Borland AppServer™ 6.7

DTDs

Borland Software Corporation
20450 Stevens Creek Blvd., Suite 800
Cupertino, CA 95014 USA
www.borland.com

Refer to the file deploy.html for a complete list of files that you can distribute in accordance with the License Statement and Limited Warranty.

BAS67DTD
December 2006

Borland®

Contents

1

Introduction to Borland AppServer

Borland AppServer (AppServer) is a set of services and tools that enable you to build, deploy, and manage distributed enterprise applications in your corporate environment.

The AppServer is a leading implementation of the J2EE 1.4 standard, and supports the latest industry standards such as EJB 2.1, JMS 1.1, Servlet 2.4, JSP 2.0, CORBA 2.6, XML, and SOAP. Borland provides two versions of AppServer, which include leading enterprise messaging solutions for Java Messaging Service (JMS) management (Tibco and OpenJMS). You can choose the degree of functionality and services you need in AppServer, and if your needs change, it is simple to upgrade your license.

The AppServer allows you to securely deploy and manage all aspects of your distributed Java and CORBA applications that implement the J2EE 1.4 platform standard.

With AppServer, the number of server instances per installation is unlimited, so the maximum of concurrent users is unlimited.

AppServer includes:

- Implementation of J2EE 1.4.
- Apache Web Server version 2.2.3
- Borland Security, which provides a framework for securing AppServer.
- Single-point management of leading JMS management solutions included with AppServer (Tibco, and OpenJMS).
- Strong management tools for distributed components, including applications developed outside of AppServer.

AppServer features

AppServer offers the following features:

- Support for BAS platforms (please refer to http://support.borland.com/kbcategory.jspa?categoryID=389 for a list of the platforms supported for AppServer).
- Full support for clustered topologies.
- Seamless integration with the VisiBroker ORB infrastructure.
- Integration with the Borland JBuilder integrated development environment.
- Enhanced integration with other Borland products including Borland Optimizeit Profiler and ServerTrace.
- AppServer allows existing applications to be exposed as Web Services and integrated with new applications or additional Web Services. Borland Web Services support is based on Apache Axis 1.2 technology, the next-generation Apache SOAP server that supports SOAP 1.2.

Borland AppServer Documentation

The AppServer documentation set includes the following:

- *Borland AppServer Installation Guide*—describes how to install AppServer on your network. It is written for system administrators who are familiar with Windows or UNIX operating systems.
- *Borland AppServer Developer's Guide*—provides detailed information about packaging, deployment, and management of distributed object-based applications in their operational environment.
- *Borland Management Console User's Guide*—provides information about using the Borland Management Console GUI.
- *Borland Security Guide*—describes Borland's framework for securing AppServer, including VisiSecure for VisiBroker for Java and VisiBroker for C++.
- Borland *VisiBroker for Java Developer's Guide*—describes how to develop VisiBroker applications in Java. It familiarizes you with configuration and management of the Visibroker ORB and how to use the programming tools. Also described is the IDL compiler, the Smart Agent, the Location, Naming and Event Services, the Object Activation Daemon (OAD), the Quality of Service (QoS), and the Interface Repository.
- Borland *VisiBroker VisiTransact Guide*—describes Borland's implementation of the OMG Object Transaction Service specification and the Borland Integrated Transaction Service components.

The documentation is typically accessed through the Help Viewer installed with your AppServer product. You can choose to view help from the standalone Help Viewer or from within a AppServer GUI tool. Both methods launch the Help Viewer in a separate window and give you access to the main Help Viewer toolbar for navigation and printing, as well as access to a navigation pane. The Help Viewer navigation pane includes a table of contents for all AppServer books and reference documentation, a thorough index, and a comprehensive search page.

The PDF books, Borland AppServer Developer's Guide and Borland Management Console User's Guide are available online at http://info.borland.com/techpubs/appserver.

Accessing AppServer online help topics

To access the online help, use one of the following methods:

Windows

Choose Start|Programs|Borland Deployment Platform|Help Topics

or, launch the Web browser and open <AppServer_Home>/doc/index.html.

UNIX

Launch a Web browser and open <AppServer_Home>/doc/index.html.

Accessing AppServer online help topics from within a AppServer GUI tool

To access the online help from within a AppServer GUI tool, use one of the following methods:

- From within the Borland Management Console, choose Help|Help Topics
- From within the Borland Deployment Descriptor Editor (DDEditor), choose Help|Help Topics

Documentation conventions

The documentation for AppServer uses the typefaces and symbols described below to indicate special text:

Convention	Used for
italics	Used for new terms and book titles.
`computer`	Information that the user or application provides, sample command lines and code.
`bold computer`	In text, bold indicates information the user types in. In code samples, bold highlights important statements.
[ ]	Optional items.
...	Previous argument that can be repeated.
|	Two mutually exclusive choices.

Platform conventions

The AppServer documentation uses the following symbols to indicate platform-specific information:

Symbol	Indicates
Windows	All supported Windows platforms.
Win2003	Windows 2003 only
WinXP	Windows XP only
Win2000	Windows 2000 only
UNIX	UNIX platforms
Solaris	Solaris only

Contacting Borland support

Borland offers a variety of support options. These include free services on the Internet where you can search our extensive information base and connect with other users of Borland products. In addition, you can choose from several categories of telephone support, ranging from support on installation of Borland products to fee-based, consultant-level support and detailed assistance.

For more information about Borland's support services or contacting Borland Technical Support, please see our web site at http://support.borland.com and select your geographic region.

When contacting Borland's support, be prepared to provide the following information:

- Name
- Company and site ID
- Telephone number
- Your Access ID number (U.S.A. only)
- Operating system and version
- Borland product name and version
- Any patches or service packs applied
- Client language and version (if applicable)
- Database and version (if applicable)
- Detailed description and history of the problem
- Any log files which indicate the problem
- Details of any error messages or exceptions raised

Online resources

You can get information from any of these online sources:

World Wide Web: http://www.borland.com

Online Support: http://support.borland.com (access ID required)

World Wide Web

Check http://www.borland.com regularly. The AppServer Product Team posts white papers, competitive analyses, answers to FAQs, sample applications, updated software, updated documentation, and information about new and existing products.

You may want to check these URLs in particular:

- http://www.borland.com/downloads/download_appserver.html (AppServer software and other files)
- http://support.borland.com (AppServer FAQs)

Borland newsgroups

You can participate in many threaded discussion groups devoted to the AppServer. Visit http://www.borland.com/newsgroups for information about joining user-supported newsgroups for Enterprise Server and other Borland products.

Note

These newsgroups are maintained by users and are not official Borland sites.

2

Application-client-borland.xml

DTD

```
<!ELEMENT application-client (ejb-ref*, resource-ref*,
            resource-env-ref*, property*)>
<!ELEMENT ejb-ref (ejb-ref-name, jndi-name?)>
<!ELEMENT resource-ref (res-ref-name, jndi-name)>
<!ELEMENT resource-env-ref (resource-env-ref-name, jndi-name)>
<!ELEMENT property (prop-name, prop-type, prop-value)>
<!ELEMENT prop-name (#PCDATA)>
<!ELEMENT prop-type (#PCDATA)>
<!ELEMENT prop-value (#PCDATA)>
<!ELEMENT ejb-ref-name (#PCDATA)>
<!ELEMENT jndi-name (#PCDATA)>
<!ELEMENT res-ref-name (#PCDATA)>
<!ELEMENT resource-env-ref-name (#PCDATA)>
```

<application-client> element

```
<!ELEMENT application-client (ejb-ref*, resource-ref*,
    resource-env-ref*, property*)>
```

This element is the root node of `application-client-borland.xml` and defines the necessary runtime information for an application client running in the VisiClient container. This node contains one or more `ejb-ref`, `resource-ref`, `resource-env-ref`, and `property` sub-elements.

Example

```
<!DOCTYPE application-client PUBLIC "-//Borland Corporation//DTD J2EE
Application Client 1.3//EN"
 "http://www.borland.com/devsupport/appserver/dtds/application-client_1_3-
borland.dtd">
<application-client>
          <ejb-ref>
            <ejb-ref-name>ejb/Sort</ejb-ref-name>
```

```
            <jndi-name>sort</jndi-name>
          </ejb-ref>
          <resource-ref>
            <res-ref-name>jdbc/CheckingDataSource</res-ref-name>
            <jndi-name>file:///net/machine/datasources/OracleDataSource</jndi-
name>
          </resource-ref>
</application-client>
```

Related Elements

- "<ejb-ref> element"
- "<resource-ref> element"
- "<resource-env-ref> element"
- "<property> element"

<ejb-ref> element

```
<!ELEMENT ejb-ref (ejb-ref-name, jndi-name?)>
```

This element is used to define EJB references used by the client. Each EJB reference contains an `ejb-ref-name` used by the client application and its associated `jndi-name` (if applicable).

Example

```
<ejb-ref>
    <ejb-ref-name>ejb/Sort</ejb-ref-name>
    <jndi-name>sort</jndi-name>
</ejb-ref>
```

Related Elements

- "<ejb-ref-name> element"
- "<jndi-name> element"

<ejb-ref-name> element

```
<!ELEMENT ejb-ref-name (#PCDATA)>
```

This element provides the name of an EJB used as a resource reference by the client application.

Example

```
<ejb-ref-name>ejb/Sort</ejb-ref-name>
```

Related Elements

- "<ejb-ref> element"

<jndi-name> element

```
<!ELEMENT jndi-name (#PCDATA)>
```

This element provides the JNDI service lookup for a resource referenced by the client application.

Example

```
<jndi-name>sort</jndi-name>
```

Related Elements

- "<ejb-ref> element"
- "<resource-ref> element"
- "<resource-env-ref> element"

<property> element

```
<!ELEMENT property (prop-name, prop-type, prop-value)>
```

This element is used to specify property values necessary for the client at runtime. Each `property` entry specifies the property's name, type, and value using the appropriate sub-elements.

Example

```
<property>
    <prop-name>vbroker.security.disable</prop-name>
    <prop-type>security</prop-type>
    <prop-value>false</prop-value>
</property>
```

Related Elements

- "<prop-name> element"
- <prop-type> element
- "<prop-value> element"

<prop-name> element

```
<!ELEMENT prop-name (#PCDATA)>
```

Specifies the name of the property to be set.

Example

```
<prop-name>vbroker.security.disable</prop-name>
```

Related Elements

- "<property> element"

<prop-type> element

```
<!ELEMENT prop-type (#PCDATA)>
```

Specifies the type of the property to be set.

Example

```
<prop-type>security</prop-type>
```

Related Elements

– “<property> element”

<prop-value> element

```
<!ELEMENT prop-value (#PCDATA)>
```

Specifies the value of the property to be set.

Example

```
<prop-value>false</prop-value>
```

Related Elements

– “<property> element”

<resource-env-ref-name> element

```
<!ELEMENT resource-env-ref (res-env-ref-name, jndi-name?)>
```

This element provides the name the client application uses to access a resource environment reference.

Example

```
<res-env-ref-name>jdbc/CheckingDataSource</res-ref-name>
```

Related Elements

– “<resource-env-ref> element”

<res-ref-name> element

```
<!ELEMENT res-ref-name (#PCDATA)>
```

This element provides the name the client application uses to access a resource reference.

Example

```
<res-ref-name>jdbc/CheckingDataSource</res-ref-name>
```

Related Elements

- "<resource-ref> element"

<resource-env-ref> element

```
<!ELEMENT resource-env-ref (res-env-ref-name, jndi-name?)>
```

This element is used to define resource environment references used by the client. Each resource environment reference contains a `res-env-ref-name` used by the client application and its associated `jndi-name` (if applicable).

Example

```
<resource-env-ref>
    <res-env-ref-name>jdbc/CheckingDataSource</res-ref-name>
    <jndi-name>file:///net/machine/datasources/OracleDataSource</jndi-name>
</resource-env-ref>
```

Related Elements

- "<resource-env-ref-name> element"
- "<jndi-name> element"

<resource-ref> element

```
<!ELEMENT resource-ref (res-ref-name, jndi-name?)>
```

This element is used to define resource references used by the client. Each resource reference contains an `res-ref-name` used by the client application and its associated `jndi-name` (if applicable).

Example

```
<resource-ref>
    <res-ref-name>jdbc/CheckingDataSource</res-ref-name>
    <jndi-name>file:///net/machine/datasources/OracleDataSource</jndi-name>
</resource-ref>
```

Related Elements

- "<res-ref-name> element"
- "<jndi-name> element"

3

ejb-borland.xml

DTD

```
 <!ELEMENT ejb-jar (enterprise-beans, datasource-definitions?,
 table-properties*, relationships?, authorization-domain?,
 property*, assembly-descriptor?)>

<!ELEMENT enterprise-beans (session | entity | message-driven)+>

<!ELEMENT session (ejb-name, bean-home-name?, bean-local-home-name?,
 timeout?, ejb-ref*, ejb-local-ref*, resource-ref*,resource-env-ref*,
property*)>

<!ELEMENT entity (ejb-name, bean-home-name?, bean-local-home-name?,
 ejb-ref*, ejb-local-ref*, resource-ref*, resource-env-ref*,
 (cmp-info | cmp2-info)?, property*)>

<!ELEMENT message-driven (ejb-name, message-driven-destination-name,
 connection-factory-name, pool?, ejb-ref*, ejb-local-ref*,
 resource-ref*, resource-env-ref*, property*)>

 <!ELEMENT pool (max-size?, init-size?, wait-timeout?)>
 <!ELEMENT ejb-ref (ejb-ref-name, jndi-name?)>
 <!ELEMENT ejb-local-ref (ejb-ref-name, jndi-name?)>
 <!ELEMENT resource-ref (res-ref-name, jndi-name, cmp-resource?)>
 <!ELEMENT resource-env-ref (resource-env-ref-name, jndi-name)>
 <!ELEMENT datasource-definitions (datasource*)>
 <!ELEMENT datasource (jndi-name, url, username?, password?,
  isolation-level?, driver-class-name?, jdbc-property*, property*)>
 <!ELEMENT jdbc-property (prop-name, prop-value)>
 <!ELEMENT property (prop-name, prop-type, prop-value)>
 <!ELEMENT cmp-info (description?, database-map?, finder*)>
 <!ELEMENT database-map (table?, column-map*)>
 <!ELEMENT finder (method-signature, where-clause, load-state?)>
 <!ELEMENT column-map (field-name, column-name?, column-type?, ejb-ref-name?)>
 <!ELEMENT connection-factory-name (#PCDATA)>
 <!ELEMENT message-driven-destination-name (#PCDATA)>
```

```
<!ELEMENT max-size (#PCDATA)>
<!ELEMENT init-size (#PCDATA)>
<!ELEMENT wait-timeout (#PCDATA)>
<!ELEMENT cmp-resource (#PCDATA)>
<!ELEMENT method-signature (#PCDATA)>
<!ELEMENT where-clause (#PCDATA)>
<!ELEMENT load-state (#PCDATA)>
<!ELEMENT prop-name (#PCDATA)>
<!ELEMENT prop-type (#PCDATA)>
<!ELEMENT prop-value (#PCDATA)>
<!ELEMENT field-name (#PCDATA)>
<!ELEMENT column-name (#PCDATA)>
<!ELEMENT column-type (#PCDATA)>
<!ELEMENT table (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT ejb-name (#PCDATA)>
<!ELEMENT bean-home-name (#PCDATA)>
<!ELEMENT bean-local-home-name (#PCDATA)>
<!ELEMENT timeout (#PCDATA)>
<!ELEMENT ejb-ref-name (#PCDATA)>
<!ELEMENT jndi-name (#PCDATA)>
<!ELEMENT res-ref-name (#PCDATA)>
<!ELEMENT resource-env-ref-name (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT username (#PCDATA)>
<!ELEMENT password (#PCDATA)>
<!ELEMENT isolation-level (#PCDATA)>
<!ELEMENT driver-class-name (#PCDATA)>
<!ELEMENT authorization-domain (#PCDATA)>
<!ELEMENT table-properties (table-name, column-properties*, property*)>
<!ELEMENT column-properties (column-name, property*)>
<!ELEMENT table-name (#PCDATA)>
<!ELEMENT cmp2-info (cmp-field*, table-name, table-ref*)>
<!ELEMENT relationships (ejb-relation+)>
<!ELEMENT ejb-relation (ejb-relationship-role, ejb-relationship-role)>
<!ELEMENT ejb-relationship-role (relationship-role-source, cmr-field?)>
<!ELEMENT relationship-role-source (ejb-name)>
<!ELEMENT cmr-field (cmr-field-name, table-ref, property*)>
<!ELEMENT cmr-field-name (#PCDATA)>
<!ELEMENT table-ref (left-table, cross-table*, right-table)>
<!ELEMENT left-table (table-name, column-list)>
<!ELEMENT right-table (table-name, column-list)>
<!ELEMENT cross-table (table-name, column-list, column-list)>
<!ELEMENT column-list (column-name+)>
<!ELEMENT cmp-field (field-name, (cmp-field-map* | column-name),property*)>
<!ELEMENT cmp-field-map (field-name, column-name)>
<!ELEMENT assembly-descriptor (security-role*)>
<!ELEMENT security-role (role-name, deployment-role?)>
<!ELEMENT role-name (#PCDATA)>
<!ELEMENT deployment-role (#PCDATA)>
```

<assembly-descriptor>

```
<!ELEMENT assembly-descriptor (security-role*)>
```

This element builds upon the same element in `ejb-jar.xml`. Using its child nodes, you provide information on one or more security roles to which modules in the archive belong.

Example

```
<assembly-descriptor>
    <security-role>
        <role-name>administrator</role-name>
        <deployment-role>administrator</deployment-role>
    </security-role>
</assembly-descriptor>
```

Related Elements

- "<ejb-jar>"
- "<security-role>"

<authorization-domain>

```
<!ELEMENT authorization-domain (#PCDATA)>
```

The name of the authorization domain to which the archive belongs.

Example

```
<authorization-domain>GroupJ</authorization-domain>
```

Related Elements

- "<ejb-jar>"

<bean-home-name>

```
<!ELEMENT bean-home-name (#PCDATA)>
```

Specifies the name used to look-up the bean's home interface.

Example

```
<bean-home-name>insurance/remote/clerk</bean-home-name>
```

Related Elements

- "<session>"
- "<entity>"
- "<message-driven>"

<bean-local-home-name>

```
<!ELEMENT bean-local-home-name (#PCDATA)>
```

Specifies the name used to look-up the bean's local-home interface.

Example

```
<bean-local-home-name>insurance/remote/clerk</bean-local-home-name>
```

Related Elements

- "<session>"
- "<entity>"
- "<message-driven>"

<cascade-delete>

```
<!ELEMENT cascade-delete />
```

Use <cascade-delete> when you want to remove entity bean objects. When cascade delete is specified for an object, the container automatically deletes all of that object's dependent objects.

Example

```
<ejb-relation>
 <ejb-relation-name>Customer-Account</ejb-relation-name>
 <ejb-relationship-role>
  <ejb-relationship-role-name>Account-Has-Customer
  </ejb-relationship-role-name>
  <multiplicity>one</multiplicity>
  <cascade-delete/>
 </ejb-relationship-role>
</ejb-relation>
```

Related Elements

- "<ejb-relationship-role>"

<cmp-field>

```
<!ELEMENT cmp-field (field-name, (cmp-field-map* | column-name),property*)>
```

Basic field mapping is accomplished using the <cmp-field> element. In this element's child nodes, you specify a field name and a corresponding column to which it maps. Many users may employ coarse-grained entity beans that implement a Java class to represent more fine-grained data. A third child node, <cmp-field-map>, defines a field map between your fine-grained class and its underlying database representation, and can be used instead of the <column-name> element.

Example

```
<cmp-field>
  <field-name>orderNumber</field-name>
  <column-name>ORDER_NUMBER</column-name>
</cmp-field>
```

Related Elements

- "<field-name>"
- "<column-name>"
- "<cmp-field-map>"

<cmp-field-map>

```
<!ELEMENT cmp-field-map (field-name, column-name)>
```

The <cmp-field-map> element defines a field map between a fine-grained Java class and its underlying database representation. Note that such classes must implement java.io.Serializable and all their data members must be public.

Example

```
<cmp-field>
  <field-name>Address</field-name>
  <cmp-field-map>
   <field-name>Address.AddressLine</field-name>
   <column-name>STREET</column-name>
  </cmp-field-map>
  <cmp-field-map>
  ...
 </cmp-field>
```

Related Elements

- "<cmp-field>"
- "<field-name>"
- "<column-name>"

<cmp-info>

```
<!ELEMENT cmp-info (description?, database-map?, finder*)>
```

Use the <cmp-info> element to provide information about CMP 1.x entity beans. (If you are using CMP 2.x, use "<cmp2-info>" .) It has two child nodes, <database-map> and <finder>, which specify the necessary data to access the bean's backing store and use the appropriate query.

Example

```
<cmp-info>
           <description/>
           <database-map>
               <table>Courses</table>
               <column-map>
                   <field-name>students</field-name>
                   <ejb-ref-name>ejb/Student.findByCourse</ejb-ref-name>
               </column-map>
           </database-map>
           <finder>
               <method-signature>findByStudent(Student s)</method-signature>
               <where-clause>SELECT course_dept, course_number FROM
  Enrollment WHERE student  = :s[ejb/Student]</where-clause>
```

```
            <load-state>False</load-state>
        </finder>
</cmp-info>
```

Related Elements

- "<database-map>"
- "<finder>"

<cmp-resource>

```
<!ELEMENT cmp-resource (#PCDATA)>
```

If the referred resource uses container-managed-persistence, set this flag to `True`. This element is valid for CMP 1.x resources only. If this flag is set, BES will ignore the corresponding resource reference in the standard `ejb-jar.xml` deployment descriptor.

Example

```
<cmp-resource>True</cmp-resource>
```

Related Elements

- "<resource-ref> element"

<cmp2-info>

```
<!ELEMENT cmp2-info (cmp-field*, table-name, table-ref*)>
```

If you are using CMP 2.0, you use the `<cmp2-info>` element to provide information to the container to manage your entity beans. This element and its child nodes contain all the data necessary to map your CMP fields to database columns.

Example

```
<cmp2-info>
 <cmp-field>
  <field-name>orderNumber</field-name>
  <column-name>ORDER_NUMBER</column-name>
 </cmp-field>
 <cmp-field>
  <field-name>line</field-name>
  <column-name>LINE</column-name>
 </cmp-field>
</cmp2-info>
```

Related Elements

- "<entity>"
- "<cmp-field>"
- "<table-name>"
- "<table-ref>"

<cmr-field>

```
<!ELEMENT cmr-field (cmr-field-name, table-ref, property*)>
```

Defines the fields used by one entity to map to another, and the underlying table mapping accompanying it.

Example

```
<cmr-field>
    <cmr-field-name>specialInformation</cmr-field-name>
    <table-ref>
     <left-table>
      <table-name>CUSTOMER</table-name>
      <column-list>CUSTOMER_NO</column-list>
     </left-table>
     <right-table>
      <table-name>SPECIAL_INFO</table-name>
      <column-list>CUSTOMER_NO</column-list>
     </right-table>
    </table-ref>
</cmr-field>
```

Related Elements

- "<ejb-relationship-role>"
- "<cmr-field-name>"
- "<table-ref>"
- "<property> element"

<cmr-field-name>

```
<!ELEMENT cmr-field-name (#PCDATA)>
```

The field used by the entity to map to another entity with which it has a relationship.

Example

```
<cmr-field-name>specialInformation</cmr-field-name>
```

Related Elements

- "<cmr-field>"

<column-list>

```
<!ELEMENT column-list (column-name+)>
```

Specifies the columns in one table that map to columns in another table. Each column is delineated by the child-node `<column-name>`.

Example

```
<column-list>
    <column-name>EMP_NO</column-name>
    <column-name>LAST_NAME</column-name>
    <column-name>PROJ_ID</column-name>
</column-list>
```

Related Elements

- "<table-ref>"
- "<left-table>"
- "<cross-table>"
- "<table-name>"
- "<column-name>"

<column-map>

```
<!ELEMENT column-map (field-name, column-name?, column-type?, ejb-ref-name?)>
```

This element is used to provide information to the CMP 1.x engine about the database columns used by the entity bean. You provide the field name of the entity bean and either information on its corresponding column or the name of an EJB reference representing the column.

Example

```
<column-map>
     <field-name>students</field-name>
     <ejb-ref-name>ejb/Student.findByCourse</ejb-ref-name>
</column-map>
```

Related Elements

- "<table>"
- "<database-map>"
- "<field-name>"
- "<column-name>"
- "<column-type>"
- "<ejb-ref-name> element"

<column-name>

```
<!ELEMENT column-name (#PCDATA)>
```

Specifies the name of the database column for entity mapping or property setting.

Example

```
<column-name>course_dept</column-name>
```

Related Elements

- "<database-map>"

- “<field-name>”
- “<column-type>”
- “<ejb-ref-name> element”
- “<cmp-field-map>”
- “<cmp-field>”
- “<column-properties>”

<column-properties>

```
<!ELEMENT column-properties (column-name, property*)>
```

Use this element to specify properties particular to a column in a database table. You provide the column name and the associated property in this element's child nodes.

Related Elements

- “<table-properties>”
- “<column-name>”
- “<property> element”

<column-type>

```
<!ELEMENT column-type (#PCDATA)>
```

Specifies the type of data stored in a database column that is part of the entity-bean's CMP queries.

Example

```
<column-type>integer</column-type>
```

Related Elements

- “<database-map>”
- “<field-name>”
- “<column-name>”
- “<column-type>”
- “<ejb-ref-name> element”

<connection-factory-name>

```
<!ELEMENT connection-factory-name (#PCDATA)>
```

The JNDI name of the JMS topic or queue connection factory which the bean uses to connect to the JMS service.

Example

```
<connection-factory-name>serial://jms/xaqcf</connection-factory-name>
```

Related Elements

- “<message-driven>”
- “<message-driven-destination-name>”

<cross-table>

```
<!ELEMENT cross-table (table-name, column-list, column-list)>
```

If you define a many-to-many relationship, you must also have the CMP engine create a cross-table which models a relationship between the left table and the right table. Do this using the <cross-table> element. You may name this cross-table whatever you like using the child-node <table-name>. The two child <column-list> elements correspond to columns in the left and right tables whose relationship you wish to model.

Example

```
<table-ref>
 <left-table>
  <table-name>EMPLOYEE</table-name>
  <column-list>
   <column-name>EMP_NO</column-name>
            <column-name>LAST_NAME</column-name>
            <column-name>PROJ_ID</column-name>
  </column-list>
 </left-table>
 <cross-table>
  <table-name>EMPLOYEE_PROJECTS</table-name>
  <column-list>
   <column-name>EMP_NAME</column-name>
         <column-name>PROJ_ID</column-name>
  </column-list>
  <column-list>
   <column-name>PROJ_ID</column-name>
           <column-name>PROJ_NAME</column-name>
  </column-list>
 </cross-table>
 <right-table>
  <table-name>PROJECT</table-name>
  <column-list>
   <column-name>PROJ_ID</column-name>
           <column-name>PROJ_NAME</column-name>
            <column-name>EMP_NO</column-name>
  </column-list>
 </right-table>
</table-ref>
```

Related Elements

- “<table-ref>”
- “<left-table>”
- “<right-table>”
- “<table-name>”
- “<column-list>”
- “<column-name>”

<database-map>

```
<!ELEMENT database-map (table?, column-map*)>
```

The <database-map> element is used to provide datasource information the CMP 1.x engine requires to populate an entity bean's fields and execute its finder methods. Its

child nodes specify the database table to use, as well as the fields and columns used by the entity bean to populate its fields.

Example

```
<database-map>
       <table>Courses</table>
       <column-map>
          <field-name>students</field-name>
          <ejb-ref-name>ejb/Student.findByCourse</ejb-ref-name>
       </column-map>
</database-map>
```

Related Elements

- "<table>"
- "<column-map>"

<datasource-definitions>

```
<!ELEMENT datasource-definitions (datasource*)>
```

If you are using old-style JDBC 1.x datasources, this element is used to provide information on the datasources used by the beans in the archive. Each datasource is defined within its own `<datasource>` element, which are child nodes of `<datasource-definitions>`. Most users will use the new-style `jndi-definitions.xml` file to define their datasources. This element is only for JDBC 1.x users.

Related Elements

- "<datasource>"

<datasource>

```
<!ELEMENT datasource (jndi-name, url, username?, password?,
  isolation-level?, driver-class-name?, jdbc-property*, property*)>
```

This element is used to describe a JDBC 1.x datasource. Most users will want to use the new-style `jndi-definitions.xml` file to define their datasources. This element and its child nodes apply only to JDBC 1.x.

Related Elements

- "<datasource-definitions>"
- "<jndi-name> element"
- "<url>"
- "<username>"
- "<password>"
- "<isolation-level>"
- "<driver-class-name>"
- "<jdbc-property>"
- "<property> element"

<deployment-role>

```
<!ELEMENT deployment-role (#PCDATA)>
```

The role name for a deployment role used by modules in the archive.

Example

```
<deployment-role>administrator</deployment-role>
```

Related Elements

– “<security-role>”

<description>

```
<!ELEMENT description (#PCDATA)>
```

Use this optional element to provide a description of its parent node.

Example

```
<description>sorting bean</description>
```

<driver-class-name>

```
<!ELEMENT driver-class-name (#PCDATA)>
```

JDBC 1.x only. The class name of the driver used to access the datasource being defined.

Related Elements

– “<datasource>”

<ejb-jar>

```
<!ELEMENT ejb-jar (enterprise-beans, datasource-definitions?,
 table-properties*, relationships?, authorization-domain?,
 property*, assembly-descriptor?)>
```

The <ejb-jar> element is the root node of ejb-borland.xml. Its child nodes define the enterprise beans deployed with the JAR and provides information on the relationships between them. You can also specify information about datasources (such as databases and JMS providers) used by the beans, as well as archive properties and security-related information.

Example

```
<ejb-jar>
    <enterprise-beans>
        <session>
            <ejb-name>AsyncSenderEJB</ejb-name>
            <bean-local-home-name>ejb/local/petstore/asyncsender/AsyncSender</
```

```
bean-local-home-name>
            <timeout>0</timeout>
            <resource-ref>
                <res-ref-name>jms/queue/QueueConnectionFactory</res-ref-name>
                <jndi-name>serial://jms/xaqcf</jndi-name>
            </resource-ref>
            <resource-env-ref>
                <resource-env-ref-name>jms/queue/AsyncSenderQueue</resource-
env-ref-name>
                <jndi-name>serial://jms/queue/opc/OrderQueue</jndi-name>
            </resource-env-ref>
        </session>
    </enterprise-beans>
    <assembly-descriptor/>
</ejb-jar>
```

Related Elements

- "<enterprise-beans>"
- "<datasource-definitions>"
- "<table-properties>"
- "<relationships>"
- "<authorization-domain>"
- "<property> element"
- "<assembly-descriptor>"

<ejb-name>

```
<!ELEMENT ejb-name (#PCDATA)>
```

Use the `<ejb-name>` element to provide a name for the enterprise javabean you are defining. This element is analagous to the same element in `ejb-jar.xml`, providing a name used to look-up the bean remotely.

Example

```
<ejb-name>clerk</ejb-name>
```

Related Elements

- "<session>"
- "<entity>"
- "<message-driven>"
- "<relationship-role-source>"

<ejb-ref> element

```
<!ELEMENT ejb-ref (ejb-ref-name, jndi-name?)>
```

This element is used to define EJB references used by the bean. Each EJB reference contains an `ejb-ref-name` used by the client application and its associated `jndi-name` (if applicable).

Example

```
<ejb-ref>
    <ejb-ref-name>ejb/Sort</ejb-ref-name>
    <jndi-name>sort</jndi-name>
</ejb-ref>
```

Related Elements

- "<ejb-ref-name> element"
- "<jndi-name> element"

<ejb-ref-name> element

```
<!ELEMENT ejb-ref-name (#PCDATA)>
```

This element provides the name of an EJB used as a resource reference by the bean.

Example

```
<ejb-ref-name>ejb/Sort</ejb-ref-name>
```

Related Elements

- "<ejb-ref> element"
- "<column-map>"

<ejb-relation>

```
<!ELEMENT ejb-relation (ejb-relationship-role, ejb-relationship-role)>
```

This element delineates the relationship between two entities. Each entity's relationship to the other is defined separately using the child node `<ejb-relationship-role>`, even in the case of a uni-directional relationship.

Example

Since these relationships can come in several different forms, the following examples are provided:

- uni-directional one-to-one relationship
- bi-directional one-to-many relationship

uni-directional one-to-one relationship

```
<relationships>
 <ejb-relation>
  <ejb-relationship-role>
   <relationship-role-source>
    <ejb-name>Customer</ejb-name>
   </relationship-role-source>
   <cmr-field>
    <cmr-field-name>specialInformation</cmr-field-name>
    <table-ref>
     <left-table>
```

```
      <table-name>CUSTOMER</table-name>
      <column-list>CUSTOMER_NO</column-list>
     </left-table>
     <right-table>
      <table-name>SPECIAL_INFO</table-name>
      <column-list>CUSTOMER_NO</column-list>
     </right-table>
    </table-ref>
   </cmr-field>
  </ejb-relationship-role>
  <ejb-relationship-role>
   <relationship-role-source>
    <ejb-name>SpecialInfo</ejb-name>
   </relationship-role-source>
  </ejb-relationship-role>
 </ejb-relation>
</relationships>
```

Since the relationship is uni-directional, no table information needs to be specified for the `SpecialInfo` bean.

bi-directional one-to-many relationship

```
<relationships>
 <ejb-relation>
  <ejb-relationship-role>
   <relationship-role-source>
    <ejb-name>Customer</ejb-name>
   </relationship-role-source>
   <cmr-field>
    <cmr-field-name>orders</cmr-field-name>
        <table-ref>
            <left-table>
                <table-name>CUSTOMER</table-name>
                <column-list>
                    <column-name>CUSTOMER_NO</column-name>
                </column-list>
            </left-table>
            <right-table>
                <table-name>ORDER</table-name>
                <column-list>
                    <column-name>CUSTOMER_NO</column-name>
                </column-list>
            </right-table>
        </table-ref>
   </cmr-field>
  </ejb-relationship-role>
  <ejb-relationship-role>
   <relationship-role-source>
    <ejb-name>Customer</ejb-name>
   </relationship-role-source>
   <cmr-field>
    <cmr-field-name>customers</cmr-field-name>
        <table-ref>
            <left-table>
                <table-name>ORDER</table-name>
                <column-list>
                    <column-name>CUSTOMER_NO</column-name>
                </column-list>
            </left-table>
```

```
            <right-table>
                <table-name>CUSTOMER</table-name>
                <column-list>
                    <column-name>CUSTOMER_NO</column-name>
                </column-list>
            </right-table>
        </table-ref>
    </cmr-field>
   </ejb-relationship-role>
  </ejb-relation>
  .
  .
</relationships>
```

Since the tables are linked in both directions, table data is provided for each direction.

Related Elements

- "<relationships>"
- "<ejb-relationship-role>"

<ejb-relationship-role>

```
<!ELEMENT ejb-relationship-role (relationship-role-source, cmr-field?)>
```

Defines a single entity and its relationship to another entity. The entity itself is provided using the child node `<relationship-role-source>`, while the fields it has in common with the other entity in the relationship are defined in the child node `<cmr-field>`.

Example

```
<ejb-relationship-role>
   <relationship-role-source>
    <ejb-name>Customer</ejb-name>
   </relationship-role-source>
   <cmr-field>
    <cmr-field-name>specialInformation</cmr-field-name>
    <table-ref>
     <left-table>
      <table-name>CUSTOMER</table-name>
      <column-list>CUSTOMER_NO</column-list>
     </left-table>
     <right-table>
      <table-name>SPECIAL_INFO</table-name>
      <column-list>CUSTOMER_NO</column-list>
     </right-table>
    </table-ref>
   </cmr-field>
</ejb-relationship-role>
```

Related Elements

- "<ejb-relation>"
- "<relationship-role-source>"
- "<cmr-field>"

<enterprise-beans>

```
<!ELEMENT enterprise-beans (session | entity | message-driven)+>
```

Use the <enterprise-beans> element to define the Java beans within the archive. The three different kinds of enterprise Java beans--session beans, entity beans, and message-driven beans--have corresponding child-nodes in which you provide information about them. You create an entry for each bean in the archive based on its type.

Example

```
<enterprise-beans>
        <session>
            <ejb-name>UniqueIdGeneratorEJB</ejb-name>
            ...
        </session>
        <entity>
            <ejb-name>CounterEJB</ejb-name>
            ...
        </entity>
</enterprise-beans>
```

Related Elements

- "<ejb-jar>"
- "<session>"
- "<entity>"
- "<message-driven>"

<entity>

```
<!ELEMENT entity (ejb-name, bean-home-name?, bean-local-home-name?,
 ejb-ref*, ejb-local-ref*, resource-ref*, resource-env-ref*,
 (cmp-info | cmp2-info)?, property*)>
```

The <entity> element is used to provide information about entity beans contained within the archive. Child-nodes of this element allow you to specify the bean's various interfaces and references. Container-Managed Persistence information can also be provided, as well as generic properties specific to the individual entity bean.

Example

```
<entity>
       <ejb-name>claim</ejb-name>
       <bean-local-home-name>Claim</bean-local-home-name>
       <cmp2-info>
           <cmp-field>
               <field-name>claimId</field-name>
               <column-name>CLAIM_ID</column-name>
           </cmp-field>
           <cmp-field>
               <field-name>policyHolderNumber</field-name>
               <column-name>POLICYHOLDER_NUMBER</column-name>
           ...
           <table-name>CLAIMS</table-name>
```

```
        </cmp2-info>
</entity>
```

Related Elements

- “<enterprise-beans>”
- “<ejb-name>”
- “<bean-home-name>”
- “<bean-local-home-name>”
- “<ejb-ref> element”
- “<resource-ref> element”
- “<resource-env-ref> element”
- “<cmp-info>”
- “<cmp2-info>”
- “<property> element”

<field-name>

```
<!ELEMENT field-name (#PCDATA)>
```

The name of an entity bean field that maps to a column in an underlying datasource.

Example

```
<field-name>students</field-name>
```

Related Elements

- “<database-map>”
- “<column-name>”
- “<column-type>”
- “<ejb-ref-name> element”
- “<cmp-field-map>”
- “<cmp-field>”

<finder>

```
<!ELEMENT finder (method-signature, where-clause, load-state?)>
```

Use this element to define the finders used by the entity bean. When you construct a finder method, you are actually constructing an SQL select statement with a where clause. The select statement includes a clause that states what records or data are to be found and returned. Under container-managed persistence, you must specify the terms of the where clause using the child nodes of `<finder>`.

Example

```
<finder>
    <method-signature>findByStudent(Student s)</method-signature>
    <where-clause>SELECT course_dept, course_number FROM
                  Enrollment WHERE student  = :s[ejb/Student]</where-clause>
    <load-state>False</load-state>
</finder>
```

Related Elements

- “<cmp-info>”
- “<method-signature>”
- “<where-clause>”
- “<load-state>”

<init-size>

```
<!ELEMENT init-size (#PCDATA)>
```

When the pool is initially created, this is the number of connections with which BES populates the pool. This is analagous to the property `ejb.mdb.init-size`.

Example

<pool> <max-size>0</max-size> <init-size>0</init-size> <wait-timeout>0</wait-timeout> </pool>

Related Elements

- “<message-driven>”
- “<pool>”
- “<max-size>”
- “<wait-timeout>”

<isolation-level>

```
<!ELEMENT isolation-level (#PCDATA)>
```

JDBC 1.x only. The isolation level for the datasource being defined. This can be one of the following values:

- TRANSACTION_NONE
- TRANSACTION_READ_COMMITTED
- TRANSACTION_READ_UNCOMMITTED
- TRANSACTION_REPREATABLE_READ
- TRANSACTION_SERIALIZEABLE

Related Elements

- “<datasource>”

<jdbc-property>

```
<!ELEMENT jdbc-property (prop-name, prop-value)>
```

JDBC 1.x only. Specifies a JDBC property to be used with the datasource being defined. You specify the property name and its value with the child nodes `<prop-name>` and `<prop-value>`, respectively.

Related Elements

- "<datasource>"
- "<prop-name> element"
- "<prop-value> element"

<jndi-name> element

```
<!ELEMENT jndi-name (#PCDATA)>
```

This element provides the JNDI service lookup for a resource referenced by the bean.

Example

```
<jndi-name>sort</jndi-name>
```

Related Elements

- "<ejb-ref> element"
- "<resource-ref> element"
- "<resource-env-ref> element"

<left-table>

```
<!ELEMENT left-table (table-name, column-list)>
```

When providing <cmp2-info>, this element defines one of two tables that share a column, one being a foriegn key of the other.

When using this element to describe <relationships>, the <left-table> is the source of that relationship, with the <right-table> being the destination. That is, the direction of the relationship proceeds from left-to-right. If you are defining a bi-directional relationship, you would create two <table-ref> elements for the same relationship, one for each direction. In many-to-many relationships, the <left-table> makes up one of two tables used to create a <cross-table> defining their intersections.

Example

```
<left-table>
     <table-name>CUSTOMER</table-name>
     <column-list>CUSTOMER_NO</column-list>
</left-table>
```

Related Elements

- "<table-ref>"
- "<right-table>"
- "<cross-table>"
- "<table-name>"
- "<column-list>"

<load-state>

```
<!ELEMENT load-state (#PCDATA)>
```

Set this flag to `true` if you want the container to load the current state of the Entity Bean upon calling its `ejbCreate()` method.

Example

```
<load-state>false</load-state>
```

Related Elements

- "<finder>"
- "<method-signature>"
- "<where-clause>"

<max-size>

```
<!ELEMENT max-size (#PCDATA)>
```

This is the maximum number of connections allowed in the in the connection pool. This is analagous to the property `ejb.mdb.max-size`.

Example

<pool> <max-size>0</max-size> <init-size>0</init-size> <wait-timeout>0</wait-timeout> </pool>

Related Elements

- "<message-driven>"
- "<pool>"
- "<init-size>"
- "<wait-timeout>"

<message-driven-destination-name>

```
<!ELEMENT message-driven-destination-name (#PCDATA)>
```

Specifies the JNDI name of the individual queue or topic to which the bean subscribes.

Example

```
 <message-driven-destination-name>serial://resources/tcf</message-driven-
destination-name>
```

Related Elements

- "<message-driven>"
- "<connection-factory-name>"

<message-driven>

```
<!ELEMENT message-driven (ejb-name, message-driven-destination-name,
 connection-factory-name, pool?, ejb-ref*, ejb-local-ref*,
 resource-ref*, resource-env-ref*, property*)>
```

Describes a message-driven bean deployed to the archive. Child-nodes of this element allow you to specify the bean's various interfaces and references. You can also provide data on the queue or topic to which the bean connects, as well as the connection factory it uses to do so. You can also specify how the bean behaves in the pool.

Example

```
<message-driven>
    <ejb-name>MsgBean</ejb-name>
    <message-driven-destination-name>serial://jms/queue/supplier/
PurchaseOrderQueue</message-driven-destination-name>
    <connection-factory-name>serial://jms/xaqcf</connection-factory-name>
    <pool>
          <max-size>0</max-size>
          <init-size>0</init-size>
          <wait-timeout>0</wait-timeout>
    </pool>
</message-driven>
```

Related Elements

- “<ejb-name>”
- “<message-driven-destination-name>”
- “<connection-factory-name>”
- “<pool>”
- “<ejb-ref> element”
- “<resource-ref> element”
- “<resource-env-ref> element”
- “<property> element”

<method-signature>

```
<!ELEMENT method-signature (#PCDATA)>
```

The method, as it appears in the entity bean, used to conduct a database query.

Example

```
<method-signature>findByStudent(Student s)</method-signature>
```

Related Elements

- “<finder>”
- “<where-clause>”
- “<load-state>”

<password>

```
<!ELEMENT password (#PCDATA)>
```

JDBC 1.x only. The password used to access the datasource being defined.

Related Elements

– "<datasource>"

<pool>

```
<!ELEMENT pool (max-size?, init-size?, wait-timeout?)>
```

Contains child nodes you to specify the resource pool properties for the MDB.

Example

```
<pool>
     <max-size>0</max-size>
     <init-size>0</init-size>
     <wait-timeout>0</wait-timeout>
</pool>
```

Related Elements

– "<message-driven>"
– "<max-size>"
– "<init-size>"
– "<wait-timeout>"

<prop-name> element

```
<!ELEMENT prop-name (#PCDATA)>
```

Specifies the name of the property to be set.

Example

```
<prop-name>vbroker.security.disable</prop-name>
```

Related Elements

– "<property> element"

<prop-type> element

```
<!ELEMENT prop-type (#PCDATA)>
```

Specifies the type of the property to be set.

Example

```
<prop-type>security</prop-type>
```

Related Elements

– “<property> element”

<prop-value> element

```
<!ELEMENT prop-value (#PCDATA)>
```

Specifies the value of the property to be set.

Example

```
<prop-value>false</prop-value>
```

Related Elements

– “<property> element”

<property> element

```
<!ELEMENT property (prop-name, prop-type, prop-value)>
```

This element is used to specify property values for various resources included in or referenced by the archive or its components. Each `property` entry specifies the property's name, type, and value using the appropriate sub-elements.

Example

```
<property>
    <prop-name>vbroker.security.disable</prop-name>
    <prop-type>security</prop-type>
    <prop-value>false</prop-value>
</property>
```

Related Elements

– “<prop-name> element”
– “<prop-type> element”
– “<prop-value> element”

<relationship-role-source>

```
<!ELEMENT relationship-role-source (ejb-name)>
```

Specifies the name of the entity bean involved in a container-managed relationship with another entity bean.

Example

```
<relationship-role-source>
    <ejb-name>Customer</ejb-name>
</relationship-role-source>
```

Related Elements

- "<ejb-relationship-role>"
- "<ejb-name>"

<relationships>

```
<!ELEMENT relationships (ejb-relation+)>
```

To specify relationships between tables, you use the <relationships> element. Within the <relationships> element, you define an <ejb-relationship-role> containing the role's source (an entity bean) and a <cmr-field> element containing the relationship. The descriptor then uses <table-ref> elements to specify relationships between two tables, a <left-table> and a <right-table>. You must observe the following cardinalities:

- One <ejb-relationship-role> must be defined per direction; if you have a bi-directional relationship, you must define an <ejb-relationship-role> for each bean with each referencing the other.
- Only one <table-ref> element is permitted per relationship.

If you define a many-to-many relationship, you must also have the CMP engine create a cross-table which models a relationship between the left table and the right table. This is performed using the <cross-table> element.

Example

```
<relationships>
 <ejb-relation>
  <ejb-relationship-role>
   <relationship-role-source>
    <ejb-name>Customer</ejb-name>
   </relationship-role-source>
   <cmr-field>
    <cmr-field-name>specialInformation</cmr-field-name>
    <table-ref>
     <left-table>
      <table-name>CUSTOMER</table-name>
      <column-list>CUSTOMER_NO</column-list>
     </left-table>
     <right-table>
      <table-name>SPECIAL_INFO</table-name>
      <column-list>CUSTOMER_NO</column-list>
     </right-table>
    </table-ref>
   </cmr-field>
  </ejb-relationship-role>
  <ejb-relationship-role>
   <relationship-role-source>
    <ejb-name>SpecialInfo</ejb-name>
   </relationship-role-source>
  </ejb-relationship-role>
```

```
 </ejb-relation>
</relationships>
```

Related Elements

- "<ejb-jar>"
- "<ejb-relation>"

<resource-env-ref-name> element

```
<!ELEMENT resource-env-ref (res-env-ref-name, jndi-name?)>
```

This element provides the name the bean uses to access a resource environment reference.

Example

```
<res-env-ref-name>jdbc/CheckingDataSource</res-ref-name>
```

Related Elements

- "<resource-env-ref> element"

<res-ref-name> element

```
<!ELEMENT res-ref-name (#PCDATA)>
```

This element provides the name the bean uses to access a resource reference.

Example

```
<res-ref-name>jdbc/CheckingDataSource</res-ref-name>
```

Related Elements

- "<resource-ref> element"

<resource-env-ref> element

```
<!ELEMENT resource-env-ref (res-env-ref-name, jndi-name?)>
```

This element is used to define resource environment references used by the bean. Each resource environment reference contains a `res-env-ref-name` used by the bean and its associated `jndi-name` (if applicable).

Example

```
<resource-env-ref>
    <res-env-ref-name>jdbc/CheckingDataSource</res-ref-name>
    <jndi-name>file:///net/machine/datasources/OracleDataSource</jndi-name>
</resource-env-ref>
```

Related Elements

- "<resource-env-ref-name> element"
- "<jndi-name> element"

<resource-ref> element

```
<!ELEMENT resource-ref (res-ref-name, jndi-name?)>
```

This element is used to define resource references used by the bean. Each resource reference contains an `res-ref-name` used by the client application and its associated `jndi-name` (if applicable).

Example

```
<resource-ref>
    <res-ref-name>jdbc/CheckingDataSource</res-ref-name>
    <jndi-name>file:///net/machine/datasources/OracleDataSource</jndi-name>
</resource-ref>
```

Related Elements

- "<res-ref-name> element"
- "<jndi-name> element"

<right-table>

```
<!ELEMENT right-table (table-name, column-list)>
```

When providing `<cmp2-info>`, this element defines one of two tables that share a column, one being a foriegn key of the other.

When using this element to describe `<relationships>`, the `<left-table>` is the source of that relationship, with the `<right-table>` being the destination. That is, the direction of the relationship proceeds from left-to-right. If you are defining a bi-directional relationship, you would create two `<table-ref>` elements for the same relationship, one for each direction. In many-to-many relationships, the `<right-table>` makes up one of two tables used to create a `<cross-table>` defining their intersections.

Example

```
<right-table>
     <table-name>CUSTOMER</table-name>
     <column-list>CUSTOMER_NO</column-list>
</right-table>
```

Related Elements

- "<table-ref>"
- "<left-table>"
- "<cross-table>"
- "<table-name>"
- "<column-list>"

<role-name>

```
<!ELEMENT role-name (#PCDATA)>
```

The role name for a security-role used by modules in the archive.

Example

```
<role-name>administrator</role-name>
```

Related Elements

– "<security-role>"

<security-role>

```
<!ELEMENT security-role (role-name, deployment-role?)>
```

Provides the name of a security role and (if applicable) a deployment role used by modules within the archive.

Example

```
<security-role>
        <role-name>administrator</role-name>
        <deployment-role>administrator</deployment-role>
</security-role>
```

Related Elements

– "<assembly-descriptor>"
– "<role-name>"
– "<deployment-role>"

<session>

```
<!ELEMENT session (ejb-name, bean-home-name?, bean-local-home-name?,
 timeout?, ejb-ref*, ejb-local-ref*, resource-ref*,resource-env-ref*,
property*)>
```

The `<session>` element is used to provide information about session beans contained within the archive. Child-nodes of this element allow you to specify the bean's various interfaces and references. Attributes particular to session beans (such as `timeout`) can also be provided, as well as generic properties specific to the individual session bean.

Example

```
<session>
       <ejb-name>UniqueIdGeneratorEJB</ejb-name>
       <bean-local-home-name>ejb/local/petstore/uidgen/UniqueIdGenerator</bean-
local-home-name>
       <timeout>0</timeout>
       <ejb-local-ref>
           <ejb-ref-name>ejb/local/Counter</ejb-ref-name>
```

```
        </ejb-local-ref>
</session>
```

Related Elements

- “<enterprise-beans>”
- “<ejb-name>”
- “<bean-home-name>”
- “<bean-local-home-name>”
- “<timeout>”
- “<ejb-ref> element”
- “<resource-ref> element”
- “<resource-env-ref> element”
- “<property> element”

<table>

```
<!ELEMENT table (#PCDATA)>
```

Specifies the name of a database table used by a CMP 1.x entity bean to populate its fields.

Example

```
<table>Course</table>
```

Related Elements

- “<database-map>”

<table-name>

```
<!ELEMENT table-name (#PCDATA)>
```

Specifies the name of a table for entity mapping or property setting.

Example

```
<table-name>CUSTOMER</table-name>
```

Related Elements

- “<cmp2-info>”
- “<table-properties>”

<table-properties>

```
<!ELEMENT table-properties (table-name, column-properties*, property*)>
```

Use this element to provide details about your entities' database resources. Using its child nodes, you provide the name of the table and any properties associated with it (such as `create_tables`). You can also specify properties specific to database columns using the `<column-properties>` child node.

Example

```
<table-properties>
 <table-name>CUSTOMER</table-name>
  <property>
   <prop-name>create-tables</prop-name>
   <prop-value>True</prop-value>
  </property>
</table-properties>
```

Related Elements

- "<ejb-jar>"
- "<table-name>"
- "<column-properties>"
- "<property> element"

<table-ref>

```
<!ELEMENT table-ref (left-table, cross-table*, right-table)>
```

When providing <cmp2-info>, you may have an entity that contains information persisted in multiple tables. These tables must be linked by at least one column representing a foreign key in the linked table. You can describe these relationships using the <table-ref> element. You use its child nodes, <left-table> and <right-table> to specify the tables and their shared column or columns.

This element is also used to describe relationships between tables using the <relationships> element. If this is the case, the descriptor uses <table-ref> elements to specify relationships between the <left-table> and the >right-table>. You must observe the following cardinalities:

- One <ejb-relationship-role> must be defined per direction; if you have a bi-directional relationship, you must define an <ejb-relationship-role> for each bean with each referencing the other.
- Only one <table-ref> element is permitted per relationship.

If you define a many-to-many relationship, you must also have the CMP engine create a cross-table which models a relationship between the left table and the right table. This is performed using the <cross-table> element.

Example

```
<table-ref>
  <left-table>
   <table-name>LINE_ITEM</table-name>
   <column-list>
    <column-name>LINE</column-name>
   </column-list>
  </left-table>
  <right-table>
   <table-name>QUANTITY</table-name>
   <column-list>
    <column-name>LINE</column-name>
   </column-list>
  </right-table>
 </table-ref>
```

Related Elements

- "<cmp2-info>"
- "<left-table>"
- "<right-table>"
- "<cross-table>"
- "<ejb-relationship-role>"

<timeout>

```
<!ELEMENT timeout (#PCDATA)>
```

Specifies the time in seconds for the session bean to wait between calls before it times out. The default is `0` seconds.

Example

```
<timeout>10</timeout>
```

Related Elements

- "<session>"

<url>

```
<!ELEMENT url (#PCDATA)>
```

JDBC 1.x only. The URL of the datasource being defined.

Related Elements

- "<datasource>"

<username>

```
<!ELEMENT username (#PCDATA)>
```

JDBC 1.x only. The username used to access the datasource being defined.

Related Elements

- "<datasource>"

<wait-timeout>

```
<!ELEMENT wait-timeout (#PCDATA)>
```

The number of seconds to wait for a free connection when `maxPoolSize` connections are already opened. When using the `maxPoolSize` property and the pool can't serve any more connections, threads looking for connections end up waiting for the connection(s) to become available for a long time if the wait time is unbounded (set to 0 seconds). You can set the wait-timeout period to suit your needs. This is analagous to the property `ejb.mdb.wait_timeout`.

Example

<pool> <max-size>0</max-size> <init-size>0</init-size> <wait-timeout>0</wait-timeout> </pool>

Related Elements

- "<message-driven>"
- "<pool>"
- "<max-size>"
- "<init-size>"

<where-clause>

```
<!ELEMENT where-clause (#PCDATA)>
```

The where clause is a necessary part of select statements when you want to delimit the extent of the returned records. Because the where clause syntax can be fairly complex, you must follow certain rules in the XML deployment descriptor file so that the EJB Container can correctly construct this clause.

To begin with, you are not obligated to use the literal "where" in your <where-clause>. You can construct a where clause without this literal and rely on the Container to supply it. However, the Container only does this if the is not an empty string; it leaves empty strings empty. For example, you could define a where clause as either:

```
<where-clause> where a = b </where-clause>
```

or

```
<where-clause> a = b </where-clause>
```

The Container converts `a = b` to the same where clause, `where a = b`. However, it leaves unmodified an empty string defined as `<where-clause> "" </where-clause>`.

Parameter substitution is an important part of the where clause. The Borland EJB Container does parameter substitution wherever it finds the standard SQL substitution prefix colon (:). Each parameter for substitution corresponds to a name of a parameter in the finder specification found in the XML descriptor.

The Container also supports compound parameters; that is, the name of a table followed by a column within the table. For this, it uses the standard dot (.) syntax, where the table name is separated from the column name by a dot. These parameters are also preceded by a colon.

An entity bean can also serve as a parameter in a finder method. You can use an entity bean as a compound type. To do so, you must tell the CMP engine which field to use from the entity bean's passed reference to the SQL query. If you do not use the entity bean as a compound type, then the Container substitutes the bean's primary key in the where clause.

Example

```
<where-clause>SELECT course_dept, course_number FROM
  Enrollment WHERE student  = :s[ejb/Student]</where-clause>
```

Related Elements

- "<finder>"
- "<method-signature>"
- "<load-state>"

4

jndi-definitions.xml

DTD

```
<!ELEMENT jndi-definitions (visitransact-datasource*, driver-datasource*, jndi-
object*)>

   <!ELEMENT visitransact-datasource (jndi-name, driver-datasource-jndiname,
   property*)>

   <!ELEMENT driver-datasource (jndi-name, datasource-class-name,
   log-writer?, property* )>

   <!ELEMENT jndi-object (jndi-name, class-name, property* )>

    <!ELEMENT property (prop-name, prop-type, prop-value)>
    <!ELEMENT prop-name (#PCDATA)>
    <!ELEMENT prop-type (#PCDATA)>
    <!ELEMENT prop-value (#PCDATA)>
    <!ELEMENT jndi-name (#PCDATA)>
    <!ELEMENT driver-datasource-jndiname (#PCDATA)>
    <!ELEMENT datasource-class-name (#PCDATA)>
    <!ELEMENT log-writer (#PCDATA)>
    <!ELEMENT class-name (#PCDATA)>
```

<jndi-definitions>

```
<!ELEMENT jndi-definitions (visitransact-datasource*, driver-datasource*, jndi-
object*)>
```

J2EE resource connection factory objects are bound to JNDI when they are deployed as a part of a JNDI Definitions Module. This module is similar to other J2EE standard Java archive types, and ends in the extension .dar. This module is also referred to as a DAR, therefore. This module adds to the standard J2EE module types like JAR, WAR, and RAR. It can be packaged as a part of an EAR, or deployed stand-alone.

The only contents of the DAR that you must provide is an XML descriptor file called jndi-definitions.xml, which contains all datasource definitions you want to bind to the JNDI namespace.

The <jndi-definitions> element is the root node of the schema. You use the <visitransact-datasource> and <driver-datasource> child nodes to define JDBC connection factory objects, and the <jndi-object> child node to define JMS resource connection factory objects.

Example

```
<jndi-definitions>
    <visitransact-datasource>
        <jndi-name>serial://datasources/Oracle</jndi-name
        <driver-datasource-jndiname>serial://datasources/OracleDriver</driver-
datasource-jndiname>
        <property>
            <prop-name>connectionType</prop-name>
            <prop-type>Enumerated</prop-type>
            <prop-value>Direct</prop-value>
        </property>
    </visitransact-datasource>
    <driver-datasource>
        <jndi-name>serial://datasources/OracleDriver</jndi-name>
        <datasource-class-
name>oracle.jdbc.pool.OracleConnectionPoolDataSource</datasource-class-name>
        <property>
            <prop-name>user</prop-name>
            <prop-type>String</prop-type>
            <prop-value>MisterKittles</prop-value>
        </property>
    </driver-datasource>
</jndi-definitions>
```

Related Elements

- "<visitransact-datasource>"
- "<driver-datasource>"
- "<jndi-object>"

<class-name>

```
<!ELEMENT class-name (#PCDATA)>
```

The name of the connection factory class(es) supplied by the JMS service provider and deployed as libraries to the BES Partition.

Example

```
<class-name>progress.message.jclient.QueueConnectionFactory</class-name>
```

Related Elements

- "<jndi-object>"

<datasource-class-name>

```
<!ELEMENT datasource-class-name (#PCDATA)>
```

Provides the name of the connection factory class supplied from the resource vendor. The class itself must be deployed to the BES Partition as a library.

Example

```
<datasource-class-name>oracle.jdbc.pool.OracleConnectionPoolDataSource</
datasource-class-name>
```

Related Elements

- "<driver-datasource>"

<driver-datasource>

```
<!ELEMENT driver-datasource (jndi-name, datasource-class-name, log-writer?,
property* )>
```

This element provides the other half of a datasource definition begun in <visitransact-datasource> by providing information on its driver. You specify the driver's JNDI name, which must be identical to the datasource's <driver-datasource-jndiname> defined in <visitransact-datasource>. You also provide the class name of the datasource driver, its logging behavior (if applicable), and properties specific to the JDBC resource, such as usernames, passwords, and so forth.

Example

```
<driver-datasource>
        <jndi-name>serial://datasources/OracleDriver</jndi-name>
        <datasource-class-
name>oracle.jdbc.pool.OracleConnectionPoolDataSource</datasource-class-name>
        <property>
            <prop-name>user</prop-name>
            <prop-type>String</prop-type>
            <prop-value>MisterKittles</prop-value>
        </property>
</driver-datasource>
```

Related Elements

- "<jndi-definitions>"
- "<visitransact-datasource>"
- "<jndi-name> element"
- "<datasource-class-name>"
- "<log-writer>"
- "<property> element"

<driver-datasource-jndiname>

```
<!ELEMENT driver-datasource-jndiname (#PCDATA)>
```

The JNDI name of the driver class supplied by the database or JMS vendor that you deploy as a library to BES Partitions. It is also the name that is referenced by the <jndi-name> child of the <driver-datasource> element that makes up the other half of a JDBC resource definition.

Example

```
<driver-datasource-jndiname>serial://datasources/OracleDriver</driver-
datasource-jndiname>
```

Related Elements

– "<visitransact-datasource>"

Element Hierarchy

initializeDocument() A tree for site navigation will open here if you enable JavaScript in your browser.

<jndi-name> element

```
<!ELEMENT jndi-name (#PCDATA)>
```

The name of the datasource as it will be referenced by JNDI. It is also the name found in the resource references of your enterprise beans.

Example

```
<jndi-name>serial://datasources/Oracle</jndi-name>
```

Related Elements

– "<visitransact-datasource>"

<jndi-object>

```
<!ELEMENT jndi-object (jndi-name, class-name, property* )>
```

You define the `<jndi-object>` element to register a JMS connection factory with JNDI. Using its child nodes, you provide the JNDI lookup for establishing JMS connections, the connection factory class, and any properties specific to the JMS provider that need to be passed to it.

Example

```
<jndi-object>
   <jndi-name>serial://jms/message</jndi-name>
   <class-name>progress.message.jclient.QueueConnectionFactory</class-name>
        <property>
            <prop-name>connectionURLS</prop-name>
            <prop-type>String</prop-type>
            <prop-value>localhost:2506</prop-value>
        </property>
        <property>
            <prop-name>sequential</prop-name>
            <prop-type>Boolean</prop-type>
            <prop-value>false</prop-value>
        </property>
        <property>
            <prop-name>loadBalancing</prop-name>
```

```
            <prop-type>Boolean</prop-type>
            <prop-value>true</prop-value>
        </property>
    </jndi-object>
```

Related Elements

- "<jndi-definitions>"
- "<jndi-name> element"
- "<class-name>"
- "<property> element"

<log-writer>

```
<!ELEMENT log-writer (#PCDATA)>
```

This element can be used to activate verbose modes for some vendor connection factory classes. Consult your resource's documentation for the use of this property.

Example

```
<log-writer>True</log-writer>
```

Related Elements

- "<driver-datasource>"

<prop-name> element

```
<!ELEMENT prop-name (#PCDATA)>
```

Specifies the name of the property to be set.

Example

```
<prop-name>vbroker.security.disable</prop-name>
```

Related Elements

- "<property> element"

<prop-type> element

```
<!ELEMENT prop-type (#PCDATA)>
```

Specifies the type of the property to be set.

Example

```
<prop-type>security</prop-type>
```

Related Elements

– "<property> element"

<prop-value> element

```
<!ELEMENT prop-value (#PCDATA)>
```

Specifies the value of the property to be set.

Example

```
<prop-value>false</prop-value>
```

Related Elements

– "<property> element"

<property> element

```
<!ELEMENT property (prop-name, prop-type, prop-value)>
```

This element is used to specify property values for various resources included in or referenced by the archive or its components. Each `property` entry specifies the property's name, type, and value using the appropriate sub-elements.

Example

```
<property>
    <prop-name>vbroker.security.disable</prop-name>
    <prop-type>security</prop-type>
    <prop-value>false</prop-value>
</property>
```

Related Elements

– "<prop-name> element"
– "<prop-type> element"
– "<prop-value> element"

<visitransact-datasource>

```
<!ELEMENT visitransact-datasource (jndi-name, driver-datasource-jndiname,
property*)>
```

Defines the datasource your application code will look up. You provide the JNDI name of the datasource and its driver, and any properties that need to be passed to it. Once defined, you then need to provide information on its driver in the `<driver-datasource>` sibling element.

Example

```
<visitransact-datasource>
  <jndi-name>serial://datasources/Oracle</jndi-name
 <driver-datasource-jndiname>serial://datasources/OracleDriver</driver-
datasource-jndiname>
  <property>
   <prop-name>connectionType</prop-name>
   <prop-type>Enumerated</prop-type>
   <prop-value>Direct</prop-value>
  </property>
</visitransact-datasource>
```

Related Elements

- "<jndi-definitions>"
- "<driver-datasource>"
- "<jndi-name> element"
- "<driver-datasource-jndiname>"
- "<property> element"

5

ra-borland.xml

DTD

```
<!ELEMENT connector (connection-factory)>
<!ELEMENT connection-factory (factory-name, factory-description?, jndi-name, ra-link-ref?, ra-
libraries?, pool-parameters?, (logging-enabled, log-file-name)?, property*, security-map*,
authorization-domain?)>
<!ELEMENT factory-name (#PCDATA)>
<!ELEMENT factory-description (#PCDATA)>
<!ELEMENT jndi-name (#PCDATA)>
<!ELEMENT ra-link-ref (#PCDATA)>
<!ELEMENT ra-libraries (#PCDATA)>
<!ELEMENT pool-parameters (initial-capacity?, maximum-capacity?, capacity-delta?, cleanup-enabled?,
cleanup-delta?, wait-timeout?, busy-timeout?, idle-timeout?)>
<!ELEMENT initial-capacity (#PCDATA)>
<!ELEMENT maximum-capacity (#PCDATA)>
<!ELEMENT capacity-delta (#PCDATA)>
<!ELEMENT cleanup-enabled (#PCDATA)>
<!ELEMENT cleanup-delta (#PCDATA)>
<!ELEMENT logging-enabled (#PCDATA)>
<!ELEMENT log-file-name (#PCDATA)>
<!ELEMENT property (prop-name, prop-type, prop-value)>
<!ELEMENT prop-name (#PCDATA)>
<!ELEMENT prop-type (#PCDATA)>
<!ELEMENT prop-value (#PCDATA)>
<!ELEMENT security-map (description?, user-role+, (use-caller-identity|run-as))>
<!ELEMENT user-role (#PCDATA)>
<!ELEMENT use-caller-identity EMPTY>
<!ELEMENT run-as (description?, role-name)>
<!ELEMENT role-name (#PCDATA)>
<!ELEMENT authorization-domain (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

<authorization-domain>

```
<!ELEMENT authorization-domain (#PCDATA)>
```

The authorization-domain element specifies the authorization domain to be used for determining the definable set of valid user roles.

Related Elements

– “<connection-factory>”

<busy-timeout>

```
<!ELEMENT busy-timeout (#PCDATA)>
```

The number of seconds to wait before a busy connection is released. If not specified, the default value of `600` is used.

Related Elements

– “<pool-parameters>”

<capacity-delta>

```
<!ELEMENT capacity-delta (#PCDATA)>
```

The `capacity-delta` element identifies the number of additional managed connections which the VisiConnect will attempt to obtain during resizing of the maintained connection pool. If not specified, the default value of `1` is used.

Related Elements

– “<pool-parameters>”

<cleanup-delta>

```
<!ELEMENT cleanup-delta (#PCDATA)>
```

The `cleanup-delta` element identifies the amount of time the Connection Pool Management will wait between attempts to reclaim unused Managed Connections. If not specified, the default value of `1` is used.

Related Elements

– “<pool-parameters>”

<cleanup-enabled>

```
<!ELEMENT cleanup-enabled (#PCDATA)>
```

The `cleanup-enabled` element indicates whether or not the Connection Pool should have unused Managed Connections reclaimed as a means to control system resources. Use the default value of `true` to reclaim unused connections.

Related Elements

- “<pool-parameters>”

<connection-factory>

```
<!ELEMENT connection-factory (factory-name, factory-description?, jndi-name,
ra-link-ref?, ra-libraries?, pool-parameters?, (logging-enabled, log-file-
name)?, property*, security-map*, authorization-domain?)>
```

The connection-factory element is the root element of the Borland specific deployment descriptor for the deployed resource adapter.

Related Elements

- “<connection-factory>”
- “<factory-description>”
- “<jndi-name>”
- “<ra-link-ref>”
- “<ra-libraries>”
- “<pool-parameters>”
- “<logging-enabled>”
- “<log-file-name>”
- “<property> element”
- “<security-map>”
- “<authorization-domain>”

<connector>

```
<!ELEMENT connector (connection-factory)>
```

The root node of the `ra-borland.xml` schema. Each connector defines a `connection-factory` used to connect to the JCA resource.

Related Elements

“<connection-factory>”

<description>

```
<!ELEMENT description (#PCDATA)>
```

Specifies a description for the security map or run-as role.

Related Elements

- “<security-map>”

<factory-description>

```
<!ELEMENT factory-description (#PCDATA)>
```

The factory-description element is used to provide text describing the parent element. The factory-description element should include any information that the deployer wants to describe about the deployed Connection Factory.

Related Elements

– "<connection-factory>"

<factory-name>

```
<!ELEMENT factory-name (#PCDATA)>
```

The factory-name element defines that logical name that will be associated with this specific deployment of the Resource Adapter and its corresponding Connection Factory.

The value of factory-name can be used in other deployed Resource Adapters via the ra-link-ref element. This will allow multiple deployed Connection Factories to utilize a common deployed Resource Adapter, as well as share configuration specifications.

Related Elements

– "<connection-factory>"

<idle-timeout>

```
<!ELEMENT idle-timeout (#PCDATA)>
```

Specifies the number of seconds to allow a pooled connection to remain in an idle state before it is closed. If not specified, the default value of 600 is used. Idle connections are only checked for state change every sixty seconds.

Related Elements

– "<pool-parameters>"

<initial-capacity>

```
<!ELEMENT initial-capacity (#PCDATA)>
```

The initial-capacity element identifies the initial number of managed connections which VisiConnect will attempt to obtain during deployment. If not specified, the default value, 1, is used.

Related Elements

– "<pool-parameters>"

<jndi-name>

```
<!ELEMENT jndi-name (#PCDATA)>
```

The jndi-name element defines the name that will be used to bind the Connection Factory Object into the JNDI Namespace. Client EJBs and Servlets will use this same JNDI in their defined Reference Desciptor elements of the Borland specific deployment descriptors.

Related Elements

– "<connection-factory>"

<log-file-name>

```
<!ELEMENT log-file-name (#PCDATA)>
```

The `log-file-name` element specifies the name of the log file which output generated from either the `ManagedConnectionFactory` or a `ManagedConnection` are sent. The full address of the file name is required.

Related Elements

– "<connection-factory>"

<logging-enabled>

```
<!ELEMENT logging-enabled (#PCDATA)>
```

The logging-enabled element indicates whether or not the log writer is set for either the `ManagedConnectionFactory` or `ManagedConnection`. If this element is set to `true`, output generated from either the `ManagedConnectionFactory` or `ManagedConnection` will be sent to the file specified by the `log-filename` element. If not specified, the default value of `false` is used.

Related Elements

– "<connection-factory>"

<maximum-capacity>

```
<!ELEMENT maximum-capacity (#PCDATA)>
```

The `maximum-capacity` element identifies the maximum number of managed connections which VisiConnect will allow. Requests for newly allocated managed connections beyond this limit will result in a `ResourceAllocationException` being returned to the caller. If not specified, the default value of `10` is used.

Related Elements

– "<pool-parameters>"

<pool-parameters>

```
<!ELEMENT pool-parameters (initial-capacity?, maximum-capacity?, capacity-
delta?, cleanup-enabled?, cleanup-delta?, wait-timeout?, busy-timeout?, idle-
timeout?)>
```

The pool-parameters element is the root element for providing Connection Pool specific parameters for this Connection Factory.

VisiConnect will use these specifications in controlling the behavior of the maintained pool of Managed Connections.

This is an optional element.Failure to specify this element or any of its specific element items will result in default values being assigned. Refer to the description of each individual element for the designated default value.

Related Elements

- "<connection-factory>"
- "<initial-capacity>"
- "<maximum-capacity>"
- "<capacity-delta>"
- "<cleanup-enabled>"
- "<cleanup-delta>"
- "<wait-timeout>"
- "<busy-timeout>"
- "<idle-timeout>"

<prop-name> element

```
<!ELEMENT prop-name (#PCDATA)>
```

Specifies the name of the property to be set.

Example

```
<prop-name>vbroker.security.disable</prop-name>
```

Related Elements

- "<property> element"

<prop-type> element

```
<!ELEMENT prop-type (#PCDATA)>
```

Specifies the type of the property to be set.

Example

```
<prop-type>security</prop-type>
```

Related Elements

- "<property> element"

<prop-value> element

```
<!ELEMENT prop-value (#PCDATA)>
```

Specifies the value of the property to be set.

Example

```
<prop-value>false</prop-value>
```

Related Elements

- "<property> element"

<property> element

```
<!ELEMENT property (prop-name, prop-type, prop-value)>
```

This element is used to specify property values for various resources included in or referenced by the archive or its components. Each `property` entry specifies the property's name, type, and value using the appropriate sub-elements.

Example

```
<property>
    <prop-name>vbroker.security.disable</prop-name>
    <prop-type>security</prop-type>
    <prop-value>false</prop-value>
</property>
```

Related Elements

- "<prop-name> element"
- "<prop-type> element"
- "<prop-value> element"

<ra-libraries>

```
<!ELEMENT ra-libraries (#PCDATA)>
```

The ra-libraries element identifies the directory location to be used for all native libraries present in this resource adapter deployment. As part of deployment processing, all encountered native libraries will be copied to the location specified.

It is the responsibility of the Administrator to perform the necessary platform actions such that these libraries will be found at runtime.

Related Elements

- "<connection-factory>"

<ra-link-ref>

```
<!ELEMENT ra-link-ref (#PCDATA)
```

The ra-link-ref element allows for the logical association of multiple deployed Connection Factories with a single deployed Resource Adapter. The specification of the optional `ra-link-ref` element with a value identifying a separately deployed Connection Factory will result in this newly deployed Connection Factory sharing the Resource Adapter which had been deployed with the referenced Connection Factory.

In addition, any values defined in the referred Connection Factories deployment will be inherited by this newly deployed Connection Factory unless specified.

Related Elements

- "<connection-factory>"

<role-name>

```
<!ELEMENT role-name (#PCDATA)>
```

The role-name element contains the name of a security role. The name must conform to the lexical rules for an `NMTOKEN`.

Related Elements

- "<run-as>"

<run-as>

```
<!ELEMENT run-as (description?, role-name)>
```

The run-as element specifies the run-as identity to be used for the execution of the enterprise bean.It contains an optional description, and the name of a security role.

Related Elements

- "<security-map>"
- "<description>"
- "<role-name>"

<security-map>

```
<!ELEMENT security-map (description?, user-role+, (use-caller-identity|run-
as))>
```

The `security-map` element specifies whether the caller's security identity is to be used for the execution of the methods of the enterprise bean or whether a specific run-as identity is to be used. It contains an optional description and a specification of the security identity to be used.

Each `security-map` element provides a mechanism to define appropriate Resource Role values for Resource Adapter/EIS authorization processing, through the use of the run-as element.

This element allows for the specification of a defined set of user roles and the corresponding run-as roles (representing EIS identities) that should be used when allocating Managed Connections and Connection Handles.

A default Resource run-as role can be defined for the Connection Factory via the map. By specifying a user-role value of `*` and a corresponding run-as role, the defined run-as will be utilized whenever the current role is NOT matched elsewhere in the map.

Although this element is optional, it must be specified in some form if Container Managed Sign-on is supported by the Resource Adapter and used by ANY client. In addition, the deployment-time population of the Connection Pool with Managed Connections will be attempted using the defined default run-as role if one is specified.

Related Elements

- "<connection-factory>"
- "<description>"

- “<user-role>”
- “<use-caller-identity>”
- “<run-as>”

<use-caller-identity>

```
<ELEMENT use-caller-identity EMPTY>
```

The `use-caller-identity` element specifies that the caller's security identity be used as the security identity for the execution of the Resource Adapter's methods. This is an empty element. If not used, the `run-as` element must be specified instead.

Related Elements

- “<security-map>”

<user-role>

```
<!ELEMENT user-role (#PCDATA)>
```

The `user-role` element contains one or more role names, defined for use as the security identity, or mapped to a appropriate Resource Role run-as identity, for interactions with the resource.

Related Elements

- “<security-map>”

<wait-timeout>

```
<!ELEMENT wait-timeout (#PCDATA)>
```

Specifies the number of seconds to wait for a free connection when the maximum number of connections are already opened. You can set an unbounded wait time by using the value `0` for this element. If not specified, the default value of `30` is used.

Related Elements

- “<pool-parameters>”

6

web-borland.xml

DTD

```
<!ELEMENT web-app (context-root?, resource-env-ref*, resource-ref*, ejb-ref*,
    property*, web-deploy-path*, authorization-domain?, security-role*)>

    <!ELEMENT ejb-ref (ejb-ref-name, jndi-name?)>
    <!ELEMENT resource-ref (res-ref-name, jndi-name)>
    <!ELEMENT resource-env-ref (resource-env-ref-name, jndi-name)>
    <!ELEMENT property (prop-name, prop-type, prop-value)>
    <!ELEMENT web-deploy-path (service, engine, host)>
    <!ELEMENT context-root (#PCDATA)>
    <!ELEMENT prop-name (#PCDATA)>
    <!ELEMENT prop-type (#PCDATA)>
    <!ELEMENT prop-value (#PCDATA)>
    <!ELEMENT ejb-ref-name (#PCDATA)>
    <!ELEMENT jndi-name (#PCDATA)>
    <!ELEMENT res-ref-name (#PCDATA)>
    <!ELEMENT resource-env-ref-name (#PCDATA)>
    <!ELEMENT service (#PCDATA)>
    <!ELEMENT engine (#PCDATA)>
    <!ELEMENT host (#PCDATA)>
    <!ELEMENT authorization-domain (#PCDATA)>
    <!ELEMENT security-role (role-name, deployment-role?)>
    <!ELEMENT role-name (#PCDATA)>
    <!ELEMENT deployment-role (#PCDATA)>
```

<web-app>

```
<!ELEMENT web-app (context-root?, resource-env-ref*, resource-ref*, ejb-ref*,
    property*, web-deploy-path*, authorization-domain?, security-role*)>
```

The root node of the deployment descriptor for a web application. This descriptor extends the `web.xml` standard deployment descriptor, allowing you to provide extra properties, supply exactly where the web application should be hosted, and provide security information for the application.

Example

```
<web-app>
    <authorization-domain>default</authorization-domain>
</web-app>
```

Related Elements

- "<context-root>"
- "<resource-env-ref> element"
- "<resource-ref> element"
- "<ejb-ref> element"
- "<property> element"
- "<web-deploy-path>"
- "<authorization-domain>"
- "<security-role>"

<authorization-domain>

```
<!ELEMENT authorization-domain (#PCDATA)>
```

The name of the authorization domain to which the application will belong.

Example

```
<authorization-domain>GroupJ</authorization-domain>
```

Related Elements

- "<web-app>"

<context-root>

```
<!ELEMENT context-root (#PCDATA)>
```

Normally the name of a web application is the same as the WAR name (only lacking the `.war` extension). Using the `<context-root>` contruct, you can change the name of the application to any name you wish.

Example

```
<context-root>alienWare</context-root>
```

Related Elements

- "<web-app>"

<deployment-role>

```
<!ELEMENT deployment-role (#PCDATA)>
```

The role name for BES role the web application will run under.

Example

```
<deployment-role>administrator</deployment-role>
```

Related Elements

– "<security-role>"

<ejb-name>

```
<!ELEMENT ejb-name (#PCDATA)>
```

Use the `<ejb-name>` element to provide a name for the enterprise javabean you are defining. This element is analagous to the same element in `ejb-jar.xml`, providing a name used to look-up the bean remotely.

Example

```
<ejb-name>clerk</ejb-name>
```

Related Elements

– "<ejb-ref> element"

<ejb-ref> element

```
<!ELEMENT ejb-ref (ejb-ref-name, jndi-name?)>
```

This element is used to define EJB references used by the web application. Each EJB reference contains an `ejb-ref-name` used by the application and its associated `jndi-name`.

Example

```
<ejb-ref>
    <ejb-ref-name>ejb/Sort</ejb-ref-name>
    <jndi-name>sort</jndi-name>
</ejb-ref>
```

Related Elements

– "<web-app>"
– "<ejb-ref-name> element"
– "<jndi-name> element"

<ejb-ref-name> element

```
<!ELEMENT ejb-ref-name (#PCDATA)>
```

This element provides the name of an EJB used as a resource reference by the web application.

Example

```
<ejb-ref-name>ejb/Sort</ejb-ref-name>
```

Related Elements

- "<ejb-ref> element"

<engine>

```
<!ELEMENT engine (#PCDATA)>
```

Contains the engine name. This must correspond to an engine defined in Tomcat's `server.xml` file.

Example

```
<engine>cyrpi</engine>
```

Related Elements

- "<web-app>"
- "<service>"
- "<host>"

<host>

```
<!ELEMENT host (#PCDATA)>
```

Containst the host name. This must correspond to a host defined in Tomcat's `server.xml` file.

Example

```
<host>it3</host>
```

Related Elements

- "<web-app>"
- "<service>"
- "<engine>"

<jndi-name> element

```
<!ELEMENT jndi-name (#PCDATA)>
```

This element provides the JNDI service lookup for a resource referenced by the web application.

Example

```
<jndi-name>sort</jndi-name>
```

Related Elements

- "<ejb-ref> element"
- "<resource-ref> element"
- "<resource-env-ref> element"

<prop-name> element

```
<!ELEMENT prop-name (#PCDATA)>
```

Specifies the name of the property to be set.

Example

```
<prop-name>vbroker.security.disable</prop-name>
```

Related Elements

- "<property> element"

<prop-type> element

```
<!ELEMENT prop-type (#PCDATA)>
```

Specifies the type of the property to be set.

Example

```
<prop-type>security</prop-type>
```

Related Elements

- "<property> element"

<prop-value> element

```
<!ELEMENT prop-value (#PCDATA)>
```

Specifies the value of the property to be set.

Example

```
<prop-value>false</prop-value>
```

Related Elements

- "<property> element"

<property> element

```
<!ELEMENT property (prop-name, prop-type, prop-value)>
```

This element is used to specify property values for various resources included in or referenced by the archive or its components. Each `property` entry specifies the property's name, type, and value using the appropriate sub-elements.

Example

```
<property>
    <prop-name>vbroker.security.disable</prop-name>
    <prop-type>security</prop-type>
    <prop-value>false</prop-value>
</property>
```

Related Elements

- "<prop-name> element"
- "<prop-type> element"
- "<prop-value> element"

<resource-env-ref-name> element

```
<!ELEMENT resource-env-ref-name (#PCDATA)>
```

This element provides the name the web application uses to access a resource environment reference.

Example

```
<res-env-ref-name>jdbc/CheckingDataSource</res-ref-name>
```

Related Elements

- "<web-app>"
- "<resource-env-ref> element"

<res-ref-name> element

```
<!ELEMENT res-ref-name (#PCDATA)>
```

This element provides the name the web application uses to access a resource reference.

Example

```
<res-ref-name>jdbc/CheckingDataSource</res-ref-name>
```

Related Elements

- "<resource-ref> element"

<resource-env-ref> element

```
<!ELEMENT resource-env-ref (res-env-ref-name, jndi-name?)>
```

This element is used to map a resource environment reference used by the web application to a name in JNDI. Each resource environment reference contains a `res-env-ref-name` used by the bean and its associated `jndi-name`.

Example

```
<resource-env-ref>
    <res-env-ref-name>jdbc/CheckingDataSource</res-ref-name>
    <jndi-name>file:///net/machine/datasources/OracleDataSource</jndi-name>
</resource-env-ref>
```

Related Elements

- "<web-app>"
- "<resource-env-ref-name> element"
- "<jndi-name> element"

<resource-ref> element

```
<!ELEMENT resource-ref (res-ref-name, jndi-name?)>
```

This element is used to define resource references used by the web application. Each resource reference contains an `res-ref-name` used by the application and its associated `jndi-name`.

Example

```
<resource-ref>
    <res-ref-name>jdbc/CheckingDataSource</res-ref-name>
    <jndi-name>file:///net/machine/datasources/OracleDataSource</jndi-name>
</resource-ref>
```

Related Elements

- "<web-app>"
- "<res-ref-name> element"
- "<jndi-name> element"

<role-name>

```
<!ELEMENT role-name (#PCDATA)>
```

The role name for a security-role used by the web application that will be mapped to a role in the BES deployed environment.

Example

```
<role-name>administrator</role-name>
```

Related Elements

- "<security-role>"

<security-role>

```
<!ELEMENT security-role (role-name, deployment-role?)>
```

Maps a role for the web application (found in `web.xml`) to a deployment-role in the Borland Enterprise Server.

Example

```
<security-role>
        <role-name>administrator</role-name>
        <deployment-role>administrator</deployment-role>
</security-role>
```

Related Elements

- "<web-app>"
- "<role-name>"
- "<deployment-role>"

<service>

```
<!ELEMENT service (#PCDATA)>
```

Contains the service name. This must correspond to a service defined in Tomcat's `server.xml` file.

Example

```
<service>tomcatX</service>
```

Related Elements

- "<web-app>"
- "<engine>"
- "<host>"

<web-deploy-path>

```
<!ELEMENT web-deploy-path (service, engine, host)>
```

Tomcat's `server.xml` file can define one or more hosts under one or more engines, which themselves are under a given service. If you would like specify exactly where to deploy the web application under the Tomcat container, use this element.

Example

```
<web-deploy-path>
    <service>tomcatX</service>
```

```
    <engine>cyrpi</engine>
    <host>it3</host>
</web-deploy-path>
```

Related Elements

- “<web-app>”
- “<service>”
- “<engine>”
- “<host>”

Index

Symbols

A

B

C

D

E

J

M

P

R

S

T

W