Oracle8*i* JVM

Deploying Enterprise JavaBeans to Oracle9*i* Application Server

Release 8.1.7

February 2001

Part No. A88705-01

In this release, Enterprise JavaBeans (EJB) 1.1 are fully supported. There are some changes to the Oracle specific deployment descriptor for EJBs that allow EJBs to function properly in both the Oracle9*i* Application Server middle tier and in the 8.1.7 database.

This document discusses the following topics:

Defining Oracle-Specific Elements for Transactions

There are three elements that you can further specify for your global transaction.

- Define whether the local resource is automatically enlisted in the transaction.
- Define a two-phase commit engine.
- Define whether branches are enabled in the global transaction.

The following shows the elements necessary for these elements, which are contained within the `<transaction-manager>` element in the Oracle-specific deployment descriptor.

```
<mappings>
   ...
   <transaction-manager>
   <jndi-name>/test/TransactionManager</jndi-name>
   <default-enlist>TRUE</default-enlist>
   <create-branches>FALSE</create-branches>
   </transaction-manager>
</mappings>
```

Defining Local Resource Enlistment for Transactions

The local resource can be a database or an Oracle9*i* Application Server middle-tier cache. If your EJBs are running in the Oracle 8.1.7 database and you want the local datasource to participate in a global transaction, you must specify `<default-enlist>` to be true.

```
<mappings>
   ...
   <transaction-manager>
      ...
   <default-enlist>TRUE</default-enlist>
   </transaction-manager>
</mappings>
```

If your EJBs are running in the Oracle9*i* Application Server middle tier, you do not want the Oracle9*i* Application Server cache to be enlisted with the transaction manager because the Oracle9*i* Application Server Database Cache is read-only. Since the default for this element is FALSE, you don't need to put the `<default-enlist>` tag in the deployment descriptor.

```
<mappings>
   ...
   <transaction-manager>
      ...
   </transaction-manager>
</mappings>
```

Defining Two-Phase Commit Engine

If you are using two-phase commit for your global transaction, you must provide the `UserTransaction` object's JNDI name to the `<transaction-manager>` element within the Oracle-specific deployment descriptor. The following example specifies the UserTransaction object `/test/myUTFor2pc`:

```
<mappings>
   ...
   <transaction-manager>
      <jndi-name>/test/myUTFor2pc</jndi-name>
   ...
   </transaction-manager>
</mappings>
```

Enabling Branches Within the Transaction

According to the X/Open XA protocol, if you want to apply two or more updates to a single database, the transaction manager monitors each update-known as a unit-of-work-through branches. The `<create-branches>` element defines whether more than one unit of

work can be performed on a single database. That is, if several separate updates are to be applied to a single database, this element must be set to true. If false, only a single unit of work is allowed per database within the global transaction.

You can specify that branches are allowed within the global transaction by specifying the `<create-branches>` element to TRUE, as follows:

```
<mappings>
   ...
   <transaction-manager>
      ...
   <create-branches>TRUE</create-branches>
   </transaction-manager>
</mappings>
```

Enabling branches is only supported when the global transaction uses

- multiple databases
- multiple sessions accessing a single database

Branching within a single transaction and session to the same database is not supported.

Accessing a single database from multiple sessions, branches will be used. However, this will result in a two-phase commit.

Defining Container-Managed Persistence

If you have chosen to use the container managed persistence entity bean, you can optionally provide a `<persistence-datasource>` where the persistent attributes should be accessed. If the `<persistence-datasource>` is not defined in the deployment descriptor, the EJB container uses the kprb driver to access the local database for the persistence attributes.

The following defines a persistence provider that uses a remote database to store the persistence fields. The `/test/empDatabase` JTA `DataSource` is defined in the following example as the remote persistence storage. This `DataSource` was bound by bindds with the `-type jta` option.

```
<persistence-provider>
   <description>specifies a type of persistence manager</description>
   <persistence-name>
      psi-ri
   </persistence-name>
   <persistence-deployer>
      oracle.aurora.ejb.persistence.ocmp.OcmpEntityDeployer
   </persistence-deployer>
```

```
   <persistence-datasource>
      /test/empDatabase
   </persistence-datasource>
</persistence-provider>
```

Changes in the Oracle-Specific DTD

The complete oracle specific DTD is listed below. Note that this includes the optional tags which were described above.

```
<!-- This is the XML DTD for the Oracle Specific EJB deployment descriptor -->
<!ELEMENT oracle-descriptor (mappings*, run-as*, persistence-provider*,
persistence-descriptor*)>
<!ELEMENT run-as (description?, mode, security-role*, method)>
<!ELEMENT method (description?, ejb-name, method-intf?, method-name,
method-params?)>
<!ELEMENT method-params (method-param*)>
<!ELEMENT method-intf (#PCDATA)>
<!ELEMENT method-name (#PCDATA)>
<!ELEMENT method-param (#PCDATA)>
<!ELEMENT mappings (ejb-mapping*, security-role-mapping*, resource-ref-mapping*,
transaction-manager*)>
<!ELEMENT transaction-manager (description?, jndi-name?, default-enlist?,
create-branches?)>
<!ELEMENT ejb-mapping (ejb-name, jndi-name)>
<!ELEMENT security-role-mapping (security-role, oracle-role)>
<!ELEMENT resource-ref-mapping (res-ref-name, jndi-name)>
<!ELEMENT ejb-name (#PCDATA)>
<!ELEMENT jndi-name (#PCDATA)>
<!ELEMENT default-enlist (#PCDATA)>
<!ELEMENT create-branches (#PCDATA)>
<!ELEMENT security-role (description?, role-name)>
<!ELEMENT role-name (#PCDATA)>
<!ELEMENT oracle-role (#PCDATA)>
<!ELEMENT ejb-ref-name (#PCDATA)>
<!ELEMENT res-ref-name (#PCDATA)>
<!--
   mode = SYSTEM_IDENTITY, SPECIFIED_IDENTITY, CLIENT_IDENTITY
   if mode is SPECIFIED_IDENTITY, security-role must be specified
   if mode is SYSTEM_IDENTITY or CLIENT_IDENTITY and security-role is
   specified, security-role is ignored
-->
<!ELEMENT mode (#PCDATA)>

<!--- persistence-provider describes the container managed persistence -->
<!ELEMENT persistence-provider (description?, persistence-name,
persistence-deployer, persistence-datasource?)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT persistence-name (#PCDATA)>
<!ELEMENT persistence-deployer (#PCDATA)>
<!ELEMENT persistence-datasource (#PCDATA)>
<!ELEMENT persistence-descriptor (description?, ejb-name, persistence-name,
persistence-param*, psi-ri*)>
```

```
<!ELEMENT persistence-param (#PCDATA)>
<!ELEMENT psi-ri (schema, table, attr-mapping+, serialize-mapping?)>
<!ELEMENT schema (#PCDATA)>
<!ELEMENT table (#PCDATA)>
<!ELEMENT attr-mapping (field-name, column-name)>
<!ELEMENT serialize-mapping (field-name+, column-name)>
<!ELEMENT field-name (#PCDATA)>
<!ELEMENT column-name (#PCDATA)>
```

Global Transactions in the Oracle9*i* Application Server Environment

In a normal global transaction, you open connections to each database that you want included in the global transaction. After the transaction completes, the transaction manager commits all changes to all databases involved in the transaction.

However, with Oracle9*i* Application Server, the Oracle Database Cache may incorrectly be treated by the transaction manager as one of the databases in the global transaction. The EJB that is active in the middle-tier retrieves a connection to both the Oracle Database Cache and to the back-end Oracle database. However, the EJB can only update the back-end database. The transaction manager must not treat the Cache as another database involved in the transaction or it will perform a two-phase commit when the transaction ends. The two-phase commit process is expensive and unnecessary. Only the back-end database should be enlisted in the global transaction; thus, prompting the transaction manager to perform a single-phase commit, which is inexpensive.

To ensure that the Cache is not treated as an enlisted database in the global transaction, do the following:

1. Ensure that the `<default-enlist>` element in the Oracle-specific deployment descriptor is either not set -- so that it defaults to FALSE -- or is set to FALSE.
2. Bind the `DataSource` for the database Cache with any non-JTA type. Only a JTA `DataSource` object can be automatically enlisted in a global transaction.