          <description>factory-wise application key fully qualified class
name for AppKey generator</description>

<config-property-name>rapApplicationKeyClass</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise custom application key
parameter</description>

<config-property-name>rapApplicationKeyClassParam</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise session profile block timeout value,
default to 60000 milliseconds</description>
          <config-property-name>spBlockTime</config-property-name>
          <config-property-type>java.lang.Integer</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise whether allows interoperate with 6.5
Tuxedo Domain, default to false</description>
          <config-property-name>spInteroperate</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
```

```
          <description>factory-wise security setting, legal values: NONE,
DM_PW, APP_PW</description>
          <config-property-name>spSecurity</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise credential propagation policy, either
LOCAL or GLOBAL</description>
          <config-property-name>spCredentialPolicy</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise number of seconds that session waits
between automatic connection establishment,
                    default to 60 seconds. A value of 0 disabled connection
retry</description>
          <config-property-name>spRetryInterval</config-property-name>
          <config-property-type>java.lang.Long</config-property-type>
        </config-property>
        <config-property>
         <description>factory-wise maximum number of times adapter will try
to establish a session connection to
                        remote Tuxedo access point. Default value is
Long.MAX_VALUE.</description>
          <config-property-name>spMaxRetries</config-property-name>
          <config-property-type>java.lang.Long</config-property-type>
        </config-property>
        <config-property>
```

```
          <description>factory-wise compression threshold, default to
Integer.MAX_VALUE</description>
          <config-property-name>spCompressionLimit</config-property-name>
          <config-property-type>java.lang.Integer</config-property-type>
        </config-property>
        <config-property>
        <description>factory-wise minimum encryption strength requirement,
legal values are 0, 40, 56, 128, 256.
                        Default value is 0.</description>
          <config-property-name>spMinEncryptBits</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
        <description>factory-wise maximum encryption strength requirement,
legal values are 0, 40, 56, 128, 256.
                        Default value is 128.</description>
          <config-property-name>spMaxEncryptBits</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise the maximum idle time before sending
application level keep alive.
                    It is measured in millisecond, and roundup to seconds.
Default value is 0.</description>
          <config-property-name>spKeepAlive</config-property-name>
          <config-property-type>java.lang.Long</config-property-type>
        </config-property>
        <config-property>
```

```
          <description>factory-wise how long adapter will wait for
acknowledgement before adapter decides the

                     connection already lost.  Measurement in millisecond,
and its default value is 10 seconds.

                      A value of 0 will disable the wait, and thus will not
close the connection</description>

          <config-property-name>spKeepAliveWait</config-property-name>

          <config-property-type>java.lang.Long</config-property-type>

        </config-property>

        <config-property>

          <description>factory-wise valid Tuxedo service names in a
comma-separated list.  If not specified then

                     default import will be used and will grant all service
request to remote Tuxedo domain</description>

          <config-property-name>impResourceName</config-property-name>

          <config-property-type>java.lang.String</config-property-type>

        </config-property>

        <config-property>

          <description>Exported resource, types of resource supported are
EJB, POJO, MDB.</description>

          <config-property-name>exportSpec</config-property-name>

          <config-property-type>java.lang.String</config-property-type>

        </config-property>


<connectionfactory-interface>javax.resource.cci.ConnectionFactory</connect
ionfactory-interface>


<connectionfactory-impl-class>com.oracle.tuxedo.adapter.cci.TuxedoConnecti
onFactory</connectionfactory-impl-class>
```

```
<connection-interface>javax.resource.cci.Connection</connection-interface>

<connection-impl-class>com.oracle.tuxedo.adapter.cci.TuxedoJCAConnection</
connection-impl-class>
      </connection-definition>
<!--
      <transaction-support>NoTransaction</transaction-support>
      <transaction-support>LocalTransaction</transaction-support>
-->
      <transaction-support>XATransaction</transaction-support>
      <authentication-mechanism>

<authentication-mechanism-type>BasicPassword</authentication-mechanism-typ
e>

<credential-interface>javax.resource.spi.security.PasswordCredential</cred
ential-interface>
      </authentication-mechanism>
      <reauthentication-support>false</reauthentication-support>
    </outbound-resourceadapter>
    <inbound-resourceadapter>
      <messageadapter>
        <messagelistener>

<messagelistener-type>com.oracle.tuxedo.adapter.intf.TuxedoMDBService</mes
sagelistener-type>
          <activationspec>
```

```
<activationspec-class>com.oracle.tuxedo.adapter.spi.TuxedoActivationSpec</
activationspec-class>
            <required-config-property>
              <config-property-name>source</config-property-name>
            </required-config-property>
          </activationspec>
        </messagelistener>
      </messageadapter>
    </inbound-resourceadapter>
  </resourceadapter>
</connector>
```

The inbound-resourceadapter element contains the interface class that must be implement in the connector based MDB and the activation specification class.

. The fully qualified interface name is `com.oracle.tuxedo.adapter.intf.TuxedoMDBService`. The fully qualified activation specification is `com.oracle.tuxedo.adapter.spi.TuxedoActivationSpec`. *You must not* change any one of these two values in the Tuxedo JCA Adapter `ra.xml` file.

Resource Adapter Deployment Descriptor Configuration for Inbound Service Requests

At runtime, the Tuxedo JCA Adapter internally uses a union of the configured ResourceAdapter and ActivationSpec Java Bean properties, to represent inbound communication configuration.

For application-based dispatching method, there is no `ActivationSpec` Java Bean property needs to be configured. For adapter-based dispatching, you must specify the required configuration property "source" in the Resource Adapter Deployment Descriptor.

The following configuration fragment is in the Resource Adapter Deployment Descriptor, `ra.xml`, to enable Oracle Tuxedo JCA Connector to accept inbound service request:

Listing 7 Inbound Service Request Example

```
<connector...>
    ...
    <resourceadapter>

<resourceadapter-class>com.oracle.tuxedo.adapter.TuxedoResourceAdapterImpl
      </resourceadapter-class>
      ...
      <outbound-resourceadapter>
        …
      </outbound-resourceadapter>
      <inbound-resourceadapter>
        <messageadapter>
          <messagelistener>
            <messagelistener-type>
              com.oracle.tuxedo.adapter.intf.TuxedoMDBService
            </messagelistener-type>
            <activationspec>
              <activationspec-class>
              com.oracle.tuxedo.adapter.spi.TuxedoActivationSpec
              <activationspec-class>
            </activationspec>
          </messagelistener>
        </messageadapter>
      </inbound-resourceadapter>
    ...
    </resourceadapter>
```

```
</connector>
```

The above Resource Adapter Deployment Descriptor only suitable for user who only wants to implement and deploy single MDB based on TuxedoMDBService interface and does the dispatching by using the service name inside its MDB.

The following Resource Adapter Deployment Descriptor fragment shows configuration suitable for Tuxedo JCA Adapter to dispatch the inbound MDB service request.

Listing 8 Inbound MDB Service Request

```
<connector...>
    ...
    <resourceadapter>

<resourceadapter-class>com.oracle.tuxedo.adapter.TuxedoResourceAdapterImpl
      </resourceadapter-class>
      ...
      <outbound-resourceadapter>
        …
      </outbound-resourceadapter>
      <inbound-resourceadapter>
        <messageadapter>
          <messagelistener>
            <messagelistener-type>
              com.oracle.tuxedo.adapter.intf.TuxedoMDBService
            </messagelistener-type>
            <activationspec>
              <activationspec-class>
              com.oracle.tuxedo.adapter.spi.TuxedoActivationSpec
```

```
            <activationspec-class>
            <required-config-property>
              <config-property-name>source</config-property-name>
            </required-config-property>
          </activationspec>
        </messagelistener>
      </messageadapter>
    </inbound-resourceadapter>
  ...
  </resourceadapter>
</connector>
```

dmconfig Configuration

To differentiate a none JMS-based MDB to an ordinary EJB, you must configure "`Export`" with type "MDB". For application-based dispatch, "`Source`" and "`SourceLocation`" are not needed and can be left out; if specified they are ignored.

Listing 9 shows an application-based dispatch dmcomfig example.

Listing 9 dmconfig Configuration

```
…
<Export name="TOLOWER">
  <RemoteName>TJA_TOLOWER</RemoteName>
  <SessionName>session_1</SessionName>
  <Type>MDB</Type>
</Export>
…
```

The following is an example to enable and use Tuxedo JCA Adapter based dispatching.

Listing 10 Enabling Tuxedo JCA Adapter Based Dispatching

```
…
<Export name="TOLOWER">
  <RemoteName>TJA_TOLOWER</RemoteName>
  <SessionName>session_1</SessionName>
  <Type>MDB</Type>
  <Source>eis/tolower</Source>
</Export>
…
```

Here, the "Source" element contains the JNDI name of the targeted MDB (required by the Tuxedo JCA Adapter to dispatch the inbound service request).

Factory-Based Configuration

To support none JMS-based MDB for inbound service request using Factory Based Configuration a new property, "exportSpec", is added. Listing 11 defines the recursive descend grammar for "`exportSpec`".

Listing 11 exportSpec

```
exportSpec        ::= expResources | EMPTY
expResources    ::= resource [resources]
resources           ::= ';' resource  [resources]
resource            ::=  resourceName [resourceDesc]
resourceDesc    ::=  '(' attrDescs ')'
attrDescs             ::= attrDesc ':' attrDescs
attrDesc                ::= attrName '=' [resourceName|resourceNameList]
```

```
resourceNameList ::= resourceName [resourceNames]
resourceNames      ::= ',' resourceName [resourceNames]
resourceName       ::= alphabet [alphanumeric]
attrName               ::= "remoteName" | "remoteAccessPointList" |
                                    "type" | "source" | "sourceLocation"
alphabet                 ::= "a"-"z" "A"-"Z"
alphanumeric         ::= {alphabet | numeric} [alphanumeric]
numeric                  ::= "0"-"9"
```

Listing 12 shows a Factory Based configuration example.

Listing 12 Factory Based Configuration Example

```
<properties>
   …
  <property>
    <name>exportSpec</name>
    <value>TOLOWER(remoteName=TJA_TOLOWER:type=MDB)</value>
   </property>
…
</properties>
```

This type of configuration is not supported for WebSphere; for WebLogic Server and JBoss it will require outbound resource adapter also being configured.

Note: Due to WebSphere late binding approach the factory based configuration method will only work if there is a connection factory lookup before any inbound MDB request from Oracle Tuxedo.

Listing 13 is another Factory-Based configuration example that also configures the JNDI name of the MDB.

Listing 13 Factory-Based Configuration with MDB JNDI Name

```
<properties>
   …
  <property>
    <name>exportSpec</name>

<value>TOLOWER(remoteName=TJA_TOLOWER:type=MDB:source=eis/tolower)</value>
   </property>
…
</properties
```

TuxedoMDBService Interface

The Tuxedo JCA Adapter provides an EJB MDB interface that must be implemented with none JMS-based MDB. The MDB interface is similar to the existing EJB supported by Tuxedo JCA Adapter; however, they are not the same.

Listing 14 is an MDN interface example.

Listing 14 MDB Interface

```
package com.oracle.tuxedo.adapter.intf;
import weblogic.wtc.jatmi.Reply;
import com.oracle.tuxedo.adapter.tdom.TPServiceInformation;
import com.oracle.tuxedo.adapter.TuxedoReplyException;
public interface TuxedoMDBService {
```

```
public Reply service(TPServiceInformation service) throws
TuxedoReplyException;
}
```

Note: This is different from JMS based MDB, it has a "service" interface instead of "onMessage" interface.

Listing 15 is shows an MDB code example based on the EJB 3.0 specification that implements this interface.

Listing 15 EJB 3.0 specification

```
@MessageDriven(name = "TolowerMDB", activationConfig =
{
@ActivationConfigProperty(propertyName="source",
propertyValue="eis/tolower")
})
@TransactionAttribute(SUPPORTS)
public class TolowerMDBBean implements TuxedoMDBService
{
  public Reply service(TPServiceInformation mydata)
    throws TuxedoReplyException
  {
     TypedString  data;
     String            lowered;
      TypedString return_data;
    String serviceName = mydata.getServiceName();
      data = (TypedString)mydata.getServiceData();
      lowered = data.toString().toLowerCase();
      return_data = new TypedString(lowered);
```

```
      mydata.setReplyBuffer(return_data);
       return mydata;`
  }
}
```

Listing 16 shows an EJB 2.1 example that does exactly the same thing.

Listing 16 sample based on EJB 2.1

```
package ejbs;
import com.oracle.tuxedo.adapter.TuxedoReplyException;
import com.oracle.tuxedo.adapter.intf.TuxedoMDBService;
import com.oracle.tuxedo.adapter.tdom.TPServiceInformation;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import javax.jms.Message;
import weblogic.wtc.jatmi.Reply;
import weblogic.wtc.jatmi.TypedString;

public class TolowerMDBBeanBean
    implements MessageDrivenBean, TuxedoMDBService
{

    public TolowerMDBBeanBean()
    {
    }
```

```
public MessageDrivenContext getMessageDrivenContext()
{
    return fMessageDrivenCtx;
}

public void setMessageDrivenContext(MessageDrivenContext ctx)
{
    fMessageDrivenCtx = ctx;
}

public void ejbCreate()
{
}

public void onMessage(Message message)
{
}

public void ejbRemove()
{
}

public Reply service(TPServiceInformation mydata)
    throws TuxedoReplyException
{
    TypedString data = (TypedString)mydata.getServiceData();
    String lowered = data.toString().toLowerCase();
```

```
        TypedString return_data = new TypedString(lowered);
        mydata.setReplyBuffer(return_data);
        return mydata;
    }

    private static final long serialVersionUID = 1L;
    private MessageDrivenContext fMessageDrivenCtx;
}
```

Oracle Tuxedo GWTDOMAIN Gateway Configuration for Inbound MDB

You must also configure the Oracle Tuxedo GWTDOMAIN gateway to communicate with the Tuxedo JCA Adapter. Listing 17 shows an Oracle Tuxedo /Domain configuration file example.

Listing 17 Tuxedo /Domain Configuration File Example

```
#
*DM_RESOURCES
#
VERSION=U22
#


#
#
*DM_LOCAL_DOMAINS
#
# NOTE: Remove DYNAMIC_RAP line if you are not running with Tuxedo 11.1.1.2.0
```

```
#
"TDOM1"    GWGRP=GROUP3
       TYPE=TDOMAIN
       DOMAINID="TDOM1_ID"
       BLOCKTIME=60
       SECURITY=NONE
            DMTLOGDEV="C:\test\JCA\inflow_tx/tdom/DMTLOG"
            DYNAMIC_RAP="YES"

#
*DM_REMOTE_DOMAINS
#
#
JDOM   TYPE=TDOMAIN
       DOMAINID="JDOM_ID"

#
#
*DM_TDOMAIN
#
TDOM1  NWADDR="//localhost:12478"
JDOM   NWADDR="//localhost:10801"
#
#
*DM_LOCAL_SERVICES
#
#Exported
```

```
#
#
*DM_REMOTE_SERVICES
#
#Imported
#
TolowerMDB
EchoMDB
INFO
ACCOUNT
```

In this example, Oracle Tuxedo *imports* the services `TolowerMDB`, `EchoMDB`, `INFO`, and `ACCOUNT`; while the Tuxedo JCA Adapter *exports* them.

Inflow Transaction Adapter Deployment

On the WebSphere Integrated Solution Console, enter https://localhost:9047/ibm/console/logon.jsp (where 9047 is the port number your application server is listening on).

Configuring the Tuxedo JCA Adapter *dmconfig* File for Websphere

Before deploying the Tuxedo JCA Adapter for WebSphere application server, a `dmconfig` configuration file must be created. Listing 18 shows a `dmconfig` file example.

Listing 18 dmconfig File Websphere Example

```
<?xml version="1.0" encoding="UTF-8"?><TuxedoConnector>
  <LocalAccessPoint name="JDOM">
    <AccessPointId>JDOM_ID</AccessPointId>
```

```
    <NetworkAddress>//localhost:10801</NetworkAddress>
  </LocalAccessPoint>
  <RemoteAccessPoint name="TDOM1">
    <AccessPointId>TDOM1_ID</AccessPointId>
    <NetworkAddress>//localhost:12478</NetworkAddress>
  </RemoteAccessPoint>
  <SessionProfile name="profile_1">
    <BlockTime>60000</BlockTime>
    <ConnectionPolicy>ON_STARTUP</ConnectionPolicy>
  </SessionProfile>
  <Session name="session_1">
    <LocalAccessPointName>JDOM</LocalAccessPointName>
    <RemoteAccessPointName>TDOM1</RemoteAccessPointName>
    <ProfileName>profile_1</ProfileName>
  </Session>
  <Export name="ECHOMDB">
    <RemoteName>ECHO</RemoteName>
    <SessionName>session_1</SessionName>
    <Type>MDB</Type>
   <Source>eis/echo</Source>
  </Export>
</TuxedoConnector>
```

Resource Adapter Deployment Descriptor

You can either create the Resource Adapter Deployment Descriptor from scratch or modify an existing one. Listing 8 shows a *Deploy Descriptor ex*ample.

Listing 19 Deploy Descriptor Example

```
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/connector_1_5.xsd"
    version="1.5">
  <display-name>Tuxedo JCA Adapter</display-name>
  <vendor-name>Oracle</vendor-name>
  <eis-type>Tuxedo</eis-type>
  <resourceadapter-version>12c(12.1.1)</resourceadapter-version>
  <license>
    <description>Tuxedo SALT license</description>
    <license-required>false</license-required>
  </license>
  <resourceadapter>

<resourceadapter-class>com.oracle.tuxedo.adapter.TuxedoResourceAdapter</re
sourceadapter-class>
    <!--
      The following is the list of properties name can be configured as
adapter-wise configuration.
      traceLevel  - java.lang.String  - a numerical value
      xaAffinity  - java.lang.String  - transaction affinity to a remote
domain, "true" or "false", default to true
      keyFileName - java.lang.String  - encryption key file name
      throwFailureReplyException - java.lang.Boolean - default to ture
     appManagedLocalTxTimeout   - java.lang.Integer - Application managed
transaction or AUTOTRAN timeout
```

```
                                                        defaults to 300 seconds
     fieldTable16Class - java.lang.String - a comma-separated list of fully
qualified FML classes
     fieldTable32class - java.lang.String - a comma-separated list of fully
qualified FML32 classes
     viewFile16Class   - java.lang.String - a comma-separated list of fully
qualified VIEW classes
     viewFile32Class   - java.lang.String - a comma-separated list of fully
qualified VIEW32 classes
      tpusrFile         - java.lang.String - path name to the TPUSR file
     remoteMBEncoding  - java.lang.String - remote Tuxedo encoding name for
multi-byte language
     mBEncodingMapFile - java.lang.String - path name to Multi-byte encoding
name mapping
      autoTran          - java.lang.Boolean- enable adapter-wise AUTOTRAN,
default to false
    -->
    <config-property>
      <config-property-name>traceLevel</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value>2000000</config-property-value>
    </config-property>
<!--
    <config-property>
      <config-property-name>xaAffinity</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value>true</config-property-value>
    </config-property>
-->
```

```
    <config-property>
      <config-property-name>dmconfig</config-property-name>
      <config-property-type>java.lang.String</config-property-type>

<config-property-value>C:\test\JCA\inflow_tx/adapter/dmconfig.xml</config-
property-value>
    </config-property>
    <config-property>
      <config-property-name>keyFileName</config-property-name>
      <config-property-type>java.lang.String</config-property-type>

<config-property-value>C:\test\JCA\inflow_tx/adapter/foo.key</config-prope
rty-value>
    </config-property>
    <config-property>
      <config-property-name>debugAdapter</config-property-name>
      <config-property-type>java.lang.Boolean</config-property-type>
      <config-property-value>true</config-property-value>
    </config-property>
    <config-property>
      <config-property-name>debugJatmi</config-property-name>
      <config-property-type>java.lang.Boolean</config-property-type>
      <config-property-value>true</config-property-value>
    </config-property>
    <config-property>
      <config-property-name>debugConfig</config-property-name>
      <config-property-type>java.lang.Boolean</config-property-type>
      <config-property-value>true</config-property-value>
```

```
        </config-property>
        <config-property>
          <config-property-name>debugSession</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
          <config-property-value>true</config-property-value>
        </config-property>
        <config-property>
          <config-property-name>debugXa</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
          <config-property-value>true</config-property-value>
        </config-property>
        <config-property>
          <config-property-name>debugPdu</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
          <config-property-value>true</config-property-value>
        </config-property>
        <config-property>
          <config-property-name>debugSec</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
          <config-property-value>true</config-property-value>
        </config-property>
    <!--
    -->
        <outbound-resourceadapter>
          <connection-definition>

    <managedconnectionfactory-class>com.oracle.tuxedo.adapter.spi.TuxedoManage
    dConnectionFactory</managedconnectionfactory-class>
```

```
        <!--
          The following is the list of properties that you can use to
          to configure connection pool or connection factory.
          User must either configure localAccessPointSpec or
          connectionFactoryName if transaction is used.
        These property described here is serving as template, user should not
          configure them here, instead user should configure them either
through WebSphere console
          or weblogic-ra.xml side file.
        -->
        <config-property>
          <description>factory-wise AUTOTRAN setting, default to false,
overrides adapter-wise setting</description>
          <config-property-name>autoTran</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
         <description>factory-wise Failure Reply Exception setting, default
to true, overrides adapter-wise setting</description>

<config-property-name>throwFailureReplyException</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise application managed transaction or
AUTOTRAN time out, overrides adapter-wise setting</description>

<config-property-name>appManagedLocalTxTimeout</config-property-name>
          <config-property-type>java.lang.Integer</config-property-type>
```

```
        </config-property>
        <config-property>
         <description>connection factory or pool name, this is required if
XA or local application managed
                        transaction is required</description>
        <config-property-name>connectionFactoryName</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
         <description>application password in either clear text or cipher
text using com.oracle.tuxedo.tools.EncryptPassword tool</description>
          <config-property-name>applicationPassword</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>local access point specification of the format
//hostname:port/domainId=DOMAINID</description>
         <config-property-name>localAccessPointSpec</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise SSL to configure whether mutual
authentication is required, default to false</description>

<config-property-name>mutualAuthenticationRequired</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
```

```
          <description>factory-wise SSL for configuring identity key store
file name, must be configured if SSL is desired</description>

<config-property-name>identityKeyStoreFileName</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise SSL setting for private key alias used
in the key store, must be configured if SSL is desired</description>
          <config-property-name>privateKeyAlias</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise trusted key store file name, must be
configured if SSL is desired</description>

<config-property-name>trustedKeyStoreFileName</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise password for identityKeyStore in clear
text</description>

<config-property-name>identityKeyStorePassPhrase</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise password for privateKeyAlias in clear
text</description>
```

```
<config-property-name>privateKeyAliasPassPhrase</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise password for trustedKeyStore in clear
text</description>

<config-property-name>trustedKeyStorePassPhrase</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
         <description>factory-wise RemoteAccessPoint specification of the
format //hostname:port/domainId=DOMAINID</description>
        <config-property-name>remoteAccessPointSpec</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
       <description>factory-wise allow anonymous access to Tuxedo, default
to false</description>
          <config-property-name>rapAllowAnonymous</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
       <description>factory-wise application key value for anonymous user,
default to -1</description>

<config-property-name>rapDefaultApplicationKey</config-property-name>
          <config-property-type>java.lang.Integer</config-property-type>
```

```
        </config-property>
        <config-property>
          <description>factory-wise application key fully qualified class
name for AppKey generator</description>

<config-property-name>rapApplicationKeyClass</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise custom application key
parameter</description>

<config-property-name>rapApplicationKeyClassParam</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise session profile block timeout value,
default to 60000 milliseconds</description>
          <config-property-name>spBlockTime</config-property-name>
          <config-property-type>java.lang.Integer</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise whether allows interoperate with 6.5
Tuxedo Domain, default to false</description>
          <config-property-name>spInteroperate</config-property-name>
          <config-property-type>java.lang.Boolean</config-property-type>
        </config-property>
        <config-property>
```

```
          <description>factory-wise security setting, legal values: NONE,
DM_PW, APP_PW</description>
          <config-property-name>spSecurity</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise credential propagation policy, either
LOCAL or GLOBAL</description>
          <config-property-name>spCredentialPolicy</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise number of seconds that session waits
between automatic connection establishment,
                    default to 60 seconds. A value of 0 disabled connection
retry</description>
          <config-property-name>spRetryInterval</config-property-name>
          <config-property-type>java.lang.Long</config-property-type>
        </config-property>
        <config-property>
         <description>factory-wise maximum number of times adapter will try
to establish a session connection to
                        remote Tuxedo access point. Default value is
Long.MAX_VALUE.</description>
          <config-property-name>spMaxRetries</config-property-name>
          <config-property-type>java.lang.Long</config-property-type>
        </config-property>
        <config-property>
```

```
          <description>factory-wise compression threshold, default to
Integer.MAX_VALUE</description>
          <config-property-name>spCompressionLimit</config-property-name>
          <config-property-type>java.lang.Integer</config-property-type>
        </config-property>
        <config-property>
        <description>factory-wise minimum encryption strength requirement,
legal values are 0, 40, 56, 128, 256.
                        Default value is 0.</description>
          <config-property-name>spMinEncryptBits</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
        <description>factory-wise maximum encryption strength requirement,
legal values are 0, 40, 56, 128, 256.
                        Default value is 128.</description>
          <config-property-name>spMaxEncryptBits</config-property-name>
          <config-property-type>java.lang.String</config-property-type>
        </config-property>
        <config-property>
          <description>factory-wise the maximum idle time before sending
application level keep alive.
                    It is measured in millisecond, and roundup to seconds.
Default value is 0.</description>
          <config-property-name>spKeepAlive</config-property-name>
          <config-property-type>java.lang.Long</config-property-type>
        </config-property>
        <config-property>
```

```
          <description>factory-wise how long adapter will wait for
acknowledgement before adapter decides the

                     connection already lost.  Measurement in millisecond,
and its default value is 10 seconds.

                      A value of 0 will disable the wait, and thus will not
close the connection</description>

          <config-property-name>spKeepAliveWait</config-property-name>

          <config-property-type>java.lang.Long</config-property-type>

        </config-property>

        <config-property>

          <description>factory-wise valid Tuxedo service names in a
comma-separated list.  If not specified then

                     default import will be used and will grant all service
request to remote Tuxedo domain</description>

          <config-property-name>impResourceName</config-property-name>

          <config-property-type>java.lang.String</config-property-type>

        </config-property>

        <config-property>

          <description>Exported resources.  Types of resource supported
are</description>

          <config-property-name>exportSpec</config-property-name>

          <config-property-type>java.lang.String</config-property-type>

        </config-property>


<connectionfactory-interface>javax.resource.cci.ConnectionFactory</connect
ionfactory-interface>


<connectionfactory-impl-class>com.oracle.tuxedo.adapter.cci.TuxedoConnecti
onFactory</connectionfactory-impl-class>
```

```
<connection-interface>javax.resource.cci.Connection</connection-interface>

<connection-impl-class>com.oracle.tuxedo.adapter.cci.TuxedoJCAConnection</
connection-impl-class>
      </connection-definition>
<!--
      <transaction-support>NoTransaction</transaction-support>
      <transaction-support>LocalTransaction</transaction-support>
-->
      <transaction-support>XATransaction</transaction-support>
      <authentication-mechanism>

<authentication-mechanism-type>BasicPassword</authentication-mechanism-typ
e>

<credential-interface>javax.resource.spi.security.PasswordCredential</cred
ential-interface>
      </authentication-mechanism>
      <reauthentication-support>false</reauthentication-support>
    </outbound-resourceadapter>
    <inbound-resourceadapter>
      <messageadapter>
        <messagelistener>

<messagelistener-type>com.oracle.tuxedo.adapter.intf.TuxedoMDBService</mes
sagelistener-type>
          <activationspec>
```

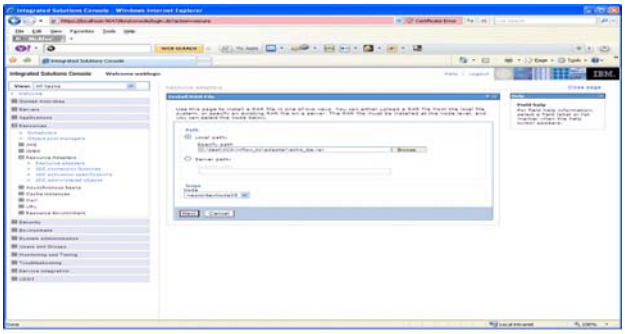
```
<activationspec-class>com.oracle.tuxedo.adapter.spi.TuxedoActivationSpec</
activationspec-class>
            <required-config-property>
              <config-property-name>source</config-property-name>
            </required-config-property>
          </activationspec>
        </messagelistener>
      </messageadapter>
    </inbound-resourceadapter>
  </resourceadapter>
</connector>
```

Deploying the Tuxedo JCA Adapter on Websphere

After jarring the resource adapter with deployment descriptor to create a Resource Archive, you can deploy it to a WebSphere application server.

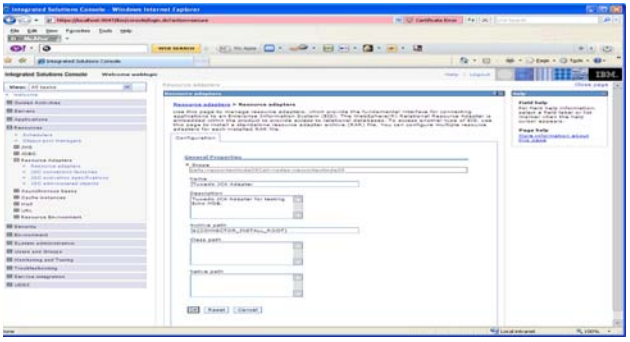
After logging in to the WebSphere Integrated Solution Console select **Resource** from the left window pane; select **Resource Adapters** as shown in Figure 2. The **Resource adapter** window appears. Click **Browse** to find your RAR file.

Figure 2 WebSphere Integrated Solution Console



Click **Next**; the **General Properties** page appears as shown in Figure 3. Enter the appropriate description in the **Description** text entry box.

Figure 3 General Properties Page

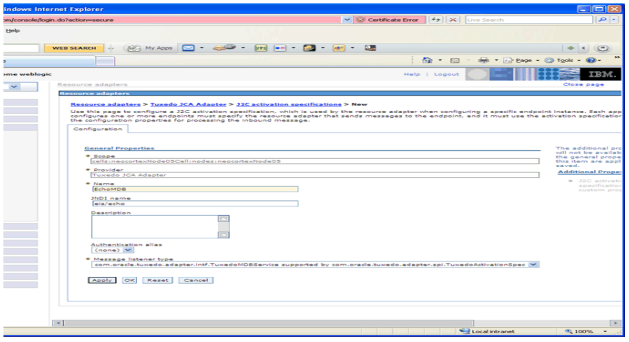


Click **OK**, then click **Save**.

Configure Activation Spec

From **Resource adapter > Tuxedo JCA Adapter**, select **J2C activation specification** under **Additional Properties**; the **J2C activation specification** page appears as shown in Figure 4. Select **New**; enter a name for the activation specification and its JNDI name (this example uses `EchoMDB` as name and `eis/echo` as its JNDI name. This JNDI name is the JNDI name `EchoMDB` uses.

Figure 4 J2C Activation Specification Page



Click **OK** to complete specification.

Configuring MDB Using WebSphere Integrated Console

Start your WebSphere application server and log in to WebSphere using the Integrated Solution Console. The console port number usually is 904X where "X" can be any digit.

Note: You can find a `logs/server1/SystemOut.log`. Look for "`TCP Channel TCP_3 is listening`."

Deploy MDB To WebSphere

Configure a Shared Library

On the left pane of console select **Environment**; this expands the menu item with a sub-menu. Select **Shared Library**; the **Shared Library** screen appears.

Click **New;** the configuration screen appears. Fill in **Name** with any name you like. For this example, enter **EchoMDBEnv** in the **Name** text entry box. In the Classpath window enter the full path name of the following two JAR files.

- `com.bea.core.jatmi_[version].jar`
- `com.oracle.tuxedo.adapter_[version].jar`

Use the Enter key as a separator then click **Save**.

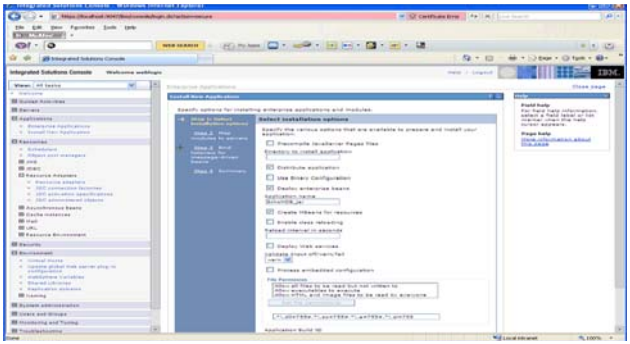
Install MDB

On left pane of the console, select **Applications**, then select **Install New Application**. The **Enterprise Application** menu appears.

In **Path to the new application** select **Local file system**. Use **Browse** to select the `EchoMDB.jar` file.

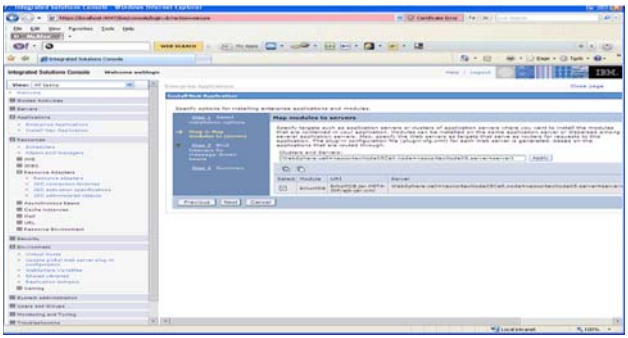
In **How do you want to install the application**, select **Show me all installation options and parameters**. Click **Next**; the **Select installation options** page appears as shown in Figure 5. Select **Deploy enterprise beans**, then click **Next**.

Figure 5 Select installation Options Page



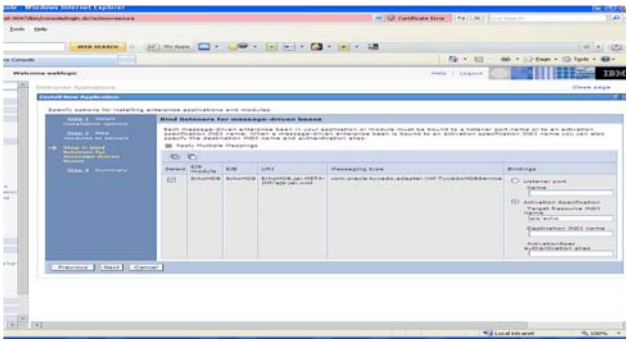
In the “Step 2: Map modules to servers” select the server where you want your Echo MDB be available. Place a check mark to “EchoMDB” then click **Apply**. Click **Next**.

Figure 6 Select EchoMDB Module



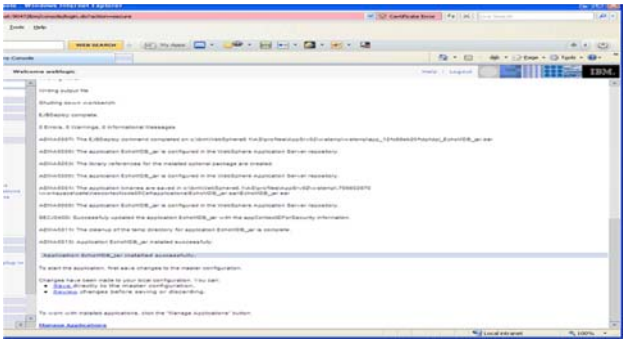
In the “Step 3: Bind listeners for message-driven beans” select **Activation Specification** as shown in Figure 7 then enter the JNDI name for this MDB (in this case, enter `eis/echo`).

Figure 7 Activation Specification



Click **Next**: the Summary page appears as shown in Figure 8. Click **Finish**. The application server compiles and deploy deploys the MDB.

Figure 8 Summary Page



Click **Save**.

Activate MDB

From the left pane of the console select **Applications** and then select **Enterprise Applications.** Select **EchoMDB.jar** and click **Star**t. **Echo" EJB** is activated.

Oracle Tuxedo Application Domain

Oracle Tuxedo Configuration

Listing 63 shows the Oracle Tuxedo UBBCONFIG file used in this example.

Listing 63 Oracle Tuxedo UBBCONFIG File Example

```
#
#Ubbconfig domain1
#

*RESOURCES
IPCKEY               51301

MASTER               site1
MAXACCESSERS100
MAXSERVERS    25
MAXSERVICES   50
MODEL                       SHM
LDBAL                       N
BLOCKTIME 1
SCANUNIT      5
SECURITY        NONE

*MACHINES
DEFAULT:
              APPDIR="C:\test\JCA\inflow_tx/tdom1"
              TUXCONFIG="C:\test\JCA\inflow_tx/tdom1/TUXCONFIG"
              TUXDIR="c:\tuxedo\tux11g"

"NEOCORTEX"LMID=site1
              MAXWSCLIENTS=2
```

```
*GROUPS
GROUP3 LMID=site1 GRPNO=3OPENINFO=NONE
GROUP2 LMID=site1 GRPNO=2OPENINFO=NONE
GROUP1 LMID=site1 GRPNO=1  TMSNAME=TMS TMSCOUNT=3
#GROUP1 LMID=site1 GRPNO=1

*SERVERS
DEFAULT:
              CLOPT="-A" RESTART=Y MAXGEN=5

DMADM         SRVGRP=GROUP2SRVID=1
GWADM         SRVGRP=GROUP3SRVID=2
GWTDOMAINSRVGRP=GROUP3SRVID=3
              ENVFILE="C:\test\JCA\inflow_tx/tdom1/gwt.env"
simpserv      SRVGRP=GROUP1SRVID=20

*SERVICES
TOUPPER_STR
```

Oracle Tuxedo/Domain Configuration

Listing 64 shows an i/Domain configuration example.

Listing 64 Oracle Tuxedo/Domain Configuration Example

```
#
*DM_RESOURCES
#
```

```
VERSION=U22
#
#
*DM_LOCAL_DOMAINS
#
# NOTE: Remove DYNAMIC_RAP line if you are not running with Tuxedo 11.1.1.2.0
#
"TDOM1"    GWGRP=GROUP3
       TYPE=TDOMAIN
       DOMAINID="TDOM1_ID"
       BLOCKTIME=60
       SECURITY=NONE
   DMTLOGDEV="C:\test\JCA\inflow_tx/tdom1/DMTLOG"
   DYNAMIC_RAP="YES"

#
*DM_REMOTE_DOMAINS
#
#
JDOM   TYPE=TDOMAIN
       DOMAINID="JDOM_ID"

#
#
*DM_TDOMAIN
#
TDOM1  NWADDR="//localhost:12478"
```

```
JDOM   NWADDR="//localhost:10801"
#
#
*DM_LOCAL_SERVICES
#
#Exported
#
TOUPPER_STR
#
*DM_REMOTE_SERVICES
#
#Imported
#
ECHO
```

Oracle Tuxedo Transactional Client Source Code

Listing 65 shows the simple Oracle Tuxedo native client that accesses the ECHO service imported from WebSphere application server.

Listing 65 ECHO Service Imported from WebSphere Application Server

```
#include <stdio.h>
#include “atmi.h”
main(int argc, char *argv[])
{
  char *sendbuf, *rcvbuf;
  long sendlen, rcvlen;
```

```
  int  ret;

  if (tpinit((TPINIT *)NULL) == -1) {
    (void)fprintf(stderr, “Tpinit failed\n”);
    exit(1);
  }
  sendlen = strlen(argv[1]);
  if ((sendbuf = (char *)tpalloc(“STRING”, NULL, sendlen + 1)) == NULL) {
      (void)fprintf(stderr, “Error allocating send buffer\n”);
      tpterm();
     exit(2);
  }
  if ((rcvbuf = (char *)tpalloc(“STRING”, NULL, sendlen + 1)) == NULL) {
      (void)fprintf(stderr, “Error allocating receive buffer\n”);
    tpfree(sendbuf);
      tpterm();
     exit(2);
  }
  (void)strcpy(sendbuf, argv[2]);
  tpbegin(45, 0);
  ret = tpcall(“ECHO”, (char *)sendbuf, 0, (char **)&rcvbuf, &rcvlen,
(long)0);
  if (ret == -1) {
    tpabort(0);
    tpfree(sendbuf);
    tpfree(recvbuf);
    tpterm();
    exit(1);
```

```
    }
    userlog("Return string: %s", rcvbuf);
    tpcommit(0);
    tpfree(sendbuf);
    tpfree(rcvbuf);
    tpterm();
    return(0);
}
```

See Also

- Tuxedo JCA Adapter Programming Guide
- Tuxedo JCA Adapter Users Guide