Borland AppServer™ 6.7

Management Console User's Guide

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.

BAS67UsersGuide
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

Using the Borland Management Console

This section provides an overview of the Borland Management Console including how to start the Management Console and what is included in the Management Console.

What is the Borland Management Console?

The Borland Management Console provides a graphical user interface which functions as the main control point to let you view management hubs and configurations on the network, start and stop hubs and managed objects, edit configurations, and manage the services and tools that enable your enterprise to build, deploy, and manage component-based enterprise applications. The Management Console also enables you to view deployed modules, set deployment properties, and monitor performance. The Management Console can run on any machine from which users want to view or modify the distributed system.

Starting and logging onto the Borland Management Console

To start the Management Console:

1 Launch the Management Console using one of the following methods:

Windows

From the Start menu choose Programs|Borland Deployment Platform| Management Console or open a Command Prompt, and type the following command:

```
console
```

Note

To recognize the `console` command, your path system variable must include the `<install_dir>\bin` directory, or you can enter the path explicitly.

UNIX

Open a command shell and enter the following command:

```
console
```

Note

To recognize the `console` command, your path system variable must include the `<install_dir>/bin` directory, or you can enter the path explicitly.

The Administration Security Credentials screen appears.

Figure 2.1 Administration Security Credentials screen with default credential settings

1 Enter your User identity, Password, and Realm.

To use the Borland default credential settings, enter the password:

```
admin
```

2 Click OK.

The Management Console appears.

Management Console views

The leftmost column of the Management Console window provides the following buttons you use to access the following views:

- Hubs—provides access to Agents and Configurations, and in particular, within a Borland AppServer (AppServer) configuration it provides access to the AppServer Partitions, AppServer Services and application archives
- Installations—provides a view to work with installations on the local machine.
- VisiBroker—opens the VisiBroker Console.

Management Console menus

The Management Console provides the following main menus:

- Console
- View
- Wizards

- Tools
- Help

Console menu

The following table describes the commands in the Console menu.

Console menu command	Description
Refresh	Manually causes state information shown in the Management Console to be updated.
Preferences	Opens the Preferences dialog box which you can use to set Management Console and configuration settings. See "Setting Management Console preferences" for information about setting preferences in this dialog.
Set Identity	Sets the credentials (User identity, Password, and Realm) used for authentication during hub and agent discovery.
Exit	Dismisses the Management Console.

View menu

The following table describes the commands on the View menu.

View menu command	Description
Messages	Shows or hides the errors window.
Tool Bar	Shows or hides the tool bar at the top of the Management Console window.
Status Bar	Shows or hides the status bar at the bottom of the Management Console window.

Wizards menu

The following table describes the commands on the Wizards menu.

Wizards menu command	Description
Deployment Wizard	Starts the wizard that helps you deploy J2EE modules.
Merge Wizard	Starts the wizard that helps you merge multiple J2EE modules into a single module.
Verify Wizard	Starts the wizard that helps you check an archive file for correctness and consistency, and to check if all the elements required for deploying your application are in place.
XML Migration Wizard	Starts the wizard that converts a J2EE version 1.2 module to version 1.3, or to convert a J2EE version 1.3 module to version 1.2.
Stub Generation Wizard	Starts the wizard to generate stub classes that can be stored in various archive types.
Remove Stubs Wizard	Starts the wizard to remove stub classes from an archive.
Apply Patch Wizard	Starts the wizard to apply one or more patches to an archive.
JAR Wizard	Starts the wizard to compress or extract objects in a JAR file.

Tools menu

The following table describes the commands on the Tools menu.

Tools menu command	Description
Archive Tool	Opens the Archive Tool. See "Using the Archive Tool" for details.
Borland Log Tool	Opens the Borland Log Tool.
Deployment Descriptor Editor	Opens the DDEditor. See "Using the Deployment Descriptor Editor" for details.
JDataStore Explorer	Opens the JDataStore Explorer. See the JDataStore documentation located at `http://info.borland.com/techpubs/jdatastore/` for details.
Launcher Generation Wizard	Starts the Launcher Generation Wizard.
License Manager	Launches the License Manager. See "Using the License Manager" for details.
Optimizeit Profiler	Starts the Optimizeit Profiler viewer if you have configured it in the Management Console Preferences. See "Tools preferences" for information about Optimizeit client configuration.
Optimizeit ServerTrace	Starts the Optimizeit ServerTrace viewer if you have configured it in the Management Console Preferences. See "Tools preferences" for information about Optimizeit client configuration.
TCP Monitor	Starts the TCP Monitor.
Tibco Admin Console	Launches the Tibco Admin Console. For more information about JMS see "JMS service".
Chainsaw Log4J Browser	Opens the Chainsaw Log4J Browser.
JNDI Browser	Opens the JNDI Browser. See "Using the JNDI Browser" for more information.
Web Services Explorer	Launches the Web Services Explorer.
VisiBroker Properties File Editor	Opens an editor allowing you to edit the VisiBroker properties file.

Help menu

This menu provides you with access to each of the online Help documents, access to the AppServer developer support web sites, and the About screen for the Borland Management Console.

The About screen contains tabs that display some important information about your Borland software.

About screen tab	Description
About	Displays the Borland Deployment Platform version number and copyright information.
General System Information	Displays various system configuration settings that AppServer has detected such as the operating system, Java version, Java vendor, Java Compiler, and so forth.
Java Properties	Shows the Java virtual machine property settings in use by AppServer.

Setting Management Console preferences

Management Console preferences enable you to specify configuration, operation, and appearance settings used by the Management Console such as the SmartAgent port, the default polling interval for performance information displayed in the Management Console, and so forth.

To set Management Console preferences:

1. Start the Management Console and choose Preferences from the Console menu. The Preferences dialog box appears with a list of preferences grouped into the following categories:
   - General: controls user interface options and settings.
   - Security: provides properties relating to the administration and management of the security environment.
   - Discovery: determines the set of hubs appearing in the tree.
   - State: controls the updating of state information.
   - Logs: controls the updating and display of the contents of log and audit files.
   - Tools: Contains configuration information for tools (such as Optimizeit) launched by the Management Console.
   - VisiBroker: controls user interface options for the VisiBroker Console.
1. Navigate through the tabs and select the preferences as desired.
2. When you have finished making your selections, click OK.

The following sections provide details on each set of preferences.

General preferences

The General preferences provide settings to control various Management Console user interface elements. This tab provides the following options:

- **Look and feel:** Sets the display format and behavior of the Management Console windows. The available options are: Metal, CDE/Motif, Windows (Microsoft Windows platforms only), or Borland.
- **Tab memory:** Specifies the view state information the Management Console uses. The following options are available:
  - **Don't remember last visited tab pane:** Tells the Management Console to open each node in the tree with the General tab displayed on the right.
  - **Remember last visited tab pane by type:** Tells the Management Console to open a node on the same type of tab (on a similar node) that was most recently viewed. For example, if the Properties tab is currently in view and you click on another node that has a Properties tab, the Management Console first displays the Properties tab for that node.
  - **Remember last visited tab pane by type and name:** Tells the Management Console to open a node that had been expanded earlier in the Management Console session to the tab that was last in view when that node was selected.
- **Sound beep on errors:**: If checked, the Management Console sounds an alarm when an error occurs.
- **Enable debug output**: Tells the Management Console to report debugging information in the Errors pane at the bottom of the Management Console.
- **HTML Browser Setup:** Specifies the path to the Web browser you wish to use with the Management Console. Use the Browse button to locate an installed Web browser.

Security preferences

The Security preferences provide settings relating to the administration and management of the security environment. This tab provides the following options:

- **Default realm:** Specifies the name of the authentication realm used by the Management Console to interact with each AppServer.
- **Default user:** Specifies the user name used by the Management Console to interact with each AppServer.
- **Enable Security:** Determines how the Management Console handles security:
  - When checked, enables the Management Console to communicate with an agent regardless if it has security enabled or not. When the Management Console receives a request from an agent with security enabled, however, the Management Console must first pass the user's login credentials (realm, username, and password) to that agent for authentication before the Management Console can access services on that agent.
  - When this box is not checked, the Management Console will communicate only with agents that do not have security enabled.

Discovery preferences

The Discovery preferences provide settings to determine the set of management hubs appearing in the Navigation pane of the Management Console. This tab provides the following options:

- **Autodiscover hubs on local subnet:** Enables the Location Service and/or the Naming Service to automatically locate all management hubs using a specified management port on the local area network and display them in the Management Console. It is strongly recommended that you do not change this value.

  Note

  The Management port number shown on this tab is not the same as the Smart Agent (also known as osagent) port.
- **Autodiscover hubs via proxy hubs:** Enables the Location Service or the Naming Service, or both, to automatically locate all management hubs using a specified management port that are hosted on a specified proxy server.
- **Enter an explicit list of hubs:** Lets you manually specify which management hubs are displayed in the Management Console.
- **Show hub/agent host in name:** Check this box if you want the host name of the machine on which the hub or agent is running to be displayed next to the hub's or agent's display name in the Management Console navigation tree.

State preferences

The State preferences provide settings to control the updating of state information in the Management Console. It also provides a legend to the icons representing the state of objects appearing in the Management Console. This tab provides the following options:

- **Enable polling for events:** When checked, tells the Management Console to automatically update information displayed about the state of the configurations, partitions, and services (such as running, stopped, and so forth).

  The following settings determine the time intervals (in milliseconds) of how often the Management Console checks to verify the state of these objects, and they specify how often the state is updated in the Navigation pane in the Management Console Hubs View:

  - **Background polling interval:** Determines how frequently the Management Console checks the state of an object in the Hubs View when no user interaction is initiated.

  - **Foreground polling interval:** Determines how frequently the Management Console checks the state of the objects when the user performs any action that causes the user name object state to change, such as stopping, starting, or restarting an object.
  - **Number of foreground cycles:** Determines how many times within the specified Foreground Polling Interval that the Management Console checks the object state.
  - **Display license warnings and events:** When checked, tells the Management Console to display warnings and events relating to licensed features.
- **Enable background refreshes:** When checked, tells the Management Console to automatically update information displayed about the changes in the navigation tree, such as when a configuration, partition, or service is added or removed. Clear this check box to reduce the processing overhead used by Management Console polling activity. If this box is unchecked you can manually update the information displayed on the Management Console by clicking the Console-Refresh menu command or the Refresh toolbar button.
  - **Refresh every:** Determines (in milliseconds) how frequently the Management Console checks and refreshes the display of the state of the navigation pane.
- **State Legend:** Shows the icons used by the Management Console to represent the various object states.

Logs preferences

The Logs preferences provide settings to control the updating and display of the contents of the log and audit files. This tab provides the following options:

- **Polling:**
  - **Show at most *n* lines:** Sets the maximum number of message lines (log entries) shown. For example, if you specify 600 lines, only the last 600 lines in the log file are shown.
  - **Enable polling of log file contents:** When checked, tells the Management Console to generate log file statistics. If this box is unchecked, polling for statistics is disabled.
  - **Polling interval:** Sets the time interval (in milliseconds) for how often log files are updated.
- **Filters:**
  - **Filters kept in history:** Sets the number of log filters used in the Management Console to be kept in the history.
  - **Clear History:** Clears the history of all log filters previously used in the Management Console.

Tools preferences

Use the Tools preferences tab to specify an absolute (fully qualified) path location in which the local Optimizeit Profiler or Optimizeit ServerTrace client is installed. Configuring these paths will enable the Management Console toolbar buttons and Tools menu options that allow you to launch the Profiler or ServerTrace viewers from the Management Console.

Enter the full path to the Optimizeit executable in the Path field, or click Browse to locate the local Optimizeit installation.

For information about configuring the Management Console to interoperate with Optimizeit on the server side, see “Using Optimizeit Profiler and ServerTrace.”

Note

If you are using the Management Console to manage a remote server, you must also install Optimizeit on the machine on which the server is running in order to generate snapshots on the remote server.

VisiBroker preferences

The VisiBroker tab is used to specify user interface options for the VisiBroker Console. This tab provides the following options:

- **Remove Stale Service Reference:** Enable this option to remove all stale VisiBroker services references that were configured in the My Services folder of the VisiBroker Console main view.
- **Enable DNS Lookup:** Enable this option to have the VisiBroker Console resolve any related DNS information to display in the Console. For example, the Console would display an IP address as its registered host name.

  Note

  Because any DNS lookup is an expensive resource operation, the VisiBroker Console provides this option to enable or disable DNS lookup.

3

Using Partition services

This section explains how to use the Partition services. Partition services are represented in the Navigation Pane of the Management Console's Hubs View as child nodes of their parent Partitions.

Starting Partition services

Partition services are started automatically when the Partition is started. To stop using a Partition service, right-click the service icon in the Navigation Pane, and select Disable.

Configuring Partition services

To access the property configuration dialog for each Partition service:

1 Right-click the service's icon in the Navigation Pane, and select Properties.

2 Click OK when you are finished.

For information about the configurable properties for each service see the following sections:

- Configuring Connector Service properties
- Configuring EJB Container properties
- Configuring JDataStore Server properties
- Configuring Naming Service properties
- Configuring Session Storage Service properties
- Configuring Transaction Manager properties
- Configuring Web Container properties

Note

The Lifecycle Interceptor Manager has no configurable properties in the Management Console.

Configuring Connector Service properties

You can configure the following Connector Service properties:

- **Trace level:** Specify the verbosity of logging to the event log from the Trace level drop-down list.
- **Use pass by value for calls:** When checked, this setting allows connectors to pass parameters by value. If not checked, all calls must be made by reference.
- **Enable statistics gathering:** If not checked, this option reduces the amount of statistical information gathered by the connector, resulting in improved performance. The default is true.
- **Use java serialization (not IIOP):** When checked, this option causes connectors to be serialized using Java serialization rather than IIOP serialization.

Configuring EJB Container properties

You can configure the following EJB Container properties:

Settings:

- **Trace level:** Specify the verbosity of logging to the event log from the Trace level drop-down list.
- **Passivation timeout:** Specify for Stateful session beans the amount of time allowed to elapse before the bean's state is persisted.
- **Passivation factory name:** Specify the name of the JSS service that manages passivation persistence for this container.

Flags:

- **Use pass by value for intra bean calls:** When checked, this setting allows EJBs to pass parameters by value when making calls to local beans. If not set, all calls must be made by reference.
- **Use java serialization (not IIOP):** When checked, causes Stateful session beans to be serialized using Java serialization rather than IIOP serialization.
- **Enable statistics gathering:** Checked by default. If unchecked, reduces the amount of statistical information gathered by the container, resulting in improved performance.
- **Enable SIDL:** When checked, enables EJBs to be accessed using SIDL (Simplified Interface Description Language).

Message-driven bean thread pool:

- **Minimum number of threads:** Set the minimum number of message-driven bean dispatch threads allowed to exist in that thread pool.
- **Maximum number of threads:** Set the maximum number of message-driven bean dispatch threads allowed to exist in that thread pool.
- **Maximum thread idle time:** Set the maximum time a message driven bean thread is allowed to be idle before it is reaped from the pool.

Configuring JDataStore Server properties

You can configure the following JDataStore Server properties:

- **Port:** Set the port used to listen to JDBC requests.
- **Socket Timeout:** Specify the time a call to `accept()` in JDS will block until an IO exception is raised.
- **Temp Directory:** Specify the temporary directory used for all JDataStore connections. If no directory is specified, the current directory is used.

Configuring Naming Service properties

You can configure the following Naming Service properties:

- **Backing store:** Specifies the type of pluggable backing store to use for storing persistent data. It can be memory, a relational database, or an LDAP directory server. The supported types of backing store adaptor are:
  - **InMemory:** In-memory adaptor.
  - **JDBC:** JDBC adaptor for relational databases. This setting requires that you enter a Login name, Password, URL, location of the JDBC driver, and Pool size value.
  - **Dx:** DataExpress adaptor. This setting requires that you enter a Login name, Password, and URL.
  - **JNDI:** For LDAP only. This setting requires that you enter a Login name, Password, URL, Initial factory, and Authentication value.

Configuring Session Storage Service properties

You can configure the following Session Storage Service properties:

- **Factory name:** Specify the name under which the Session Storage Service registers itself with the Smart Agent. If not specified, Borland AppServer (AppServer) uses a value that is combined from the server name and the Partition name as follows: <server name>/<partition_name>.
- **Working directory:** Specify the directory location where the Session Storage Service looks for the JDataStore back end database. The default location is the working Partition directory, for example: <install_dir>/var/servers/<server name>/partitions/<partition name>.
- **Persistent store:** Specify the name of the JDataStore file which the Session Storage Service uses as back end database. The default value is `jss_storage`.
- **User name:** Specify the user name under which Session Storage Service communicates with the JDataStore back end database. The default value is `default-user-name`.
- **Max idle:** Specify (in seconds) the least amount of time that a session stays present in the back end database without getting accessed. If set to 0 (zero), the session stays in the database indefinitely.
- **Soft commit:** When checked (default), the Session Storage Service uses the JDataStore back end database with a Soft Commit mode enabled. Unchecking this check box improves performance of the Session Storage Service, but can lead to recently committed transactions being rolled back after a system crash. For more information, see the JDataStore online documentation at `http://www.borland.com/techpubs/jdatastore`.
- **Debug:** When checked, this setting tells the Session Storage Service to run with debug information enabled. By default this option is unchecked.

For more information, see “Managing and configuring the JSS” in the Java Session Service (JSS) configuration chapter of the *AppServer Developer's Guide*.

Configuring Transaction Manager properties

You can configure the following Transaction Manager properties:

- **Allow Unrecoverable Completion:** When checked, this setting tells the Transaction Service to commit a transaction regardless of whether or not it was completed successfully in the target database application. Allowing unrecoverable transactions is useful for testing deployment scenarios that involve disparate database transactional systems, but is not recommended for production servers because recovery and roll-back operations are not available. This option is unchecked by default.
- **Debug:** When checked, this setting tells the Transaction Service to run with debug information enabled. This option is unchecked by default.

Configuring Web Container properties

By default, the Web container hosts the HTTP and the IIOP (Internet Inter-ORB Protocol) services. The Configure Web Container dialog allows you to modify the XML configuration file for these web services. The dialog presents the elements and attributes of the XML file in an easy to navigate hierarchy. See Modifying the Borland web container IIOP configuration in the *AppServer Developer's Guide* for information about the attributes.

Viewing Partition services information

To view properties and statistics for each of the Partition services, click on the service's icon in the Navigation Pane.

The following sections describe the Partition services properties that you can view using the Management Console.

- Connector Service information
- EJB Container information
- JDataStore Server information
- Naming Service information
- Session Storage Service information
- Transaction Manager information
- Web Container information

Connector Service information

In addition to the General Properties displayed on this tab, described in “General tab”, the following properties appear on the General tab for the Connector Service:

- **Trace level:** Specifies the verbosity of logging to the event log.
- **Use pass by value for calls:** When `true`, allows connectors to pass parameters by value. If not set, all calls must be made by reference.
- **Enable statistics gathering:** If set to `false`, reduces the amount of statistical information gathered by the connector, resulting in improved performance. The default is `true`.
- **Use java serialization (not IIOP):** When `true`, causes connectors to be serialized using Java serialization rather than IIOP serialization.

JCA Summary Statistics

The JCA Summary Statistics tab displays Connector statistics. There are four graphs on this tab that display the Outbound States, Outbound Requests, Inbound States, and Inbound Requests levels over time. For information about enabling and configuring statistics gathering, see “Statistics properties”.

EJB Container information

In addition to the General Properties displayed on this tab, described in “General tab”, the following properties EJB Container properties appear on the Properties tab:

Settings:

- **Trace level:** Specifies the verbosity of logging to the event log.
- **Passivation timeout:** Specifies for Stateful session beans, the amount of time allowed to elapse before the bean's state is persisted.
- **Passivation factory name:** Specifies the name of the JSS service that manages passivation persistence for this container.

Flags:

- **Use pass by value for intra-bean calls:** When `true`, allows EJBs to pass parameters by value when making calls to local beans. If not set, all calls must be made by reference.
- **Enable statistics gathering:** Set to `true` by default. If set to `false`, reduces the amount of statistical information gathered by the container, resulting in improved performance.
- **Use java serialization (not IIOP):** When `true`, causes Stateful session beans to be serialized using Java serialization rather than IIOP serialization.
- **Enable SIDL:** When set, enables EJBs to be accessed using SIDL (Simplified Interface Description Language).

Message-driven Bean Thread Pool:

- **Minimum number of threads:** Specifies the minimum number of message-driven bean dispatch threads allowed to exist in that thread pool.
- **Maximum number of threads:** Specifies the maximum number of message-driven bean dispatch threads allowed to exist in that thread pool.
- **Maximum thread idle time:** Specifies the maximum time a message driven bean thread is allowed to be idle before it is reaped from the pool.

Container Services

The Container Services tab displays statistical data about the main services that the EJB Container provides to the EJBs that it hosts.

There are two graphs on the tab that display the CMP JDBC and Transactions levels over time. The Transactions graph displays the following statistics:

- **Begun Transactions:** Number of transaction begun since last timeslice.
- **Rolledback Transactions:** Number of transactions rolled back since last timeslice.
- **Committed Transactions:** Number of transactions committed since last timeslice.

The CMP JDBC graph displays the following statistics:

- **CMP JDBC Query:** Number of queries since last timeslice.
- **CMP JDBC Update:** Number of updates since last timeslice.

Note

For information about enabling and configuring statistics gathering, see “Statistics properties”.

Bean Requests

The Bean Requests tab displays the aggregate EJB request activity. There are four graphs on the tab that display the number of requests over time for the four types of beans: entity, message-driven (MDB), stateful session (SFSB), and stateless session (SLSB). Each graph displays the number of bean requests of its type to the EJB container since the last timeslice. In addition, the SFSB graph displays the following statistics:

- **Active**: Number of active SFSBs.
- **Activated:** Number of SFSB activated since last timeslice.
- **Passivated:** Number of SFSB passivated since last timeslice. Passivation is controlled by the EJB container's `ejb.sfsb.passivation_timeout` attribute in `partition.xml`. All SFSBs that have not been accessed for this period are passivated.

Bean States

The Bean States tab displays the aggregate EJB states at each timeslice broken down by bean type. There are four graphs on the tab that display the state over time for the four types of beans: entity, message-driven (MDB), stateful session (SFSB), and stateless session (SLSB).

The Entity Bean graph displays the number of Entity Beans in POOLED state and the number of Entity Beans in READY state. The MDB graph displays the number of MDBs in READY state. The SFSB graph displays the number of SLSBs in PASSIVE state and the number of SLSBs in READY state. The SLSB graph displays the number of SLSBs in READY state.

JDataStore Server information

In addition to the General Properties displayed on this tab, described in “General tab”, the following properties appear on the General tab for the JDataStore service:

- **Port:** Sets the port used to listen to JDBC requests.
- **Socket timeout:** Specifies the time a call to accept() in JDS will block until an IO exception is raised.
- **Temporary directory:** Specifies the temporary directory used for all JDataStore connections. If no directory is specified, the current directory is used.

Naming Service information

In addition to the General Properties displayed on this tab, described in “General tab”, the following properties appear on the General tab for the Naming Service:

- **Backing store:** Specifies the type of pluggable backing store to use for storing persistent data. It can be memory, a relational database, or an LDAP directory server. The supported types of backing store adaptor are:
  - InMemory: In-memory adaptor
  - JDBC: JDBC adaptor for relational databases
  - Dx: DataExpress adaptor
  - JNDI: for LDAP only

- **JDBC driver:** Specifies the JDBC driver needed to access the database used as your backing store. The default is the JDataStore JDBC driver. Other supported drivers are Sybase, Oracle, Borland Interbase, and IBM DB2.
- **URL:** Specifies the URL location of the database that you want to access.
- **Login name:** Specifies the login name associated with the database. The default is `VisiNaming`.
- **Password:** Specifies the login password associated with the database. The default is `VisiNaming`.
- **Pool size:** Specifies the number of database connections in the AppServer connection pool when using the JDBC Adaptor as the backing store.
- **JNDI Initial Factory:** This JNDI adaptor property specifies the JNDI initial factory.
- **JNDI Authentication:** This JNDI adaptor property specifies the JNDI authentication type supported by the JNDI backing server.

For more information, see Using the VisiNaming Service in the *VisiBroker for Java Developer*'s Guide.

Session Storage Service information

In addition to the General Properties displayed on this tab, described in "General tab", the following properties appear on the General tab for the Session Service:

- **Factory name:** Specifies the name under which the Session Service registers itself with the Smart Agent. If not specified, Borland AppServer uses a value that is combined from the server name and the Partition name as follows:

  `<server name>/<partition_name>`.

- **Working directory:** Specifies the directory location where the Session Service looks for the JDataStore back end database. The default location is the working Partition directory, for example:

  `<install_dir>/var/servers/<server name>/partitions/<partition name>`.

- **Persistent store:** Specifies the name of the JDataStore file which the Session Service uses as back end database. The default value is `jss_factory`.jds.
- **User name:** Specifies the user name under which Session Service communicates with the JDataStore back end database. The default value is `default-user-name`.
- **Max idle:** Specifies (in seconds) the least amount of time that a session stays present in the back end database without getting accessed. If set to 0 (the default), the session stays in the database indefinitely.
- **Soft commit:** If set to `true` (the default), the Session Service uses the JDataStore back end database with a Soft Commit mode enabled. Setting this property to `true` improves performance of the Session Service, but can lead to recently committed transactions being rolled back after a system crash. For more information, see the JDataStore online documentation at `http://www.borland.com/techpubs/jdatastore`.
- **Debug:** If set to `true`, tells the Session Service to run with debug information enabled. The default is `false`.

For more information, See "Managing and configuring the JSS" in the Java Session Service (JSS) configuration chapter of the *AppServer Developer's Guide*.

Transaction Manager information

In addition to the General Properties displayed on this tab, described in "General tab", the following properties appear on the General tab for the Transaction Service:

- **Allow unrecoverable completion:** When set to `true`, tells the Transaction Service to commit a transaction regardless of whether or not it was completed successfully in the target database application. Allowing unrecoverable transactions is useful for testing deployment scenarios that involve disparate database transactional systems, but is not recommended for production servers because recovery and roll-back operations are not available. The default is `false`.
- **Debug:** When set to `true`, tells the Transaction Service to run with debug information enabled. The default is `false`.

Web Container information

In addition to the General Properties displayed on this tab, described in "General tab", the following properties appear on the General tab for the Web Container service:

- **Enable naming:** When set to `true` (the default), tells the Borland web container service to use the JNDI (Java Naming and Directory Interface) implementation to represent the default JNDI naming context provided to Web applications.
- **Debug:** When set to `true`, tells the Borland web container service to run with debug information enabled. The default is `false`.

Container Statistics and Servlet/JSP Statistics

The Container Statistics and Servlet/JSP Statistics tab displays Web Container statistics. For information about enabling and configuring statistics gathering, see "Statistics properties".

4

Using Borland AppServer Services

The Borland AppServer (AppServer) services are those services available to all applications being hosted on AppServer. The services are:

- Apache 2.2.3 Web Server
- Smart Agent
- JMS services
- 2PC Transaction Service

For each service, the right pane displays tabs containing additional information about the service:

- General: includes display name, name, agent, description, version, vendor. For the 2PC Transaction Service you can also view timeout, sleep and, cache properties. These are described in the section on the 2PC Transaction Service below.
- XML: displays the AppServer service managed object's configuration in the `configuration.xml`.
- Logs: displays a collection of error and status messages associated with a service

These panels are read-only. To configure any information go to the sections below on configuring the services.

Note

Each AppServer Service is a “Managed Object” whose *management* settings are configured on the Advanced properties tab.

Apache 2.2.3 Web Server

The AppServer includes the Apache Web Server version 2.2.3. The Apache Web Server is a robust, commercial grade reference implementation of the HTTP protocol. The Apache Web Server is highly configurable and extensible through the addition of third-party modules. Borland has added an IIOP Connector Module to its implementation of the Apache Web Server. This Connector Module allows Apache and the Tomcat Web container (see below) to communicate via Internet Inter-ORB Protocol (IIOP), allowing users to add the power of CORBA with their Web applications in new ways. For more information about the AppServer Apache implementation, see Web components in the *AppServer Developer's Guide*.

Configuring the Apache 2.2.3 Web Server

To configure the Apache 2.2.3 Web Server:

1. In the Management Console, Hubs View, go to Configurations.
2. Expand the configuration your service is in.
3. Right-click the Apache 2.2.3 node (labeled apache) and choose Properties.

The following are property configuration panels for Apache:

- **General:** This lists the Object type and Object name (these are not configurable). It also lists all of the properties found in the right pane General tab, which are configurable here.
- `httpd.conf:` This is the main Apache server configuration file. It contains the configuration directives that give the server its instructions
- **mime.types:** This file controls which Internet media types are sent to the client for given file extensions, such as application, text, or image files.
- **magic:** This is the magic (hex) data for the `mod_mime_magic` Apache module.
- **Uri Map File:** Entries in this file map URIs to Web Clusters (CORBA instances) configured in the WebClusters.properties file.
- **Web Clusters File:** Entries in this file associate the Web Cluster names with the web containers that service the requests. The web containers in this case are actually CORBA services that implement the `ReqProcessor` idl.
- **Advanced:** This tab allows you to configure Managed Object settings, such as local restart, escalate stop, ping policy, and ping interval. Additionally, you can configure your Management Action settings for start, stop, and kill, which include strategy (“run-apache-process”), ping interval, and timeout.

Smart Agent

The Smart Agent is a distributed directory service provided by the VisiBroker ORB used in AppServer. The Smart Agent provides facilities used by both client programs and object implementations, and must be started on at least one host within the local server network.

Note

If you are using the Web Services Edition, you do not have to use the Smart Agent if you expect your web server and web container to communicate through HTTP or another web protocol.

Configuring the Smart Agent

To configure the Smart Agent:

1. In the Management Console, Hubs View, go to Configurations.
2. Expand the configuration your service is in.
3. Right-click the Smart Agent node (labeled osagent) and choose Properties.

The following are configuration panels for the Smart Agent:

- **General:** This tab lists the Object type and Object name (these are not configurable). It also lists all of the properties found in the right pane General tab, which are configurable here.
- **Settings:** This tab allows you to configure the Smart Agent `${osagent.port}` argument. Additionally, you can configure control options, such as start and stop or monitor the Smart Agent.
- **Address File:** This tab allows you to edit a file specifying IP addresses of remote Smart Agents.
- **Local Address File:** This tab allows you to edit a file specifying the default listening address.
- **Advanced:** This tab allows you to configure Managed Object settings, such as local restart, escalate stop, ping policy, and ping interval. Additionally, you can configure your Management Action settings for start, stop, and kill, which include strategy ("run-process"), ping interval, and timeout.

JMS service

The AppServer provides support for standard JMS pluggability, and currently bundles the Tibco messaging service. To customize JMS provider configurations for Tibco, or other JMS providers see JMS provider pluggability in the *AppServer Developer's Guide*. For additional specific information on Tibco, refer to their documentation in <install_dir>\jms\tibco\doc\html.

Note

The Tibco Admin Console is launched from the Management Console Tools menu.

To configure the JMS Service:

1. In the Management Console, go to Configurations.
2. Expand the configuration your service is in.
3. Right-click the JMS Service node (labeled tibco for Tibco) and choose Properties.

The following are configuration panels for the JMS Service:

- **General:** This lists the Object type and Object name (these are not configurable). It also lists all of the properties found in the right pane General tab, which are configurable here.
- **Settings:** This tab allows you to configure JMS settings such as the JMS home directory and the JMS server URL.
- **tibjmsd.conf:** This tab displays the Tibco JMS daemon, which lists all of the configurable Tibco processes. Parameters are in name = value syntax.
- **users.conf:** This tab allows you to edit the users.conf file that defines all users, such as `<user-name>:[password]:<description>`. It should be noted that the password is *always* and only set by the system, and must NOT be entered manually in the text editor. When adding a user in the text editor you should always leave the password empty.

- **groups.conf:** This tab allows you to edit the groups.conf file that defines all groups, such as:

```
<group-name1>:[<description>]
          <user-name1>
    <user-name2>
       ...
    <user-nameN>
```

- **topics.conf:** This tab allows you to edit the topics.conf file that defines all topics, such as `<topic-name> [<property1,property2,...>]` where the properties are variables listed in the file.
- **queues.conf:** This tab allows you to edit the queues.conf file that defines all queues, such as `<queue-name> [<property1,property2,...>]` where the properties are variables listed in the file.
- **ad.conf:** This tab allows you to edit the ad.conf file that lists all the permissions on topics and/or queues for all users or groups, such as:

```
TOPIC=<topic> USER=<user>   PERM=<permissions>
TOPIC=<topic> GROUP=<group> PERM=<permissions>

QUEUE=<queue> USER=<user>   PERM=<permissions>
QUEUE=<queue> GROUP=<group> PERM=<permissions>
```

- **factories.conf:** This tab allows you to edit the factories.conf file that defines the connection factories for JNDI, such as:

```
[<factory-name>]
type=topic|queue|generic|xatopic|xaqueue|xageneric
url=<url-string>
clientID=<client-id>

<ssl-prop1=value>
<ssl-prop2=value>
  ...
<ssl-propN=value>
```

Note

SSL properties are optional.

- **routes.conf:** This tab allows you to edit the routes.conf file that defines routes between the AppServer Tibco Enterprise for JMS server and other Tibco Enterprise for JMS servers, such as:

```
 [<route-name>]
# url=<url-string>
#
# <ssl-prop1=value>
# <ssl-prop2=value>
```

- **bridges.conf:** This tab allows you to edit the bridges.conf file that defines Bridges for destinations. Bridges can bridge (resend) messages sent into source destinations to target destinations. This works like an application sending messages directly to the target destinations
- **transports.conf:** This tab allows you to edit the transports.conf file that defines configurations for RV and RVCM transports.

Note

The transports defined in this file are only activated if `tibjmsd.conf` contains the `tibrv_transports=enabled` parameter.

– **tibrvcm.conf:** This tab allows you to edit the tibrvcm.conf file that contains the Anticipated Tibco RVCM (listener program) listener configuration, such as `[<transport-name>] <listener-name> <rv cm subject>`.

– **Advanced:** This tab allows you to configure Managed Object settings, such as local restart, escalate stop, ping policy, and ping interval. Additionally, you can configure your Management Action settings for start, stop, and kill, which include strategy ("run-process"), ping interval, and timeout.

2PC Transaction Service

The AppServer includes a two-phase commit transaction management service called the VisiTransact Transaction Manager (which is based on the CORBA OTS specification). For more information on transactions and VisiTransact, see Transaction management in the *AppServer Developer's Guide*.

In addition to the General Properties, the following 2PC Transaction Service Properties appear on the General tab for the 2PC Transaction Service:

– **Name:** Specifies the name of the Transaction Service instance used when registering its interface with the Smart Agent. The default is `<server_name>_ots`.

– **Default transaction timeout:** Sets the default transaction timeout value (in seconds) for the selected Transaction Service instance. If not set, the default is 600 seconds.

– **Default max transaction timeout:** Sets the maximum allowed transaction timeout value (in seconds) for the selected Transaction Service instance. If not set, the default is 3600 seconds.

– **Log Sleep:** Specifies how long a transaction waits (in milliseconds) before returning from a synchronous log write operation. The default value is 0 (zero). For multi-threaded environments with less than five concurrent transactions, this default value is recommended. For multi-threaded environments with five or more concurrent transactions, a value of 10 milliseconds is recommended. If the number of concurrent transactions is considerably higher, a value greater than 10 milliseconds should be considered. Higher values increase throughput in high concurrency situations at the expense of individual thread latency.

– **Log Cache:** Specifies the size of the log cache (in kilobytes). The default is 64 KB. If the cache size is too small, an increased number of disk flushes are forced. For most environments, the default value is suitable.

– **Purge old log transaction records:** Determines if transaction records will be purged from the log. A value of `false` (the default) specifies that existing transaction records will not be purged from the log. A value of `true` specifies that existing transaction records from previous invocations of the transaction manager will be purged from the log. When set to `true`, all live records for the previous run of this instance of the Transaction Service are forgotten (such as records for transactions that are pending completion). Additionally, setting this property to `true` enables you to change the directory location of the Transaction Service log when you restart the Transaction Service.

To configure the 2PC Transaction Service:

1 In the Management Console, Hubs View, go to Configurations.

2 Expand the configuration your service is in.

3 Right-click the Transaction Service (2PC) node (labeled ots) and choose Properties.

The following are configuration panels for the Transaction Service:

– **General:** This lists the Object type and Object name (these are not configurable), as well as display name, agent name, data directory, version, vendor, and description.

- **Settings:** This tab allows you to configure the factory name, default transaction timeout, default max transaction timeout, log directory, log sleep, log cache, and set the purge old transaction records properties.
- **Advanced:** This tab allows you to configure Managed Object settings, such as local restart, escalate stop, ping policy, and ping interval. Additionally, you can configure your Management Action settings for start, stop, and kill, which include strategy (“run-process”), ping interval, and timeout.

5

Viewing J2EE component configurations

This section describes J2EE components and how they are viewed in the Management Console.

Viewing archive attributes

Deployed modules possess a variety of attributes that you can view in the Management Console. These attributes are shown in the General, Details, References, and XML tabs. Different modules may have different properties displayed, depending on what type of archives they are.

To view deployed module attributes:

1. In the Management Console, Hubs View, go to Configuration
2. Expand a partition within a configuration and select a member of the Deployed Modules group. Information about the deployed module is displayed in tabs in the Content pane of the Management Console.

General tab

The General tab lists a module's properties: module name, display name, module type (such as web or application), J2EE version, module version, and a description.

Module Storage and Mobility Settings pertain to hosting an archive pathed outside of a Partition (Hosted from server path) and hosting an exploded archive on a Partition (Stored as exploded archive). See “Deploying modules and libraries to a Partition” for more information.

– **Stored as exploded archive:** Typically, you cannot deploy an exploded (or “expanded”) archive to a Partition. This feature allows a Partition to host an exploded archive in a directory local to the server. This directory must be visible from the server's file system. As with explicitly-pathed archives, exploded archives must be deploy-ready. That is, all necessary stubs and skeletons need to have been generated before they are deployed. Exploded archives can only be explicitly-pathed archives, and are deployed to the Hosted Modules folder in the Partition's tree. The “Stored as exploded archive” box is checked when indicating that a specific archive being hosted by the Partition is expanded.

  A typical use case for hosting exploded archives is for WARs, enabling the application to modify itself while it is running. Any added or modified JSPs are recognized and dynamically hosted by the Partition in the Hosted Modules folder.

– **Hosted from server path:** This feature allows a Partition to host any archives in directories local to the server, known as *explicitly-pathed archives*. That is, these archives must be visible from the server's file system (they do no have to be on the `localhost`). These archives are not “deployed” to the Partition. Rather, the Partition must be explicitly told to host the archive. Explicitly-pathed archives are located in the Hosted Modules folder in the Partition's tree.

  Hosted archives must be deploy-ready with all stubs and skeletons already produced. Normally, stubs and skeletons are generated at deployment. Since these archives are hosted, you must use the Stub Generation Wizard on the archive you wish to host. Once the necessary stubs and skeletons are generated, you can add the archive to the Partition.

  If modules are deployed to the Partition's repository, the Deployed option is selected.

Additional Web Module Properties section (WARs)

Note that WARs in the Deployed Modules folder display additional Web module properties on the General tab. The context root and any URLs for a selected WAR appear in the Additional Web Module Properties section. You can select a URL in the URLs window and click the Test URL button to test the URL address.

Details tab

The Details tab displays the contents of an archive: child modules (JARs, WARs, etc.) and meta-inf., or the Manifest (XML files). Clicking on the column headers allows you to organize by name, type, date, and so on. Double-clicking on any of the contents displays file contents.

References tab

The References tab is a read-only display available for certain deployed modules, wherein all EJB references are graphically displayed and can be viewed in a variety of ways using the tools at the top of the tab. This tab is also in the Deployment Descriptor Editor (DDEditor).

XML tab

When you open the XML tab, a second pane opens under the navigation tree which allows you to click on either descriptor file whose contents will appear in the content pane. The XML tab displays (read-only) the module's XML deployment descriptors which may include a standard descriptor (web.xml for example) and a Borland-specific descriptor (web-borland.xml for example). For more information on standard and vendor-specific deployment descriptors, see “Using the Deployment Descriptor Editor.”

Note

The XML is NOT editable in this view. You can edit descriptors using the DDEditor. However, you can go to, copy, and paste by right-clicking in the XML content pane.

XML tab for library modules

The XML tab for library modules is relevant to timed Partition operations such as initialization and cleanup, or Partition Lifecycle Interceptors. It displays the XML necessary for library modules to register classes to respond or interact with specific function calls associated with Partition lifecycle events. Since only lib modules are allowed to host such code, users declare a class name and parameters (to be passed to the function for the event) in the XML file for lib modules. For specific information on XML for library modules and Partition Lifecycle Interceptors, refer to Implementing Partition Interceptors in the *AppServer Developer's Guide*.

Axis Properties tab

The Axis Properties tab shows the elements (services, handlers, type mappings, transports, chains, and global configurations) of the Axis service. For more information see “Viewing Web Services WSDD properties”.

Summary Statistics tab

A Summary Statistics tab is available in the Management Console to view statistics on your deployment.

Removing modules from a Partition

To remove deployed modules from a Partition:

1. Select the node of the deployed module you want to remove in the Management Console.
2. Right-click the module and choose Remove from the Context menu.

   Note

   Removing permanently removes modules from the Partition.

About Enterprise Archives (EARs)

The Management Console enables you to view and manage enterprise application archives, which are packaged as Enterprise Archives (EARs). EARs are deployable units and represent an application in the server. EARs are packaged in a similar way to JAR files and contain all of the modules and deployment descriptors that constitute the application.

Viewing EARs

To view attributes for deployed EARs:

1. Open the Management Console.
2. Select the Configuration that contains your deployed modules.
3. Expand the node that represents the Partition in the navigation tree.
4. Expand the Deployed Modules node and select the EAR that you want to view.

The EAR includes any JAR or WAR files that may be associated with it and can have multiple JARs and WARs.

1. To view the EAR attributes, select the specific EAR from the Deployed Modules folder node.
2. Select the Details, XML, or References tab to view the EAR's attributes.
3. To open a file in the work pane, double-click the file.

Specifying the ordering scheme for EARs

If you have multiple EARs that you would like to load in a particular sequence, you must modify the `partition.xml` file to specify that order. For example, if there are five ears—a.ear, b.ear, c.ear, d.ear and e.ear, and if you want to ensure that e is loaded before d, d before c, and so on, you must modify the `partition.xml` file under the relevant partition/properties folder as follows:

```
<archives ear.repository.path="ears" war.repository.path="wars"
ejbjar.repository.path="ejb_jars" dar.repository.path="dars"
rar.repository.path="rars" lib.repository.path="lib"
classes.repository.path="classes">
    <archive name="e.ear" disable="false" order=”1”/>
    <archive name="d.ear" disable="false" order=”2”/>
    <archive name="c.ear" disable="false" order=”3”/>
    <archive name="b.ear" disable="false" order”=4”/>
    <archive name="a.ear" disable="false" order=”5”/>
</archives>
```

The above EAR(s) will be loaded in the order : e, d, c, b, and then a.

All EAR's for which no order is specified will be loaded in the end (alphabetically). If two ears have the same order then they will loaded in alphabetical order. This scheme is only implemented for EARs. So if you want other archives to have a load order, they have to put them in ears and then modify `partition.xml` accordingly.

About web archives

Web archives (WARs) can be viewed and managed in the Management Console. WARs can be, but are not necessarily, encapsulated within an EAR and represent Web objects exclusively. Typically, they contain the following items:

- Java class files and helper classes for any servlets
- JSP pages and their helper classes
- Static HTML documents
- Applets and their class files
- XML deployment descriptors: a web.xml file and a Borland-specific web-borland.xml file

Viewing WARs

To view attributes for deployed WARs:

1. Open the Management Console.
2. Select the Configuration that contains your deployed modules.
3. Expand the node that represents the Partition in the navigation tree.
4. Expand the Deployed Modules node and select the WAR that you want to view.

You can view the attributes of the WAR in the Details, References, XML, and Summary Statistics tabs. For information about creating a WAR for deployment, see “Using the Deployment Descriptor Editor.”

Viewing Web Services WSDD properties

Using the Management Console, you can view the properties of any Web Service deployment descriptor (WSDD) that is packaged in a WAR file.

To view the `server-config.wsdd` file from the Management Console:

1. Open the Management Console.
2. Select the Configuration that contains your deployed modules.
3. Expand the node that represents the Partition in the navigation tree.
4. Expand the Deployed Modules node and select the WAR with the Web Service that you want to view.
5. At the bottom of the content Pane, click the Axis Properties tab. The property elements of the `server-config.wsdd` file are mapped into a read-only layout.
6. From the Elements drop-down list, choose the type of element you want to view.

When you select an element, any associated parameters are automatically displayed in the adjacent table. You can drill-down to view the Parameters of Element details.

About libraries

The libraries in AppServer contain classes such as JDBC drivers that can be shared amongst modules and are stored as JAR or ZIP files. When the libraries are used by modules they are placed on the classpath of the server.

For example, if a module requires a set of drivers to access a database, those drivers can be included in a JAR, then deployed to “Installed Libraries”. From here, they can be accessed by all modules that need to access that database.

Viewing libraries

To view an installed library in the Management Console:

1. Open the Management Console.
2. Select the Configuration that contains your deployed library.
3. Expand the node that represents the Partition in the navigation tree.
4. Expand the Deployed Modules node and select the library that you want to view.

Viewing library attributes

Libraries possess a variety of attributes that you can view. These attributes include the library name, type, date size, percent (compression percentage), packed (compressed) size and path values.

To view library attributes, select the library you want to view. The library attributes will appear in the right pane.

Note

For more information about deploying libraries, see Connecting to Resources with Borland AppServer: using the Definitions Archive (DAR) in the *AppServer Developer's Guide*.

About resource adapters

Resource Adapters provide you with a standard and reliable means of accessing data in third-party Enterprise Information Systems (EIS). Resource Adapters are packaged as Resource Adapter Archives (RARs), deployed to Borland AppServer (AppServer). They enable transactional access to the EIS from servlets, JSP pages, enterprise beans, J2EE Application clients and CORBA clients. RARs use the XA_TRANSACTION mode as specified in Sun's J2EE 1.3 specifications.

Viewing resource adapters

To view attributes for deployed RARs:

1. Open the Management Console.
2. Select the Configuration that contains your deployed modules.
3. Expand the node that represents the Partition in the navigation tree.
4. Expand the Deployed Modules node and select the RAR (which may also be in an EAR file) that you want to view.

Viewing Resource Adapter information

You can view the following types of information associated with RARs:

- **Attributes:** RARs possess a variety of attributes that you can view. These attributes are shown on the Details tab, which include RAR properties, values, and details of classes in the RAR.
- **State panel:** Shows connection factory instance counts (can be viewed as pie chart, line chart or in a table), connection factory state diagram, and connection factory statistic properties (editable).

To view the RAR attributes:

1. Select the RAR.
2. Select the type of attribute that you want to view in the work pane.

To view the RAR State panel:

1. Select the RAR.
2. Select the factory icon for the RAR.

Viewing JAR information

Java archives (JARs) can be viewed and managed in the Management Console. JARs can be modules within a WAR or an EAR. EJBs are encapsulated and viewed within a JAR file.

Typically, JARs contain:

- Java class files
- Standard XML files
- Borland-specific or vendor-specific XML files

Viewing JARs

To view attributes for deployed JARs:

1. Open the Management Console.
2. Select the Configuration that contains your deployed modules.
3. Expand the node that represents the Partition in the navigation tree.
4. Expand the Deployed Modules node and select the JAR that you want to view (may be in a WAR or EAR file).

You can view the attributes of the JAR in the Details, XML, or References tabs. For information about creating a JAR for deployment, see “Using the Deployment Descriptor Editor,”

Viewing EJB information

EJBs can be viewed and managed in the Management Console. EJBs are encapsulated and viewed within a JAR file. JARs can be modules within a WAR or EAR.

You can view the following types of information associated with EJBs:

- **State panel:** Shows bean instance counts, bean state diagram, and bean statistic properties.

Enterprise bean types

The Borland EJB container supports three types of enterprise beans:

- **Stateless Session beans:** In these beans, any state (if required) is maintained by the client or in an external location such as a database. Because no state is maintained, stateless Session beans aren't tied to any specific client. Therefore, any available instance of a stateless Session bean can be used to service a client.
- **Stateful Session beans:** In these beans, state is maintained by the enterprise bean. This means the application server manages client-bean pairs. Each instance of the enterprise bean is created on behalf of a client and is intended to be a private resource to that client. Stateful Session beans can access persistent resources (such as databases and files), but unlike Entity beans, they don't actually represent the data.
- **Entity beans:** These beans are persistent objects and represent an object view of data stored in permanent storage. An Entity bean lives in an EJB container in much the same way a record lives in a database. Unlike Stateful Session beans, Entity beans can be accessed by multiple clients concurrently. This concurrency is managed by the EJB container.

You can view and edit the following beans' attributes in the Deployment Descriptor Editor (DDEditor):

- Bean Name
- Bean Type
- Home Name
- Remote Name
- Local
- Local Home
- JNDI Home
- JNDI Local Home
- Bean Class
- Session Type
- Transaction Type
- Storage Timeout

Hosted Modules

Hosted modules are modules hosted by the Partition that are not in the Partition's home directory `<install root>`. These modules can be in any file local to the server, that is, the file must be visible from the server's file system (they do no have to be on the `localhost`). When loaded, hosted modules appear in the Hosted Modules folder.

Hosted modules must be deploy-ready with all stubs and skeletons already generated. Normally, stubs and skeletons are generated at deployment. You can use the Stub Generation Wizard on the module you wish to host. Once the necessary stubs and skeletons are generated, you can add the module to the Hosted Modules folder.

Exploded Archives

Exploded archives are archives in an “unzipped” state. Typically, you cannot deploy an exploded (or “expanded”) archive to a Partition. The Host Additional Module feature allows a Partition to host an exploded archive in a directory local to the server, that is, the directory must be visible from the server's file system.

As with hosted modules, exploded archives must be deploy-ready. All necessary stubs and skeletons must be generated before they are deployed. Exploded archives can only be hosted modules, and are deployed in the Hosted Modules folder.

A typical use case for hosting exploded archives is for WARs, enabling an application to modify itself while it is running. Any added or modified JSPs are recognized and dynamically hosted by the Partition in the Hosted Modules folder.

To host an additional module or expanded archive in Hosted Modules:

1. Right-click the Partition or Hosted Modules node and select “Host additional module”.
2. Select:
   - **Select File** for a hosted module. Browse to the file where the module resides.
   - **Select Directory** for an exploded archive. Browse to the directory where the exploded archive resides.
1. Enter a name for the module if you wish to rename it. Leave it blank for the default name.
2. Click OK.

About EJB Containers

The AppServer provides integrated EJB container services which are hosted within Partitions. In addition to standard container services, the EJB container provides a range of transaction and persistence services and access to the J2EE service and communication APIs. The following section describes persistence support for containers.

Persistence support

Modeled after version 2.1 of the EJB specification published by Sun Microsystems, the AppServer EJB container supports built-in persistence. This persistence can be bean-managed or container-managed.

For bean-managed persistence, the developer overrides the `ejbLoad()` and `ejbStore()` methods and provides custom code (for example, using JDBC) to transfer the data from the object's fields to permanent storage and vice versa. Bean-managed persistence ties the enterprise bean to one method of persistence and requires the developer to write and maintain persistence code.

For container-managed persistence, Borland supplies mapping tools to map the Entity bean's field to fields in JDataStore. The Entity bean can be persisted into different data sources by remapping the fields in the deployment descriptor.

For more information on persistence support, see Entity Beans and CMP 1.1 in Borland AppServer in the *AppServer Developer's Guide*.

Note

Any JAR or Application (EAR) files present under Deployed Modules represent contained EJBs. The “EJB Container” that appears as one of the Partition services is a representation of the services provided by an EJB container. These services include Container-Managed Persistence (CMP) for database storage, a transaction service, a naming service for referencing objects, a Web container, and other services.

An EJB server can have multiple partitions. Partitions are defined and configured as needed to deploy objects on the network. You can add customized partitions for EJB containers for various databases (such as Oracle) and support them separately from EJB development.

Viewing EJB container services attributes

The EJB container services include a variety of attributes you can view and edit. Generally, to view the container services, select the EJB Container in the Partition which you are using. To edit the services' properties, right click on the service and select Properties. The services' attributes that are viewable are on the General, Properties, Container Services, Bean Requests, and Bean States tabs.

Note

This section includes information about viewing container services' attributes. For information about EJB attributes, see “Viewing EJB information”.

To view EJB container services attributes:

1. Open the Management Console.
2. Select the Configuration that contains your EJB Container.
3. Expand the node that represents the Partition in the navigation tree.
4. Click the EJB Container node that you want to view.

About web containers

The Web container is designed to support development and deployment of web applications. This version of AppServer provides Tomcat 5.5.17 as its web container. Web container properties are located in the General, Container Statistics and Servlet/JSP Statistics tabs.

Viewing the Web container

To view the Web container:

1. Open the Borland Management Console.
2. Select the Configuration that contains the web container.
3. Expand the node that represents the Partition in the navigation tree.
4. Expand the Deployed Modules.
5. Select Web Container.

6

The Installations view

The Installations View allows you to view and configure Borland Deployment Platform components deployed on the local host running the Console. You can use the Installations View to:

- view and edit the local installation's property and configuration files
- view archives deployed to Borland AppServers (AppServer) installed on the host
- view and edit security profiles stored on the local host
- view and edit XML templates for Partitions, Configurations, and Managed Objects
- view information about the locally installed agent, if any.

Warning

Borland recommends using the Management Console's Hubs View to configure your system. The Installations View can only provide options for Borland AppServer installations on the host local to the running Console. If you do choose to use the Installations View for this purpose, be sure to shut down the server before editing any of its configuration files, property files, or other services. Otherwise, your changes may be lost and undesirable server behavior may occur.

The Installations view contains the following areas:

- A view of all installations of Borland AppServers. The installation who's console is running is indicated with a red dot on the installations folder.
- An expandable and collapsible tree browser that shows the servers, services, partitions, and modules installed on the same machine as the Borland Management Console. To view or modify an object in the tree, navigate to it by expanding its icon (node). If the object in the tree contains other objects, you can also double-click the icons to expand the view.
- The Borland Management Console opens a pane beneath the tree when it discovers associated configuration, property, security configuration, or log files for an object selected in the tree.
- The window on the right displays and the contents of a selected file or tree node. In some instances, dialogs allowing you to configure the selected component are displayed. For property and configuration files, and editing window displays the contents of the file which you may alter and save.

Working with Property and Configuration Files

If a component selected in the tree has property and/or configuration files associated with it, appropriate tabs appear in the content pane: Property Files, and Configuration Files. To view these files:

1. Select the node in the tree whose files you wish to view. If a node has configurable property or configuration files, the “Property Files” and/or “Configuration Files” tabs will be available in the Content pane.
2. Click the appropriate Tab.
3. The structure pane displays a list of the configuration or property files associated with the component.
4. Select a file from the list in the structure pane. Its contents appear in the content pane. By default, the contents of the first file in the list are displayed when either the Configuration Files or Property Files tab is selected.
5. Make any changes to the files in the editing window. Be sure to read the Borland-provided comments instructing in the proper use of the file. These comments appear in green text. Editable portions of the files are in black text. Changes you make appear in blue text.
6. When you are finished, click the “Apply Changes” icon at the top left of the content pane's toolbar.

You can also edit the launcher configuration files for the various Borland Deployment Platform tools. (For example, you can edit the configuration file for the Tibco JMS service, or the Deployment Descriptor Editor.) To edit any of these files:

1. In the navigation pane, expand the root node of your installation. (For example, `C:/BDP`.)
2. Select the “Tool Configuration” node.
3. The structure pane displays a list of the available tools you can configure.
4. Select a file from the list in the structure pane. Its contents appear in the content pane. By default, the contents of the first file in the list are displayed when the “Tool Configuration” node is first selected.
5. Make any changes to the files in the editing window. Be sure to read the comments instructing in the proper use of the file. These comments appear in green text. Editable portions of the files are in black text. Changes you make appear in blue text.
6. When you are finished, click the “Apply Changes” icon at the top left of the content pane's toolbar.

Working with Local Archives

The Installations View allows you to examine the contents of archives deployed on the host local to the running console. The “Archives” node expands to show all hosted archives by type. You can also perform some operations on specific archives.

Viewing Archive Data

To view archive information:

1. In the navigation pane, expand the root node of your installation. (For example, `C:/BDP`.)
2. Click the Archives node. The content pane displays information about the hosted archives. You can view an archive's name, type, size, and the path to its location.
3. Expand the Archives node to display additional folders representing the types of archives recognized by the Borland Deployment Platform.
4. Drill down by either expanding the tree (recommended) or by double-clicking the folders that appear in the content pane.
5. Select an individual archive file to view information about that archive. The content pane displays tabs for the various archive characteristics you can view. These are:
   - General tab: displays module information, such as its name, type, and Java version.
   - Details tab: displays the contents of the archive, including class files.
   - References tab: graphically displays references between beans and other components within the archive.
   - XML tab: displays the archive's deployment descriptor.
1. Note that you cannot edit archive information. It is read-only.

Performing Archive Operations

You can also perform some operations on an individual archive by right-clicking its node in the tree and selecting a choice from the context menu. The options are:

- Remove: remove an archive from its hosting Partition.
- Redeploy to another Partition: remove the archive from its current Partition and redeploy it to another of your choosing.
- Download copy of module: store a copy of the module in another location. This option does not remove the module from its hosting environment.
- Edit deployment descriptor: launches the Deployment Descriptor Editor and loads the module's deployment descriptors for editing.
- Edit with assembly tool: launches the Application Archive Assembly Tool for making changes to the structure the archive itself.
- Run: start the module.

Working with Security Profiles

You can view the status of Security Profiles stored on the console's host. To view Security Profile status:

1. In the navigation pane, expand the root node of your installation. (For example, `C:/BDP`.)
2. Expand the Security Profiles Node. Available profiles are displayed as child nodes.
3. Click the profile who's status you want to check. Its general status and SSL status are displayed in the content pane.

You can configure an individual security profile by right-clicking its node and selecting Configure from the context menu. For information on configuring security profiles, see Configuring Security Profiles for Domains in the *Borland Security Guide.*

Working with Templates

Borland AppServer uses XML to describe the behavior of the system components it manages. You can view this XML, as well as the XML for an entire set of components working in concert—a “configuration”. AppServer uses templates to describe the default behavior of common components. You can edit these templates to suit your own needs, and then create managed objects and/or configurations using your customized templates.

Note

Templates you customize are stored only in the local repository. However, they are only applied using the Console's Hubs View. In order to use your customized templates, you must use a console running on a host local to them.

Partitions Templates

You can view the contents of Partitions residing on the local host and see what modules are deployed to them. To view and edit Partition template information:

1. In the navigation pane, expand the root node of your installation. (For example, `C:/BDP`.)
2. Expand the Templates node. Three child nodes appear: Configurations, Managed Objects, and Partitions.
3. Expand the Partitions node. Available Partition templates are displayed in the tree.
4. Click on an individual Partition to view its contents.
5. To edit its template, locate the `template.xml` file in the Contents pane and double-click it. An editing window appears.
6. Make your changes and click “Save” when finished.

Configuration Templates

To view and edit the XML template for a configuration:

1. In the navigation pane, expand the root node of your installation. (For example, `C:/BDP`.)
2. Expand the Templates node. Three child nodes appear: Configurations, Managed Objects, and Partitions.
3. Expand the Configurations node. Child nodes appear representing various types of configurations, as well as a node for the “blank” template.
4. Expand the child nodes and locate the template you want to edit, or choose the “blank” template.
5. Select the individual template. Its XML appears in the content pane.
6. Make your changes in the content pane.
7. When you are finished, click the “Apply Changes” icon on the top-left of the content pane's toolbar.

Managed Object Templates

To view and edit the XML template for an individual managed object:

1. In the navigation pane, expand the root node of your installation. (For example, `C:/BDP`.)
2. Expand the Templates node. Three child nodes appear: Configurations, Managed Objects, and Partitions.
3. Expand the Managed Objects node. Child nodes appear representing various types of Managed Objects.
4. Expand the child nodes and locate the template you want to edit.
5. Select the individual template. Its XML appears in the content pane.
6. Make your changes in the content pane.
7. When you are finished, click the “Apply Changes” icon on the top-left of the content pane's toolbar.

7

Using the Management Console wizards

This section describes how to use the Management Console wizards that guide you through such actions as stub generation, deployment of modules, merging of modules and libraries, JAR compression and patch application, verification, and migration of J2EE modules.

The Management Console makes the following wizards available in the Wizards menu:

- Deployment Wizard
- Merge Wizard
- Verify Wizard
- XML Migration Wizard
- Stub Generation Wizard
- Remove Stubs Wizard
- Apply Patch Wizard
- JAR Wizard
- Configure Agents Wizard

In addition to the wizards available from the Wizards menu, the Management Console also offers the following wizards:

- Registration Wizard (see “Borland product registration wizard” in the *AppServer Installation Guide)*
- Export EJB as a Web Service

Deployment Wizard

The Deployment Wizard guides you through the process of deploying modules. In addition to deploying J2EE modules to a Partition, you can create a JAR file suitable for manual deployment. This wizard generates JAR files containing all of the necessary stubs for a manual deployment to a server, which is useful if you want to create a deployable JAR, but do not want to deploy it right away.

Before you begin deployment, you need to supply the runtime data (such as the JNDI name of an EJB). This is stored in a Borland-specific XML file, the name of which is dependent upon the module type. For example, for EARs the Borland XML file is called `application-borland.xml`. This file needs to be created and stored within the J2EE module you want to deploy. The easiest way to do this is to open the J2EE module using the DDEditor and fill in the missing XML information. See "Using the Deployment Descriptor Editor" for more information.

To add one or more J2EE modules to a server:

1. From the Wizards menu on the Management Console, select Deployment Wizard.
2. In Step 1 of the Deployment Wizard, click Add to browse the directory tree and locate the modules you want to deploy to a container.
3. In the Add J2EE Module dialog window, select the modules you want to add, then click OK. You can add as many modules as you want. Once the files have been added, the wizard displays an expandable tree browser that shows the module selections and their contents. (To remove a module, select its node icon in the tree and click Remove.)
4. Specify additional options:
   - **Each module is independent (Don't share any generated stubs).** Checked by default, this option allows you to generate stubs for each module independently of any other modules included in the deployment.
   - **Restart Partitions on deploy (cold deploy).** Not checked by default, use this option to deploy library JARs or RARs and add them to the server's classpath, when the server needs to be restarted. Check this option if you want this action to occur as part of the deployment process.
   - **Verify deployment descriptors.** Checked by default, this option runs a verification tool which checks that supplied deployment descriptors for the added modules are correct.
   - **Generate stubs.** Checked by default, this option runs the stub generator. Stub generation is required so that modules are in a form that the server is expecting. For EJBs, this involves generating client and server-side stubs.
1. If you want to supply additional information to generate stubs or verify the deployment descriptors, click Advanced Options to open the Advanced Deployment Options dialog box:
   - The Stub Generator tab contains options to add classpath information and command line arguments for the java2iiop and javac compilers.

     Click Edit to open the Classpath Editor where you can add an archive or specify the location of any classpath dependencies. Use the Java2IIOP arguments and Javac arguments fields to specify any command line arguments (click More Info to display Java2IIOP usage information).

     Stub generation is required so that modules are in a form that the server is expecting. For EJBs, this process involves generating client and server-side stubs. Since stubs in the server are generated from the Management Console and not on the server, any classes that are needed during stub generation (that is, compile time dependencies of the Home and Remote interfaces of the beans within the EJB) will need to be available to the Management Console. By default, if you are deploying multiple modules, then all modules are placed on the classpath of the stub generator. So, if your beans have references to beans in

other JARs, a multiple deployment requires that you add the other JARs to the classpath. In addition, you can enter any arguments that may be needed by the file. Click the More Info button for a detailed usage statement and consult Programmer tools for Java in the *VisiBroker for Java Developer's Guide* for information about the Java2IIOP tool.

Note

If you add a classpath here, remember that you must add the classpath on the server-side too.

- The Verifier tab contains options to choose whether or not to verify that the deployment descriptors for the added modules are correct and, if so, to show all warnings (in addition to errors) and use pedantic or strict verification procedures to produce warning messages related to the compliance of your J2EE modules with J2EE specifications.

  Verification helps you find exactly what information you need to specify.

1. On Step 2 of the Deployment Wizard, select the Partitions to which you will deploy the modules. Click Refresh List if the available Partitions on the server have changed recently.
2. Click Finish to tell the wizard to complete the deployment process. The wizard analyzes the files and displays the results in the Deploying Modules window. The modules are checked for validity, and any errors or warnings encountered are shown. If the Partition you are deploying to fails to load the module, errors will display the Partition's Log tab.

If one or more files have errors, you can take any of the following actions:

- Stop the Deployment Wizard and use the DDEditor to fix the problem. For more information, see "Using the Deployment Descriptor Editor."
- Use an IDE tool to fix the problem.

Note

Prior to, or after a module is deployed, you can use the Deployment Descriptor Editor to change deployment information as desired. For example, you can use the DDEditor to specify or change the Transaction Policies and Security Roles for an enterprise bean.

Merge Wizard

The Merge Wizard produces JARs and EARs by merging your components and modules. The wizard analyzes your EJB JAR and EAR files and attempts to determine the minimal set of classes required by clients to access them. A JAR or EAR file containing the minimal set of classes is then produced. In addition, the wizard will produce a library file for your selected files.

To assemble an application, the deployment descriptors must be edited for the modules to link dependencies between components in different archives. All dependencies must be linked before deployment. For example, the description of EJBs in a WAR file should match with their description in the EJB JAR file.

If you are merging EJBs and want the wizard to try to match JNDI references with bean JNDI names in the deployment descriptor, select Convert EJB JNDI references to links where possible.

In addition, if the JAR files contain EJB 1.1 deployment descriptors, they will be consolidated into a single deployment descriptor.

The Merge Wizard lets you:

- Merge modules to produce a 1.2 EAR
- Merge modules to produce a 1.3 EAR

- Merge libraries

To merge modules to produce an EJB client JAR files and J2EE EAR files:

1. From the Wizards menu in the Management Console, select Merge Wizard.
2. Select the type of module that you want the wizard to produce and the J2EE version of the resulting file, then click Next.
3. Click Add to browse the directory tree and select the JAR or EAR files. You can add as many files as you want.
4. Select the modules that you to merge, then click OK. (If you want to remove one or more files, highlight the file or files, then click Remove.)
5. Click Next.
6. Designate an output file from the merge process, either by entering a file name or browsing to a directory using the Browse button.
7. Click Finish.

Verify Wizard

The Verify Wizard lets you check an archive file for correctness and consistency, and lets you check that all the elements required for deploying your application are in place. After verifying archives individually and ensuring there are no errors, the assembly level verification involves verifying other resources that will be built into an application. For example, the Verify Wizard will verify the existence and correctness of URIs (Uniform Resource Identifiers), but not EJB or JNDI links.

To verify an archive:

1. From the Wizards menu on the Management Console, select Verify Wizard.
2. Click Add to browse the directory tree and select the modules you want to verify. You can add as many modules as you want.
3. Select the modules that you to verify, then click OK. (If you want to remove one or more files, highlight the file or files, then click Remove.)
4. Choose the verification role level:
   - **Developer:** This is the lowest verification level. All XML is checked for syntax as well as standard and proprietary keywords relevant to the current archive type. Consistency across the archive is checked, but no external resources are verified at this level.
   - **Assembler:** Once the archives are individually verified and are correct, other resources built into an application will start to be verified. For example, this level will verify the existence and correctness of URIs, but not EJB or JNDI links.
   - **Deployer:** (the default) All checks are turned on. EJB and JNDI links are checked at this level as well as the operational environment in which the application is to be deployed.
1. Specify additional options, if desired:
   - **Classpath.** You can indicate classpath information. Click Edit to open the Classpath Editor where you can add an archive or specify the location of any classpath dependencies.
   - **Show all warnings and Use strict (pedantic) checks.** Shows all warnings (in addition to errors) and uses pedantic or strict verification procedures to produce warning messages related to the compliance of your J2EE modules with J2EE specifications.
1. click Finish to begin the verification process.

If errors are found, they appear in a new window.

XML Migration Wizard

The XML Migration Wizard helps you migrate JAR files containing EJBs based on version 1.2 of the J2EE specification to EJBs based on version 1.3 of the J2EE specification. This wizard also delivers functionality to migrate modules from J2EE 1.3 to J2EE 1.2. The wizard migrates the XML deployment descriptors only. Any non-Borland, vendor-specific information stored in the 1.2 deployment descriptor (that is, a `*.ser` file) will not be migrated.

Note

The XML Migration Wizard does not migrate CMP 1.1 to CMP 2.0.

To fully migrate modules from J2EE 1.2 to J2EE 1.3, some additional steps are required:

1. The source code must be modified to match the EJB 2.0 specification and new class files generated. For example, the signature of the `ejbCreate()` method is different between 1.0 and 1.1 versions, and needs to be modified.
2. You should use the Deployment Descriptor Editor to add and/or modify deployment information that was absent, or was not migrated from the original deployment descriptor, for example, EJB JNDI names, transaction attributes and other information.

Note

You can also migrate JAR files using iastool commands. For more information, see iastool command-line utility in the *AppServer Developer's Guide*.

To upgrade an EJB 1.2 JAR file to a 1.3 JAR file or downgrade a 1.3 JAR to a 1.2 JAR:

1. From the Wizards menu in the Management Console, select XML Migration Wizard.
2. Select the Upgrade or Downgrade option.
3. Click Browse to select the module you want to migrate.
4. Make your selections in the Select File dialog window, and click OK.
5. To designate an output file from the process, either enter a file name or click Browse to navigate to a directory.
6. Click Finish.
7. The results are displayed in a new window.

Stub Generation Wizard

Stub generation is required so that modules are in a form that the server is expecting. For EJBs, this involves generating client and server-side stubs. Because stubs are generated from the Management Console and not on the server, any classes that are needed during stub generation (that is, compile time dependencies of the Home and Remote interfaces of the beans) will need to be available to the Management Console.

The Stub Generation Wizard lets you create the following types of archives:

- A single client library JAR file that is suitable for use by a non-container client application (for example, a CORBA server/client). As a result, EAR modules will be "flattened out" so that standard Java classloaders can be used.
- A single "stubs only" JAR file containing just the generated stub classes (both client-side and server-side) required for server deployment.
- A manually-deployable JAR file that contains all the stubs required for manual deployment to a server.

Creating client stubs for non-container applications

To create client stubs for non-container applications:

1. From the Wizards menu on the Management Console, select Stub Generation Wizard.
2. Choose Create a client library JAR file, then click Next.
3. Click Add to select the modules that you want to generate the stub files for, then click Next.
4. To designate an output file from the process enter a file name (or click Browse to navigate to a directory location and then enter a file name).
5. Optionally, click the Edit button to open the Class Path Editor to add archives and classpaths in the Classpaths field:
   - Click Add Archive to browse for JARs.
   - Click Add Path to browse for paths.

   By default, if you are deploying multiple modules, then all modules are placed on the classpath of the stub generator. So, if your beans have references to beans in other JARs, then a multiple deployment requires that you add the other JARs to the classpath.
1. Optionally, enter any command line arguments for the java2iiop and javac compilers that may be needed by the file. Click the More Info button for a detailed usage statement of Java2IIOP.
2. Click Finish.

Creating a "stubs only" library JAR file

This wizard generates a single JAR file containing just the generated stub classes.

To create a "stubs only" JAR file containing just the generated stub classes:

1. From the Wizards menu on the Management Console, select Stub Generation Wizard.
2. Choose Create a "stubs only" library JAR file, then click Next.

3 Click Add to select the modules that you want to generate the stub files for, then click Next.

4 If you do not want to generate server-side stub files, check Generate client stubs only.

5 To designate an output file from the process, either enter a file name (or click Browse to navigate to a directory location and then enter a file name).

6 Enter any JARs or classpaths on which your beans depend in the Classpath field or click Edit to use the ClassPath Editor.

7 Optionally, enter any command line arguments for the java2iiop and javac compilers that may be needed by the file. Click the More Info button for a detailed usage statement of Java2IIOP.

8 Click Finish.

Creating a JAR suitable for manual deployment

You can use the Stub Generation Wizard will generate JAR files containing all of the necessary stubs for a manual deployment to a server. This can be useful if you want to create a deployable JAR and deploy it later.

To create a server-side deployable JAR for manual deployment:

1 From the Wizards menu on the Management Console, select Stub Generation Wizard.

2 Choose Create archive files for manual deployment, then click Next.

3 Click Add to select the modules that you want to generate the stub files for, then click Next.

4 To designate an output file from the process, either enter a file name (or click Browse to navigate to a directory location and then enter a file name).

5 Enter any JARs or classpaths on which your beans depend in the Classpath field or click Edit to use the ClassPath Editor.

6 Optionally, enter any command line arguments for the java2iiop and javac compilers that may be needed by the file. Click the More Info button for a detailed usage statement of Java2IIOP.

7 Click Finish.

Remove Stubs Wizard

The Remove Stubs Wizard lets you remove any existing stubs from your JAR files.

To remove existing stubs from a JAR file:

1 From the Wizards menu on the Management Console, select Remove Stubs Wizard.

2 Click Add to select the modules from which you want to remove stubs.

3 Either click Overwrite the original modules (each modified JAR will be given the same name as the original module within this directory, or to keep the originals intact, enter an output file), or click Browse to navigate to a directory location and then select a file.

4 Click Finish.

Apply Patch Wizard

The Apply Patch Wizard lets you apply a patch or a set of patches to a JAR file.

To apply a patch to a JAR file:

1. From the Wizards menu on the Management Console, select Apply Patch Wizard.
2. Enter the name of the JAR file to which you want to apply the patch (or click Browse to navigate to a directory location and then select a file).
3. Click Add to select the modules that contain the patches you want to apply.
4. Enter the name of the output file that will be created as a result of this process (or click Browse to navigate to a directory location and then select a file).
5. Click Finish.

JAR Wizard

Use the JAR Wizard to compress or decompress files in a JAR file.

To compress or decompress JAR files:

1. From the Wizards menu on the Management Console, select JAR Wizard.
2. Select either Compress input jar or Decompress input jar.
3. Enter the name of the JAR file to compress or decompress (or click Browse to navigate to a directory location and then select a file).
4. Enter the name of the output file that will be created as a result of this process (you can click Browse to navigate to a directory location).
5. Click Finish.

Configure Agents Wizard

Use the Agent Configuration Wizard to help you configure AppServer agents on your server.

To configure an AppServer agent:

1. From the Wizards menu on the Management Console, select Configure Agents.
2. Select the installation location and the name of the agent you wish to configure from the drop-down lists.
3. You can configure the following settings for the selected agent:
   - Management port.
   - Agent type.
   - Agent shutdown policy.
   - Security profile.
1. Click Finish.

Export EJB as a Web Service Wizard

Use the Export EJB as a Web Service Wizard to export a stateless Session bean as a Web service. Upon completion, the wizard automatically sends requests to the server to generate and deploy the service:

Note

This demonstration uses the Animal example located in the *install_dir*/examples/webservices/ejb directory and you must first run the ant utility to create the animal_ejb.jar file and its beans. Then you must deploy the animal_ejb.jar on any Partition using the Deployment Wizard or the iastool utility deploy tool.

For more information on the Deployment Wizard, see "Deployment Wizard". For more information on the iastool deploy tool, see iastool command-line utility in the *AppServer Developer's Guide*. For more information on the ant utility and how to use the Web services examples, see Borland Web Services examples in the *AppServer Developer's Guide*.

1. In the Management Console, select the Hubs View if it is not already in view.
2. Navigate to a deployed module and expand the node to view the beans it contains.
3. Right-click on a stateless Session bean, then click Export EJB as a Web Service.
4. Enter a name for the Web service and a name for the WAR file you wish to create and deploy. For the Animal example, the Web Service Name is Animal and the WAR Name is animal_ejb. At this point, either click Finish to create your Web service with the default settings, or click Next to supply additional information.
5. In the Select methods screen, Select the Remote Interface Methods listed that you want to make available in the Web service. For the example, the sleep and talk methods are selected in the Animal example. At this point, either click Finish to create your Web service with the remaining default settings, or click Next to supply additional information.
6. In the Edit XML screen, Edit the XML deployment descriptor file as needed, then either click Finish to create your Web service with the default settings, or click Next to supply additional information.
7. In the Additional options needed to generate stubs screen, Enter any JARs or classpaths on which your beans depend in the Classpath field, or click Edit to use the ClassPath Editor. Optionally, enter any command-line arguments for the java2iiop and javac compilers that may be needed by the file. Click the More Info button for a detailed usage statement of Java2IIOP or refer to Programmer tools for Java in the *VisiBroker for Java Developer's Guide* for information about Java2IIOP.
8. When you are done, click Finish.
9. When the process has completed, click Close.
10. To verify that the deployment was successful, click on the newly created WAR file node in the navigation tree, select a URL located in the Additional Web Module Properties box near the bottom of the content pane on the right side of the Management Console, and click Test URL. (You may need to use the scroll bar on the right side of the content pane to view the Additional Web Module Properties box.)
11. To examine the WSDL (Web Service Description Language) file, complete the location or address field in your Web browser with the full URL address of the WSDL file that corresponds to the EJB's Web service. For example, enter services/Animal?wsdl to supply an address such as:

    ```
    http://myserver:8080/animal_ejb/services/Animal?wsdl
    ```

8

Using the Deployment Descriptor Editor

The Borland Management Console includes a Deployment Descriptor Editor (DDEditor) that can be used to change and add deployment information in a J2EE archive's deployment descriptor file. The DDEditor is available in all three editions of the Borland AppServer (AppServer). The AppServer is J2EE 1.4 compliant but also back-compatible with J2EE 1.3 descriptors. The objects that can be edited are:

Descriptors (producing standard and proprietary Borland XML)

- EARs (producing application.xml)
- WARs (producing web.xml)
- EJB 1.1 JARs (producing ejb-jar.xml)
- EJB 2.0 JARs (producing ejb-jar.xml)
- Application Client JARs (producing application-client.xml)
- JNDI Definitions (Borland proprietary product)

Archives

- J2EE 1.3 Application EARs
- J2EE 1.2 Application EARs
- Servlets 2.2 WARs
- Servlets 2.3 WARs
- JNDI Definitions—DARs (Borland proprietary product)
- Resource Adapter Archives (1.0)
- Application Client JARs (1.2 and 1.3)
- EJB JARs (1.1 and 2.0)

This chapter gives an overview of component information included in the descriptor file, explains how to use the DDEditor included with the AppServer, and provides a list of descriptor information you can edit in the file.

Archive directory structures

Applications are developed within a specified directory structure so that they can be archived and deployed on a J2EE server. All classes, files, servlets and other resources are organized hierarchically in a directory.

About the descriptor

J2EE applications are composed of one or more J2EE components and one J2EE application deployment descriptor. The deployment descriptor describes the application's components as modules.

A component deployment descriptor is an XML file that describes the units within an archive (module) being deployed. The information provided in the deployment descriptor is necessary for deploying to the AppServer. Deployment descriptors tell the Deployer how to deploy an application. The J2EE specification defines some of the deployment descriptors like the `ejb-jar.xml` and `web-app.xml` descriptors. Borland provides additional deployment descriptors with the required information to deploy components in the AppServer.

You can edit deployment descriptors by using the AppServer's DDEditor tool. Code generation tools such as JBuilder can also automatically generate deployment descriptors, or you may code them yourself.

Usage conditions

The DDEditor can be used with a wide range of archive and descriptor types. There are a number of usage considerations for each of these.

General

1. The DDEditor automatically creates a deployment descriptor that is XML-based so you do not have to learn XML.
2. You can edit an archive file that contains embedded XML or you can edit the XML file directly.
3. Basic code editing features are included in the DDEditor when editing XML directly:
   - *CTRL+F* = find and replace
   - *CTRL+G* = go to line number
1. The DDEditor conforms to the semantic rules specified in Sun's DTD. It implements these rules on the data you enter.
2. The DDEditor automatically sets up the Borland-specific extensions in a separate file.
3. Navigation pane, in this document refers to the left pane of the DDEditor, and work pane refers to the right pane of the DDEditor.

EARs

The EAR is the basic deployment unit. The EAR is not however, very useful without archives with which to populate it. Therefore, the basic procedure of creating an EAR involves:

1. Creating a new EAR.
2. Populating the EAR with its component parts such as WARs, Application Clients and EJB JARs, or create them within the EAR.

WARs

WARs are archives that are intended for use with Web applications. They support objects such as servlets JSPs, HTML and so on.

If the distributable flag is set on servlets, it indicates to the web container (Tomcat) that this web application's servlets and JSPs are capable of running in a separate Virtual Machine and/or failing over to another container or Partition. Accordingly, there are a number of restrictions/implementation requirements that any servlet marked as distributable must observe:

- You cannot use the servlet context to share information between servlets. You have to use a Session instead.
- Any object that you want to share in the Session *must* implement `serializable`. If you try to stuff something non-serializable into the session, you'll get a Invalid Argument exception.

WAR directory structures

The root of this hierarchy defines the document root of your Web application. All files under this root directory can be served to the client except for files under the `WEB-INF` directory (under the root directory). The name of your Web application is used to resolve requests for components of the Web application.

Place all private files in the `WEB-INF` directory. All files under `WEB-INF` are private and are not be served to the client.

`WebApplicationName/`: Static files such as HTML and JSP files should go in this directory. This directory is the document root of the Web Application.

`/WEB-INF/web.xml`: This is the Web application deployment descriptor that configures the Web application.

`/WEB-INF/borland.xml`: This is the Borland-specific deployment descriptor that defines how named resources in the `web.xml` file are mapped to resources that are located somewhere else in the server. This descriptor file is also used to define JSP and HTTP session attributes.

`/WEB-INF/classes`: This contains server-side classes such as servlets and utility classes.

`/WEB-INF/lib`: This contains the JAR files used by the Web application. This includes JSP tag libraries.

`/WEB-INF/classes`: This contains server-side classes such as servlets and utility classes.

`/WEB-INF/lib`: This contains the JAR files used by the Web application. This includes JSP tag libraries.

EJBs

Some things to remember about using the editor with EJBs include:

- You can edit an EJB JAR file that contains embedded XML, or you can edit the XML file directly.
- If you launch the DDEditor as a stand-alone application and the AppServer is running, you should not edit a JAR file deployed in a Borland Partition.
- The Set Classpath button works with EJB JAR files only.
- Entity Beans can use either JDBC 1 or JDBC 2 datasources. JDBC 1 datasources are local to an EJB JAR. JDBC 2 datasources can be used by any EJB.

For information about working with JDBC2 datasources, see the *AppServer Developer's Guide*.

JNDI Definitions

Remember when using the DDEditor with JNDI definitions that they can be deployed independently of an application or as part of an application archive (EAR).

Starting the editor

You can run the DDEditor as a stand-alone application or as part of the Management Console.

You can also start the editor by selecting a file from the Deployed Modules node in the navigation tree of the Console. Right-click on the module you want to edit and select "Edit deployment descriptor" from the context menu. A process window will appear and then launch the DDEditor.

To start the DDEditor:

1. Depending on the choices you made during installation, use one of the following methods to start the editor:
   - On Windows NT, click the Start button and choose Deployment Descriptor Editor from the AppServer program group.
   - Start the Console and choose DDEditor from the Tools menu.
   - If you have added the AppServer's `bin` directory to your path, open a command window and enter the following command

     ```
     ddeditor <filename>
     ```

     where *filename* is the name of the XML or archive file you wish to edit. (Otherwise change to the `bin` directory and enter the same command.)
1. If necessary, select the deployment descriptor you want to edit.
2. Choose Open from the File menu to open an existing deployment descriptor.
3. Choose New from the File menu to create a new deployment descriptor or archive.
4. When the main Descriptor Editor window appears, use the tabs to examine or add descriptor information. You can use the Navigation panel (left side of Borland Management Console) to examine, create or rename your files.

The following sections describe the types of information you can edit in the deployment descriptor.

Adding descriptors

The Borland DDEditor enables you to create deployment descriptors for each type of archive, as well as for JNDI definitions.

The deployment descriptor is an XML file which follows the Document Type Definition (DTD) approved by Sun Microsystems. Deployment descriptors contain a set of properties that describe how the container deploys an archive.

The types of information in the descriptor

The information in the deployment descriptor can be divided into two basic kinds of information:

- **Structural information.** Structural information describes the structure of archive files (and JNDI definitions) and their modules. For archive files, this involves general information like " Display Name", "EJB JAR name", "EJB Client JAR name", and "Authorization domain". In the case of enterprise beans, the structural information declares the enterprise bean's external dependencies. This information is required and cannot, in general, be changed (doing so could break the enterprise bean's function).
- **Application assembly information.** Application assembly information describes how the archive files and enterprise beans are composed into a larger application deployment unit and how they interact with the application as a whole. Assembly level information can be changed without breaking modules' function.

The following section describes the "Structural" and "Assembly" information for the various deployable modules.

Enterprise bean structural information

All enterprise beans:

- enterprise bean class
- enterprise bean home interface
- enterprise remote interface
- enterprise bean local home interface
- enterprise bean local interface
- enterprise bean home JNDI name
- enterprise bean local home JNDI name
- environment entries
- EJB references (if an enterprise bean references another enterprise bean)
- resource (factory) references (if a datasource is used)
- security role references
- security identity

Enterprise session beans:

- Session bean's state management type
- Session bean's transaction type

Enterprise entity beans:

- Entity bean's persistence type
- Entity bean's primary key class
- Entity bean's reentrant behavior

Enterprise entity beans with container-managed persistence:

- Container-managed fields
- Container-managed relationships

Enterprise bean application assembly information

You can specify any of the following application assembly information in your deployment descriptor.

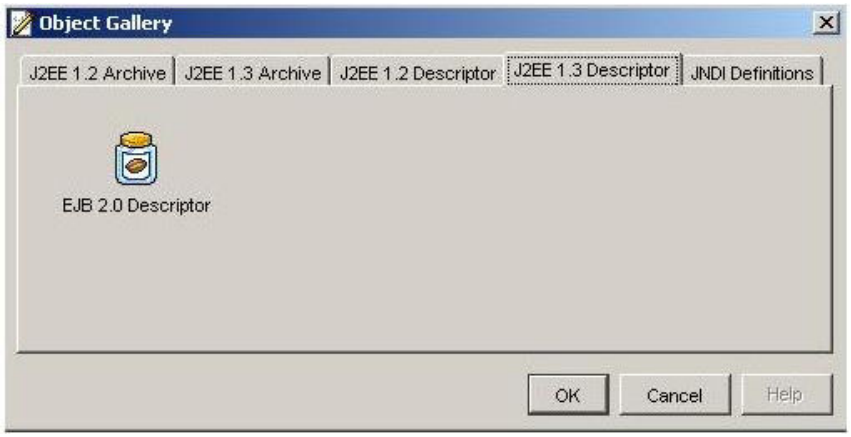
- Binding of enterprise bean references
- Security roles
- Method permissions
- Security role reference links
- Exclude lists
- Container transactions for Entity beans

Creating Deployment Descriptors

Deployment descriptors contain the same information that is used in the creation of an archive, so no detailed explanation will be included here.

To create new definition descriptors:

1. Open the DDEditor.
2. Select New from the file menu.
3. The Object Gallery window appears. Select a descriptor or archive type from one of the four tabs in the Object Gallery.
4. Assign properties to the descriptor.
5. Once the descriptor is as you wish, select Save from the File menu.
6. Select a folder and click Save. The descriptor is saved as the *Descriptor_Type*.xml. In all cases, excepting JNDI descriptors there is also a proprietary XML file.



Creating a JNDI definition descriptor

The process of creating a JNDI definition descriptor is similar to creating any descriptor type.

To create a JNDI definition descriptor:

1. Open the DDEditor.
2. Select New from the file menu.
3. Select the JNDI Definitions tab in the Object Gallery.
4. Select JNDI Definitions and click OK.
5. To add one of the following datasources:
   - New JDBC Datasource
   - New JMS object
   - New JNDI object

   right-click the jndi-definitions object in the Navigation pane and select one of the options. When the JNDI name window appears, enter a name (after the prepended name given).
6. Once the descriptor is as you wish, select Save from the File menu. The descriptor is saved as the `jndi-definitions.xml`.

Note

New JDBC Datasources, JMS objects and JNDI objects added will appear in the “Standard XML” tab.

JNDI definitions and datasource archives (DARs)

Borland provides mechanisms for deploying both JDBC resource factories and JMS connection factories.

The AppServer's EJB Container creates datasources for you. The only thing you need to do for the Container is specify the information about the resources to which your application(s) need to connect and bind the datasource to JNDI. Once bound, you can reference the datasource from your enterprise beans using the `<resource-reference>` element in your deployment descriptor.

Datasources are bound to JNDI when they are deployed as a part of a JNDI definitions module. The module is similar to other J2EE standard Java archive types and ends in the extension `.dar`. The module is also referred to as a DAR. The DAR adds to the standard J2EE module types and can be packaged as part of an EAR or deployed as a standalone. Since datasources are contained in their own module, they appear in the Server tree of the AppServer's Console making it easy to enable, disable and edit (by launching the DDEditor) by right-clicking their representations in the tree.

Note

A DAR is not part of the J2EE specification. It is a Borland-specific implementation designed to simplify the deployment and management of connection factories. Do not package connection factory classes in this archive type. Connection factory classes must be deployed as a library to individual partitions.

Since the Container constructs the connection factory class, the only thing you must provide the DAR is an XML descriptor file called `jndi-definitions.xml`. Like other descriptors, this is in the `META-INF` directory of the DAR. A DAR looks like this:

```
META-INF/jndi-definitions.xml
```

You deploy the DAR containing the descriptor file just as you would any other J2EE module using either the Console or command-line utilities, or as part of an EAR.

Creating a JNDI definitions archive (DAR)

To create a JNDI definitions archive (DAR):

1 Open the DDEditor.

2 Select New from the file menu.

3 Select the JNDI Definitions tab in the Object Gallery.

4 Select JNDI Definitions Archive and click OK.

5 To add one of the following datasources:

- New JDBC Datasource
- New JMS object
- New JNDI object

6 Right-click the jndi-definitions object in the Navigation pane and select one of the options. When the JNDI name window appears, enter a name (after the prepended name given).

7 Once the archive is as you wish, select Save from the File menu.

8 Assign a name to the archive and it is saved as a `.dar` file.

Setting properties for datasources

Once you've created a datasource you can view and configure its properties. To view or configure datasource properties:

1 Select the datasource, for example, “database” (for a JDBC datasource) in the navigation pane.

2 For JDBC datasources you can set main properties, pool properties and driver properties.

- **Main:** This tab allows you to select a pre-configured, commonly used Datasource Type (JDataStore, Oracle, Oracle.XA, Sybase, JDBC2, etc.) or add one not listed. In addition, if you select a pre-configured Datasource Type, other properties required for that source are provided such as Class name, Max pool size, etc. There are several fields you configure, such as Driver type, Database name, Server name, port number, user, password, etc. These properties are part of the XML also, which can be viewed and edited in the XML tab. See XML example below:

```
<driver-datasource>
        <jndi-name>datasources/myDriver</jndi-name>
        <datasource-class-name>oracle.jdbc.xa.client.OracleXADataSource</
datasource-class-name>
        <log-writer>False</log-writer>
        <property>
            <prop-name>driverType</prop-name>
            <prop-type>Enumerated</prop-type>
            <prop-value>thin</prop-value>
        </property>
        <property>
            <prop-name>portNumber</prop-name>
            <prop-type>Integer</prop-type>
            <prop-value>1521</prop-value>
        </property>
        <property>
```

Pool Properties: This tab allows you to set the pool properties of Name, Type and Value. Depending on the Type, Value options change. The Names include:

- initSQL
  - description
  - reuseStatements
  - busyTimeout
  - idleTimeout
  - queryTimeout
  - maxPoolSize
  - dialect
  - isolationLevel
  - resSharingScope
  - refreshFrequency
  - connectionType
  - waitTimeout
  - dbPingSQL
- **Driver Properties:**This tab allows you to configure and add driver properties.

3 For JMS objects you can set main and additional properties.

- **General:** This tab shows the class name for your JMS object and the pre-configured, commonly used Connection Factory, `QueueConnectionFactory`. If you use the bundled Tibco JMS services provider with AppServer, the class name appears as: `com.tibco.tibjms.appserver.borland.TibjmsBorlandQueueConnectionFactory`.
- **Properties:** Any parameters that are required for JMS objects are configured here. These properties are part of the XML `<,jndi-object>` element, which can be viewed and edited in the XML tab. See XML example below:

```
<jndi-object>
        <jndi-name>jms/message</jndi-name>
        <class-name>progress.message.jclient.QueueConnectionFactory
           </class-name>
        <property>
            <prop-name>brokerURL</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>
```

Note

For JMS services vendor-specific configuration and property information, refer to the JMS provider pluggability in the *AppServer Developer's Guide*.

Note

One JNDI Definition module is pre-deployed in the AppServer Partition and is easily configurable using the server's tools. This module is called `default-resources.dar` and `jms-resources.dar`. Each of these contains some pre configured connection factories that you can edit by right-clicking the module in the server tree and selecting the DDEditor from the context menu.

Note

For more information on defining JDBC datasources and connection factories refer to Connecting to resources in AppServer: using the Definitions Archive (DAR) in *AppServer Developer's Guide*.

Adding EAR archives

When creating J2EE compliant applications, the EAR forms the basis of deployment. When creating an EAR for deployment with AppServer, the recommended procedure is:

1. Open the DDEditor.
2. Select New from the file menu.
3. Once the Object Gallery appears, select either 1.3 Application Archive from the J2EE 1.3 Tab or 1.2 Application Archive from the J2EE 1.2 Tab.
4. Assign properties to the EAR archive.
5. Once the EAR archive is as you wish, select Save As from the File menu.
6. Assign a name to the archive with the extension `.ear` and click Save.

EAR properties

Several EAR archive properties are displayed on the Properties pane and can be edited after creation.

General Tab

Use the General Tab to enter or change general information about the EAR archive. The information includes:

- **Application Display Name:** The logical name assigned to the application that is displayed by the Console.
- **Application File Name:** The name and location of the EAR file that is defined by the archive.
- **Description:** A summary of the application's purpose and function. This information is optional.
- **Authorization domain:** An authorization domain defines the set of rules that determines whether a user belongs to a role or not. When an access decision is performed (see the sections on "Assigning method permissions" and "Adding security roles and method permissions" in this document), the authorization domain maps the logical roles required to access a resource to a set of rules. It consults a database of rules that determine whether a user belongs to a given role. Rules describe what attributes and corresponding values are required for a user to belong to a role. The caller's security attributes are then matched with the required rules to determine whether the caller's security attributes satisfy the rules required for access.

  The authorization domain an application uses is determined by the `<authorization-domain>` deployment descriptor, or the `domain()` method. The `authorization-domain` tag is used to bind a JAR, WAR, or EAR to an authorization domain. This tag does not define an authorization domain but merely binds the resources in the J2EE application to the specified authorization domain.

  The `authorization-domain` element is a Borland-specific element and appears in the Vendor XML tab.

Context Roots Tab

This tab allows you to specify the proper URI configuration for your servlets, ensuring they are only called from the proper URI mapping.

- **Web URI:** The URI pattern of the servlet, for example, “helloweb.war”.
- **Context Root:** Name of the Web application from which the servlet is called.

Properties Tab

This tab allows you to set various environment properties relative to your WAR:

- **Name:** The name of the environment variable.
- **Type:** The environment variable type.
- **Value:** The value of the environment variable.

XML Tab

You may, if you choose, edit the XML directly. Choose either the Standard or Vendor XML Tabs to edit. Once you are finished, click Apply Changes and save the archive.

Adding WAR information

To create a WAR archive:

1. Open the DDEditor.
2. Select New from the file menu.
3. Once the Object Gallery opens, select either Servlets 2.2 WAR from the J2EE 1.2 Archive Tab or Servlets 2.3 WAR from the J2EE 1.3 Archive Tab.
4. Assign properties to the WAR archive.
5. Once the WAR archive is as you wish, select Save from the File menu.
6. Assign a name to the archive with the extension `.war` and click Save.

WAR properties

WAR archive properties are displayed on the Properties pane and can be edited after creation.

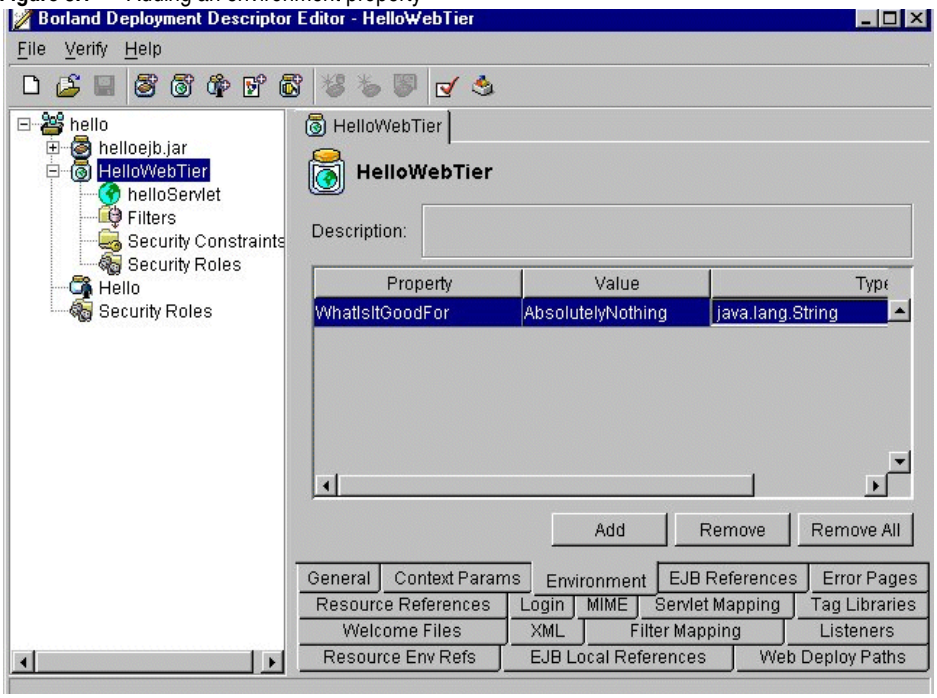
General

- **WAR Display Name:** The logical name assigned to the Web component that is displayed by the Console, for example, “HelloWebTier”.
- **WAR File Name:** The name and location of the WAR file that is defined by the archive, such as, “helloweb.war”.
- **Description:** A summary of the Web component's purpose and function. This information is optional.
- **Context Root:** Name of the web application calling the servlets, for example, “hello”.
- **Authorization domain:** See previous section on Authorization domains.
- **Distributable:** This check box indicates whether the component is programmed appropriately to be displayed in a distributed service container.
- **Session Timeout**: Determines how long before an inactive session is terminated.

Context Params

- **Context Parameter:** Servlet context initialization parameters.
- **Value:** The value of the servlet context initialization parameters.
- **Environment Description:** A brief description of the purpose of the environment variable.
- **Property:** The name of the environment variable.
- **Value:** The value of the environment variable.
- **Type:** The environment variable type. The options are:
  - Boolean
  - String
  - Integer
  - Character
  - Double
  - Byte
  - Short
  - Long
  - Float

Figure 8.1 Adding an environment property



EJB References

An EJB reference is a pseudonym for the JNDI location that you want to lookup a bean. The pseudonym may not actually correspond to the actual JNDI location that is in your bean. Your code looks up a home via its pseudonym, and then you bind the pseudonym to the JNDI location, perhaps using symbolic links (links are described below).

To use a programmatic example, if Bean A needs to use Bean B, you can simply declare all the necessary information about Bean B in an EJB reference. The deployer then knows that Bean A uses one other bean, Bean B. This is useful because the deployer now knows which class files Bean A depends on and which JNDI location needs to be bound. The Container's tools can easily inspect the deployment descriptor and verify that the deployer has done this correctly.

```
...
<enterprise-beans>
    <session>
           <ejb-name>BeanA</ejb-name>
           <home>examples.AHome</home<
               ...
    </session>
    <session>
           <ejb-name>BeanB</ejb-name>
           <home>examples.BHome</home<
               ...
           <ejb-ref>
                  <description>
         The EJB ref says Bean B uses Bean A
                  </description>
                  <ejb-ref-name>ejb/AHome</ejb-ref-name>
                  <ejb-ref-type>Session</ejb-ref-type>
                  <home>examples.AHome</home>
                  <remote>examples.A</home>
                  <ejb-link>BeanA</ejb-link>
           </ejb-ref>
       </session>
   </enterprise-beans>
```

- **Description:** A brief description of the function of the EJB.
- **Name:** The JNDI name used in the code.
- **IsLink:** Check to specify that this is a reference to local beans, otherwise beans are referenced by JNDI names.
- **Link:** Links the EJB Reference to the target enterprise bean. The Link value is the name of the target enterprise bean. This information is required if IsLink is checked, otherwise it is not used.
- **Type:** Specify the EJB type. The options are:
  - Entity Bean
  - Session Bean
- **Home:** The fully-qualified name of the of the bean's home interface.
- **Remote:** The fully-qualified name of the of the bean's remote interface.
- **JNDI Name:** The JNDI name of the bean's home interface. This is not used if ISLink is specified.

EJB Local References

The EJB Local References panel lists all the enterprise bean references to the homes of other enterprise beans within a single archive.

Each EJB Local Reference describes the interface requirements that the referencing enterprise bean has for the referenced enterprise bean. The reference contains:

- **Description:** A brief description of the bean that is referenced. This information is optional.
- **Name:** The name of the referenced bean. The tool tip information shows what you use to access the Name in code.
- **IsLink:** If checked, there is a link to a local enterprise bean and the following fields are filled in and read-only: Type, Home, and Remote. If unchecked, reference is to an enterprise bean outside the JAR file and you must fill in Type, Home, and Remote.
- **Link:** Links the EJB Reference to the target enterprise bean. The Link value is the name of the target enterprise bean.
- **Type:** The expected type of the referenced bean.
- **Local Home:** The expected Java type of the referenced bean's home interface.
- **Local:** The expected Java type of the referenced bean's local interface.
- **JNDI Name:** The JNDI name of the referenced bean.

Error Pages

- **Type:** Error type. The options are “HTTP Error Code” or “Java Exception Type”.
- **Error/Exception:** The name of the Error/Exception file.
- **Location:** The location of the Error/Exception file. This is relative to the root of the web application and must start with a “/”.

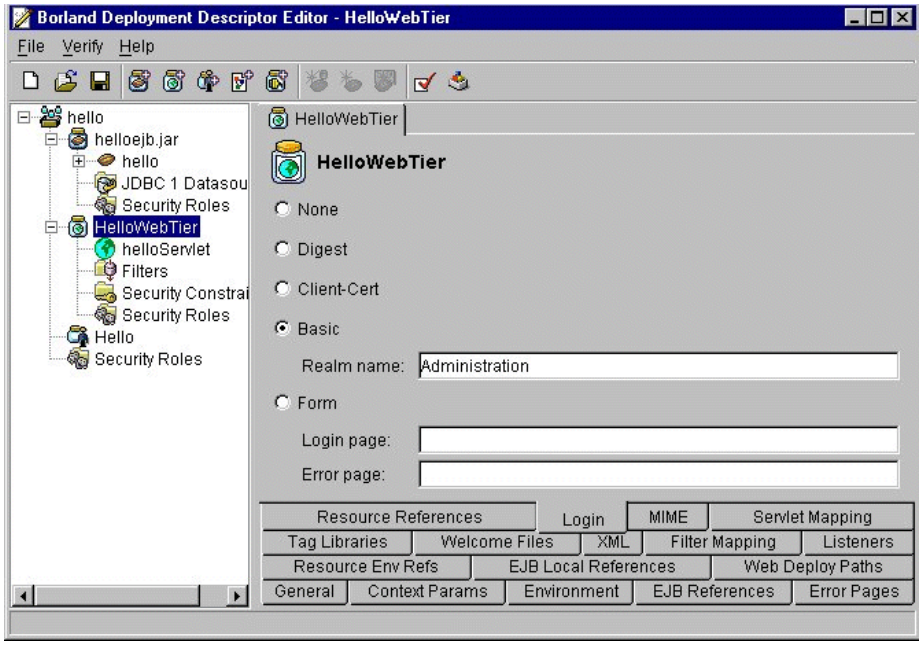
These are represented in the standard XML element `<error-page>` with `<error-code>` and `<location>` as subordinate elements.

Login

This tab is used to select an authentication method for credential verification. When you have a Web browser client connecting to a servlet/JSP layer, the Web browser user usually supplies credentials to the servlet/JSP layer and the servlet/JSP layer then may use JAAS to authenticate the browser user. The web browser can supply the credentials to verify the login to the web component by using any one of the methods below.

- **None**
- **Digest.** The web client supplies a special message digest to the web server. This message digest is a symbolic transformation of the user's password and the HTTP message. The password itself is not sent to the server. The web server then reproduces the message digest by performing the same symbolic transformation, but the server this time uses a secure copy of the user's password kept in permanent storage. If the message digests match, the user is authenticated.
- **Client-Cert.** The client can establish an identity with X.509 certificates. Optionally, the client can also ensure that a third party is not impersonating the server by receiving X.509 certificates that authenticate the server.
- **Basic.** The web client supplies a user name and password to the web server. The server checks the credentials against a permanent storage of user names and passwords. Note that you must also specify the realm to which to authenticate here.
- **Form.** This is just like basic authentication except the application uses a customizable form such as a special login screen. Note that you must also specify URLs for the login page and error page here.

Figure 8.2 Setting login and error pages for Form-based authentication



MIME

This refers to the standard XML element `<mime-mapping>` . The mime-mapping element defines a mapping between an extension and a mime type. Two corresponding attributes are `<extension>` and `<mime-type>`.

- **Extension:** The file extension that denotes a MIME type.
- **Mime Type:** The defined MIME type which is recognized by the Web component, for example, “text/plain”.

Resource References

The Resource References panel lists all the enterprise bean's resource factory references. This enables the application assembler and/or bean deployer to locate all references used by the enterprise bean.

Information in the panel includes:

- **Description:** A description of the resource reference. This information is optional.
- **Name:** The name of the environment entry used in the enterprise bean's code.
- **Type:** The Java type of the resource factory expected by the enterprise bean's code. This is the Java type of the resource *factory*, not the Java type of the resource. The types available are:
  - javax.sql.DataSource
  - java.net.URL
  - javax.mail.Session
  - javax.jms.QueueConnectionFactory
  - javax.jms.TopicConnectionFactory

  The Deployment Settings (parameters) you can specify depend upon the Java type selected.
- **Authentication:** An Application authentication indicates that the enterprise bean performs the resource sign-on programmatically. A Container authentication

indicates that the Container signs on to the resource (on behalf of the web application) based on the principal mapping information supplied by the deployer.

- **Sharing Scope:** The `<res-sharing-scope>` element specifies whether connections obtained through the given resource manager connection factory reference can be shared. The values of this element must be either `<Shareable>` or `<Unshareable>`. The default is `Shareable`.

Servlet Mapping

- **Servlet:** The name of the servlet.
- **URL Pattern**: Map URL to the servlet name.

Tag Libraries

The `<taglib>` element is used to describe a JSP tag library.

- **Library URI:** The URI of the path (relative to the location of the web.xml document) to be added on `onLoad`.
- **Description File Location:** The `<taglib-location>` is the location path of the Tag Library description file for the tag library.

Welcome Files

The `<welcome-file-list>` element contains an ordered list of welcome files elements. The `<welcome-file>` element contains a file name to use as a default welcome file, such as `index.html`.

Specify the URL of the page that is first displayed when users access the web component.

XML

You may, if you choose, edit the XML directly. Choose either the Standard or Vendor XML Tabs to edit. Once you are finished, click Apply Changes and save the archive.

Filter Mappings

- **Filter:** The logical name of the filter used to map the filter. Each filter name in a web application is unique.
- **Mapping Type:** The type the filter is mapped to; a servlet or URL pattern.
- **Mapping:** The container uses the filter-mapping declarations to decide which filters to apply to a request, and in what order. The container matches the request URI to a servlet in the normal way. To determine which filters to apply, it matches filter-mapping declarations either on a servlet-name, or on url-pattern for each filter-mapping element, depending on which style is used. The order in which the filter mappings are invoked is the same order they appear in the list of filter-mapping elements. You may wish to view the generated XML after filling in these fields to make sure your filter mappings are in the proper order. The mapping will be the value of the mapping type in the descriptor.

Listeners

Listener Classes: Declares a class or classes in the application that must be registered as a web application listener bean. This value must be the fully-qualified class name of the listener class.

Resource Env Refs

The <resource-env-ref> element contains a declaration of a web application's reference to an administered object associated with a resource in the application's environment.

Resource Environment Ref Name: Specifies the name of a resource environment reference. The reference must be a fully qualified name of a Java class or interface. It must be unique within a web application.

Type: The resource manager connection factory type expected by servlet code. It is the fully qualified name of a Java language class or interface.

JNDI Name: JNDI name of the resource manager connection factory.

Web Deploy Paths

The Web Deploy Paths tab allows you to specify exactly where to deploy the web application (service, engine, and host). The Borland web container (based on Tomcat) has a notion of a host being part of an engine, which in itself is a part of a service. There can be multiple hosts under an engine and there can be multiple engines under a given service. A given web application can be deployed to one or more of these hosts. The service, engine, and host you specify using this element, override the defaults. However, this element does accept multiple entries.

The default is service=HTTP, engine=HTTP, and host=* (deploy to all hosts available under the specified engine).

Note

The Tomcat web container does not handle the consumption of EARs as easily as it does WARs. In order for the IIOP connector to handle deployment of EARs, embedded WAR files need to indicate they should be deployed to the IIOP connector.

To do this use the <web-deploy-path> element provided in the web-app_2_3-borland DTD (Borland-specific). This element indicates where in the Tomcat web container "hierarchy" to deploy the WAR.

The XML needed for this is shown below. You can hand code it or use the DDEditor's Web Deploy Paths tab:

```
<web-deploy-path>
  <service>IIOP</service>
  <engine>IIOP</engine>
  <host>localhose</hose>
</web-deploy-path>
```

Web Services

To view or modify the server-config.wsdd file using DDEditor:

1. In the Management Console, choose Tools|Deployment Descriptor Editor.

   The Borland Deployment Descriptor Editor appears.

2. Choose File|Open File.

   The Open J2EE archive or descriptor window appears.

3. Navigate to the directory where the WAR file resides.

4. Select the WAR file and click OK.

5. At the top of the navigation tree, click the name of the archive.

6. On the bottom right of the contents pane, click the XML tab.

7. Click the web services tab.

   The raw server-config.wsdd file displays.

8. Make your edits.
9. Click Apply Changes.
10. Choose File|Save.

For information on creating, editing, and deployment descriptors for Web Services using AppServer refer to Borland AppServer Web Services in the *AppServer Developer's Guide*.

EJB JAR properties

EJB archive properties are displayed on the Properties panes and can be edited after creation.

General

- **EJB Jar Display Name:** The logical name assigned to the EJB JAR that is displayed by the Console.
- **EJB JAR File Name:** The name and location of the EJB JAR file (read-only).
- **Description:** A summary of the EJB's purpose and function. This information is optional.
- **EJB Client JAR Filename:** The name and location of the client EJB JAR file.
- **Authorization Domain:** The name of the authorization domain the JAR belongs to. The authorization domain typically maps the logical roles required to access a resource to a set of pre-defined roles.
- **Small Icon:** The file name of the 16X16 pixel icon used to identify this EJB.
- **Large Icon:** The file name of the 32X32 pixel icon used to identify this EJB.

XML Tab

You may, if you choose, edit the XML directly. Choose either the Standard or Vendor XML Tabs to edit. Once you are finished, click Apply Changes and save the archive.

Properties (environment)

Information in the Properties/Environment panel(s) is used in setting environment attributes/properties. Properties define how the object operates within a particular environment. Setting the properties in this panel allows you to customize the object's business logic when the it is assembled or deployed without accessing or changing the it's source code.

- **Property:** The name of the environment variable.
- **Value:** The value of the environment variable.
- **Type:** The environment variable type. The options are:
  - Boolean
  - String
  - Integer

EJB Designer

For information about the EJB Designer, see “The EJB Designer”.

Application Client JAR properties

Application Client JAR archive properties are displayed on the Properties panes and can be edited after creation.

General

- **Application Client Display Name:** The logical name assigned to the Application Client that is displayed by the Console.
- **Application Client File Name:** The name and location of the Application Client file.
- **Description:** A summary of the Application Client purpose and function. This information is optional.
- **Small Icon:** The file name of the 16X16 pixel icon used to identify this Application Client.
- **Large Icon:** The file name of the 32X32 pixel icon used to identify this Application Client.

Environment

See “Environment” description above.

EJB References

- **Description:** A brief description of the function of the EJB.
- **Name:** The JNDI name used in the code.
- **IsLink:** Check to specify that there are references to local beans, otherwise beans are referenced by JNDI names.
- **Type:** Specify the EJB type. The options are:
  - Entity Bean
  - Session Bean
- **Home:** The fully-qualified name of the of the bean's home interface.
- **Remote:** The fully-qualified name of the of the bean's remote interface.
- **JNDI Name:** The JNDI name of the bean's home interface. This is not used if ISLink is specified.

Resource References

The Resource References panel lists all the enterprise bean's resource factory references. This enables the application assembler and/or bean deployer to locate all references used by the enterprise bean.

See information on the resource reference panel above.

- **Authentication:** An Application authentication indicates that the enterprise bean performs the resource sign-on programmatically. A Container authentication indicates that the Container signs on to the resource based on the principal mapping information supplied by the deployer.

Some things to remember about EJB References include:

- The target enterprise bean must be type compatible with the declared EJB reference.
- All declared EJB references must be bound to the homes of enterprise beans that exist in the operating environment.
- If a Link value is specified, the enterprise bean reference must be bound to the home of the target enterprise bean.

EJB Local References

See "EJB Local References Panel (EJB 2.0 only)"

XML Tab

You may, if you choose, edit the XML directly. Choose either the Standard or Vendor XML Tabs to edit. Once you are finished, click Apply Changes and save the archive.

Adding or changing bean information

You can add a new bean to the deployment descriptor or change an existing bean.

Note

You can use the Add Beans buttons on the Toolbar or use the right-click context menu commands to add beans.

To add an enterprise bean:

1. Start the DDEditor.
2. Select the type of bean you wish to add:
3. To add a bean, right-click the JAR in the Navigation pane and select a New Bean type from the context menu.

   Note

   Adding Message-Driven Beans is for EJB 2.0 only.

   A name dialog appears.
4. Enter the name of the enterprise bean and click OK.

   The new bean appears in the Navigation pane.

   Note

   The bean name is a logical name assigned to the enterprise bean by the bean provider. Each enterprise bean has a logical name. There is no structural relationship between the bean's logical name and the JNDI name assigned to the bean. The bean deployer may change the bean's logical name.
5. Select the bean.

   A template for the enterprise bean appears in the Work pane. Use the tabs at the bottom of the pane to enter bean information.

Setting classpath information

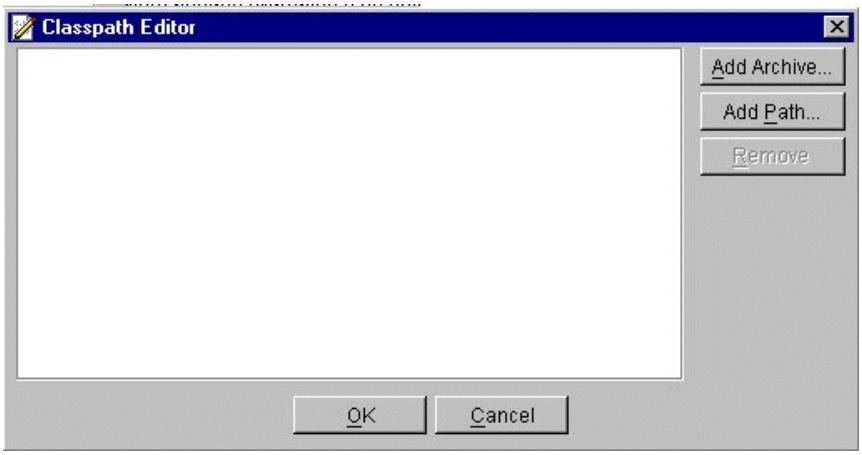
To set a classpath using the DDEditor:

1 Start the DDEditor and open an EJB JAR descriptor file or archive.

2 Choose Set ClassPath from the Verify menu or use the Set ClassPath button.

The Classpath Editor appears.

Figure 8.3 The Classpath Editor



3 Using the buttons provided, you can add a classpath or one or more JAR files, add a path to class files, add an archive or a JAR file, or remove a JAR file. There are several cases:

4 If you need to edit the XML for classes in the file system (that is, not in a JAR file), you must add that path to the classpath using the Add Path button.

5 Classes in the directory containing the `META-INF` directory are on the classpath by default.

6 If your JAR has dependencies on another JAR, you must add it to the JAR's classpath.

7 You may also add specific drivers to the classpath for the Test button on the JDBC 1 Datasource panel and the Get Metadata button on the CMP 1.1 panel.

8 Click OK when you're finished.

Changing bean information

To change bean information, use the following steps:

1 Start the DDEditor and open a descriptor file.

2 Select the bean in the Navigation pane.

A detailed description of the bean appears in the Main pane. Use the tabs at the bottom of the panel to view or edit bean information. (See the following sections for details.)

Enterprise bean information

This section describes the type of information you can create and store for enterprise beans in the deployment descriptor file.

General Tab

Use the General panel to enter or change general information about the enterprise bean (EJB 1.1 or 2.0). This information includes:

- **Bean Class:** The fully-qualified name of the Java class that implements the bean's business methods. This information must be specified.
- **Home Interface:** The fully-qualified name of the enterprise bean's home interface. This information must be specified.
- **Remote Interface:** The fully-qualified name of the enterprise bean's remote interface. This information must be specified.
- **Local Home Interface:** The fully-qualified name of the enterprise bean's local home interface (for beans in the same process). This information must be specified.
- **Local Interface:** The fully-qualified name of the enterprise bean's local interface (for beans in the same process). This information must be specified.
- **JNDI Name:** The JNDI name of the enterprise bean's home interface.
- **Local Home JNDI Name:** The JNDI name of the enterprise bean's local home interface.
- **Description:** A summary of the bean's purpose and function. This information is optional.

For Session beans, the General panel also includes the following:

- **Session Type:** Specifies whether the enterprise bean is Stateless or Stateful.
- **Transaction Type:** Specifies whether Transaction Policies are set by the bean or the Container.
- **Timeout:** The timeout limit applied to transactions on Stateful beans.

For Entity beans, the General panel also includes the following:

- **Primary Key Class:** The fully-qualified name of the Entity bean's primary key class. The primary key class must be specified.
- **Reentrant:** A check mark indicates the bean is reentrant. Borland recommends you avoid making a bean reentrant in order to avoid unintended multithreading.

  The reentrant element dictates whether the bean can call itself through another bean. For example, a given bean A is reentrant if it calls bean B which then calls back on bean A. This a case of multithreading, but it is really only one path that loops back on itself.

  If your bean doesn't call itself through another bean set this value to “False” by leaving the Reentrant box for entity beans unchecked to avoid unintended multithreading. If you want to support reentrant behavior set it to “true” by checking the Reentrant box for entity beans so that the container will allow two threads to run inside a bean at once.

  Note

  You should take special care if you are declaring an entity bean reentrant. The critical issue is that the Container can not generally distinguish between a (loopback) call within the same transaction and a concurrent invocation (in the same transaction context) on the same entity bean.
- **Persistence Type:** Select either “container” or “bean” from the drop-down list.

Environment panel

Information in the Environment panel is used in setting EJB environment attributes/ properties. Properties define how the EJB operates within a particular environment. Setting the properties in this panel allows you to customize the bean's business logic when the bean is assembled or deployed without accessing or changing the bean's source code.

Each enterprise bean defines its own set of environment entries. All instances of an enterprise bean share the same environment entries. Enterprise bean instances are not allowed to modify the bean's environment at runtime.

To add an environment entry:

1. Click Add to create a new entry.

   A new, blank row appears in the panel.

2. Enter a property in the Property column and a property value in the Value column.

3. Choose a property type from the Type drop-down menu.

   Property types can be: String, Integer, Boolean, Byte, Short, Long, Double, Character, and Float.

4. Continue to add environment entries as desired.

Some things to remember about environment entries include:

- The bean provider must declare all the environment entries accessed from the enterprise bean's code.
- If the bean provider includes a value for the environment entry, the value can be changed later during assembly or deployment.
- During assembly, you can modify the values of the environment entries set by the bean provider.
- During deployment, you must ensure that the values of all environment entries are set to meaningful values.

EJB References panel

The EJB References panel lists all the enterprise bean references to the homes of other enterprise beans.

Each EJB Reference describes the interface requirements that the referencing enterprise bean has for the referenced enterprise bean.

Refer to “EJB References”.

Security Identity panel

The security identity panel (EJB 2.0)allows you to control security propagation by specifying your bean security identities when calling methods from other beans. When you call a method in EJB, the container propagates your security information by implicitly passing your security context within the stubs and skeletons. For example, you may have a client that is authenticated with the proper security identity. If the client calls bean A which in turn calls bean B, you may want to control whether or not bean B receives the client's security identity or whether it receives a different principal.

This panel provides two options for bean security identity:

- **Use caller identity**
- **Run as**

The following examples demonstrate usage for both options.

Use caller identity. You can control how security information is propagated in your deployment descriptor. The following code example takes the bean's security identity and propagates it to all the other beans it calls:

```
...
<enterprise-beans>
       ...
       <session>
            <ejb-name>AccountAdministration</ejb-name>
            <home>examples.AccountAdministrationHome</home>
            ...
             <security-identity>
                <use-caller-identity/>
             </security-identity>
             ...
       </session>
          ...
  </enterprise-beans>
```

To specify security identity for your bean, as shown above, check the “Specify Security Identity” checkbox, enter a description for your identity (if desired), and select the *Use caller identity* identity type from the drop-down list.

Run as. In comparison to the above example, you may want your bean to just propagate a role like “admin” to all other beans it calls. In this case you set up the bean's security identity to run as a pre-defined `<role-name>`. The following is a code example for Run as:

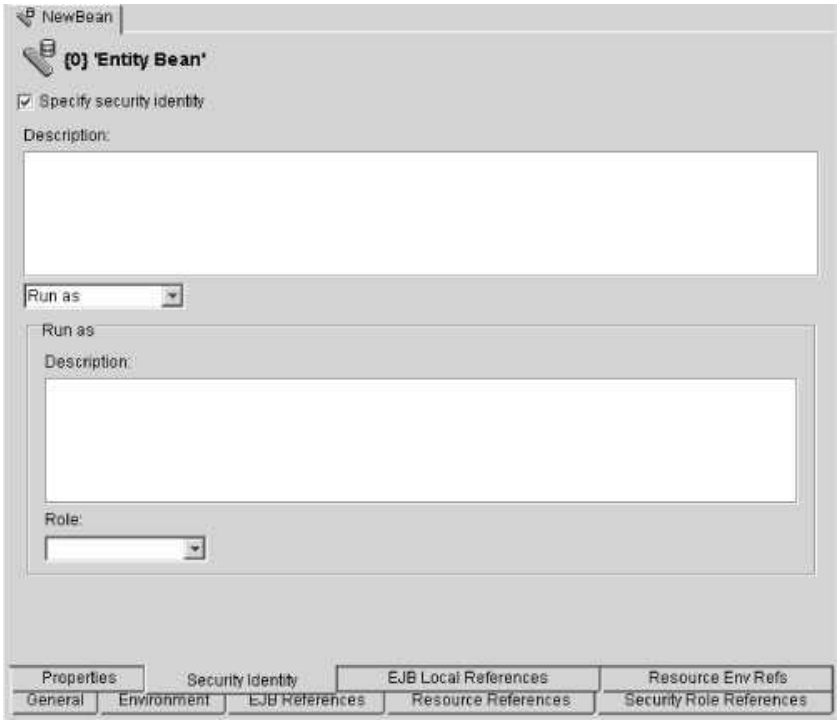
```
...
<enterprise-beans>
       ...
       <session>
            <ejb-name>AccountAdministration</ejb-name>
            <home>examples.AccountAdministrationHome</home>
            ...
             <security-identity>
                <run-as>
                      <role-name>admin</role-name>
             </security-identity>
             ...
       <session>
<assembly-descriptor>
       ...
       <security-role>
                <description>
                   This role is for people authorized to perform account
                      administration.
                </description>
                <role-name>admin</role-name>
       </security-role>
          ...
</assembly-descriptor>
</enterprise-beans>
```

If you choose this security identity option, you also must choose a run-as role from the *Run* drop-down list that appears. Roles are defined on the *Security Role References* tab. Roles are created in the navigation pane under “Security Roles”.

Note

Many beans do not require a security identity. Those that do not should leave the “Specify Security Identity” checkbox blank.

Figure 8.4 Security Identity Panel



Security Role References panel

The Security Role References panel lists all the enterprise bean's declared security roles and links to its “abstract security role”. The deployer might be assembling beans from different sources, bean developers, who may have declared security roles differently. The deployer uses the Security Role References panel to *link* the declared security role references to specific security roles defined by the application assembler or deployer.

For example, a specific bean developer may have defined the role-name for `AccountAdministration` as `administrator` while the deployer uses `admin` as the role-name. In this case the deployer will have to generate the security role that the application will use. Using the `<role-link>` element solves this, as shown in the following code sample.

```
...
<enterprise-beans>
       ...
       <session>
            <ejb-name>AccountAdministration</ejb-name>
            <home>examples.AccountAdministrationHome</home>
               ...
            <security-role-ref>
                <description>
                    This role is for people authorized to perform account
                      administration.
                </description>
                <role-name>administrator</role-name>
             <role-link>admin</role-link>
            </security-role-ref>
               ...
       </session>
       <assembly-descriptor>
```

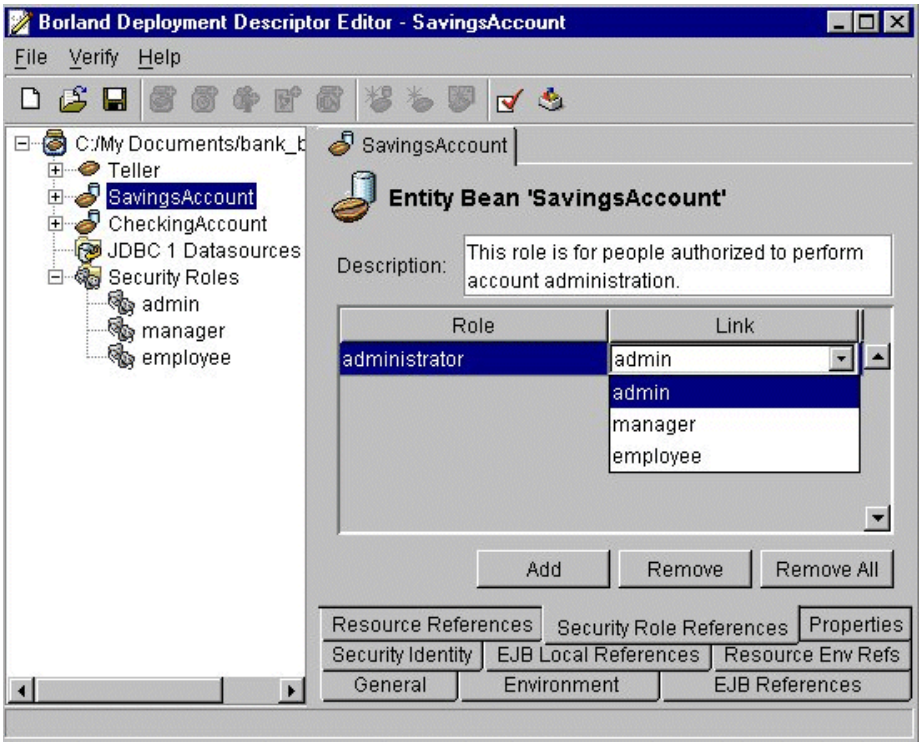
```
        ...
    <security-role>
        <description>
            This role is for people authorized to perform account
             administration.
        </description>
        <role-name>admin</role-name>
    </security-role>
        ...
  </assembly-descriptor>
</enterprise-beans>
```

To create security role reference links (*links* are the names of the security roles for the deployment of the application), you must first create a security role. To do this, refer to "Creating a security role".

After deployment roles are created, they should appear under the *Link* field in the Security Role References Panel.

To link this role to the name of the security role specified by the bean developer, enter the name specified by the bean developer in the *Role* field. Enter a description of the role if desired.

Figure 8.5 Security role references panel



Information in the panel includes:

- **Description:** This is an optional field that describes the Security Role.
- **Role:** This is the name of the Security Role specified by the bean developer.
- **Link:** This is the name of the Security Role used when the application is deployed. Typically, this role is defined by the application assembler or deployer to work in a specific operating environment.

Some things to remember about Security Role references include:

- The Security Role references used by the enterprise bean is a “generic” name that must be mapped to a specific operating environment.
- You can add or remove rows as necessary.

EJB Local References Panel (EJB 2.0 only)

The EJB Local References panel lists all the enterprise bean references to the homes of other enterprise beans within a single archive.

Each EJB Local Reference describes the interface requirements that the referencing enterprise bean has for the referenced enterprise bean. The reference contains:

- **Description:** A brief description of the bean that is referenced. This information is optional.
- **Name:** The name of the referenced bean. The tool tip information shows what you use to access the Name in code.
- **IsLink:** If checked, there is a link to a local enterprise bean and the following fields are filled in and read-only: Type, Home, and Remote. If unchecked, reference is to an enterprise bean outside the JAR file and you must fill in Type, Home, and Remote.
- **Link:** Links the EJB Reference to the target enterprise bean. The Link value is the name of the target enterprise bean. This information is optional.
- **Type:** The expected type of the referenced bean.
- **Local Home:** The expected Java type of the referenced bean's home interface.
- **Local:** The expected Java type of the referenced bean's local interface.
- **JNDI Name:** The JNDI name of the referenced bean.

Resource References panel

The Resource References panel lists all the enterprise bean's resource factory references. This enables the application assembler and/or bean deployer to locate all references used by the enterprise bean.

Information in the panel includes:

- **Description:** A description of the resource reference. This information is optional.
- **Name:** The name of the environment entry used in the enterprise bean's code.
- **Type:** The Java type of the resource factory expected by the enterprise bean's code. This is the Java type of the resource *factory*, not the Java type of the resource. The types available are:
  - javax.sql.DataSource
  - java.net.URL
  - javax.mail.Session
  - javax.jms.QueueConnectionFactory
  - javax.jms.TopicConnectionFactory

  The Deployment Settings (parameters) you can specify depend upon the Java type selected.

- **Authentication:** An Application authentication indicates that the enterprise bean performs the resource sign-on programmatically. A Container authentication indicates that the Container signs on to the resource (on behalf of the web application) based on the principal mapping information supplied by the deployer.
- **Sharing Scope:** The `<res-sharing-scope>` element specifies whether connections obtained through the given resource manager connection factory reference can be shared. The values of this element must be either `<Shareable>` or `<Unshareable>`. The default is `Shareable`.

Message-Driven Bean Panel

EJB 2.0 architecture supports Message-Driven Beans (MDBs). If you choose to create a new message-driven bean or descriptor for a message-driven bean, you will see the General Message-Driven Bean panel when you click on the bean in the Navigation pane. This panel allows you to set the following message-driven bean information:

- **Transaction Type:** whether or not the transaction is bean- or container-managed. If you select bean-managed transactions, you will also have to select one of the following acknowledge modes:
- **Auto-acknowledge:** automatically acknowledges a client's receipt of a message when it has either successfully returned a call to receive or the `MessageListener` it has called to process the message successfully returns.
- **Message Selector:** string of rules used by the client to specify only the messages it's interested in, by header. The syntax of the rules is based on a subset of the SQL92 conditional expression syntax. See the JMS Specification from Sun Microsystems for details.
- **Destination:** the JMS destination from which the message-driven bean instance consumes messages.
- **Connection Factory Name:** JNDI name of the connection factory used to establish a connection with the message broker.
- **Destination Name:** the JNDI name of the queue or topic to which the message-driven bean listens.
- **Destination Type:** specifies whether the destination is a queue, topic, or unspecified; javax.jms.Queue, javax.jms.Topic, or Not Specified.
- **Subscription Durability:** whether or not component subscriptions to the MDB remain beyond the scope of their initial connection.
- **Initial Pool Size:** specifies the initial number of message-driven bean instances that the container should create immediately after deployment.
- **Maximum Pool Size:** specifies the maximum number of message-driven bean instances that can be created and kept in the message-driven bean instance pool.
- **Wait Timeout:** specifies the number of seconds before timeout.

The module references editor

The DDEditor has a module references editor on the References tab. This tab displays interdependencies between EJB JARs, WARs, and clients within a module, as well as inter-module links or EJB dependencies between modules. References can be added or edited at the EAR, WAR, EJB JAR and Client JAR levels here.

The module references editor displays the following kinds of references (see Legend help for more information):

- Local EJB link
- Remote EJB link
- External local link or JNDI reference
- External remote link or JNDI reference
- Unresolved link or empty JNDI reference
- Resource reference

It also has several viewing options:

- **Module layout:** Displays modules in an EAR; it shows inter-module dependencies.
- **Tiered layout:** Displays a client/server tiered layout. The client resides on the left and is followed by server tiers to the right, session beans, entity beans, and data resources. The dependencies are displayed.
- **Circular layout:** Displays WARs, session beans, entity beans, clients, MDBs, resources and their dependencies in a circular format.
- **External nodes view:** You can toggle the external nodes references visibility, choosing to view them or not. The “Show All” button shows all nodes and references.
- **Legend:** Toggles a view of the references legend.

For each, you can zoom in, out or to fit, save your layout, or restore to default layout.

Using the module references editor

To add or edit references select a module under the Deployed or Hosted modules nodes in the Management Console, or open one from within the DDEditor. Select the References tab in the work pane.

Using the module references editor, you can:

- Add an EJB JNDI reference
- Add an EJB link
- Add a resource reference
- Add a resource environment reference
- Edit EJB reference and link remote and remote home interfaces
- Edit EJB reference and link local and local home interfaces
- Convert an EJB JNDI reference to a link
- Delete EJB references

To add an EJB JNDI reference:

Note

In many cases the remote or local interface information is automatically entered for a new or updated reference/link. Optionally, you can click the ellipses and choose an interface.

1. Right-click the EJB you want to add a JNDI reference to and select “Add EJB JNDI reference”.
2. Enter the JNDI name (location), such as `jndi:com/borland/examples/j2ee/hello/Hello` and the bean reference name, such as “Reference1”.
3. Select whether the EJB is an Entity or Session bean.
4. Select either Remote or Local and enter the home and remote or local home and local interfaces.
5. Click OK.

To add an EJB link:

1. Right-click the EJB you want to add a link to and select “Add EJB link”.
2. Go to the EJB you are linking to and click once. The New EJB Link panel will appear.
3. Enter the reference name of the EJB you are linking to, such as `ejb/local/CreditCard`.
4. The remote or local interface information is automatically entered. Optionally, you can click the ellipses and choose an interface.
5. Click OK.

To add a resource reference:

1. Right-click the EJB you want to add a resource reference to and select “Add Resource Reference”.
2. Enter the JNDI name, reference name and type, such as a JDBC resource or a JMS connection factory. In the Type drop-down menu choose from datasource, JMS topic or queue, MDB, and URL.
3. Select the authentication type:
   - **Application:** indicates that the enterprise bean performs the resource sign-on programmatically.
   - **Container**: indicates that the Container signs on to the resource (on behalf of the application) based on the principal mapping information supplied by the deployer.
1. Select the sharing scope:
   - **Sharable:** specifies whether connections obtained through the given resource manager connection factory reference can be shared. This is defined by the `<res-sharing-scope>` element. The default is `Shareable`.
   - **UnSharable:** indicates connections cannot be shared.
1. The description is optional.
2. Click OK.

To add a resource environment reference:

1. Right-click the EJB you want to add a resource environment reference to and select “Add Resource Environment Reference”.
2. Enter the JNDI name and reference name.
3. In the Type drop-down menu choose from JMS topic or queue.
4. The description is optional.
5. Click OK.

To convert an EJB JNDI reference to a link:

1. Right-click the JNDI reference (represented by dotted lines) you want to convert to a link and select “Convert to Link”.
2. In the “To” drop-down menu choose the bean to link to. Verify the interfaces in the interface type information
3. Verify the interfaces in the interface type information or choose interfaces by clicking the ellipses.
4. Click OK.

To delete or edit a reference:

1. Right-click the colored line representing the reference or link. To edit you can double-click the line.
2. Select delete or edit.
3. Click OK.

Container transactions

Enterprise beans that use container-managed transactions must have the Transaction Policies set by the Container. The DDEditor enables you to set container-managed Transaction Attributes and then associate these attributes with methods in the enterprise bean's home and remote interface.

Adding a container transaction

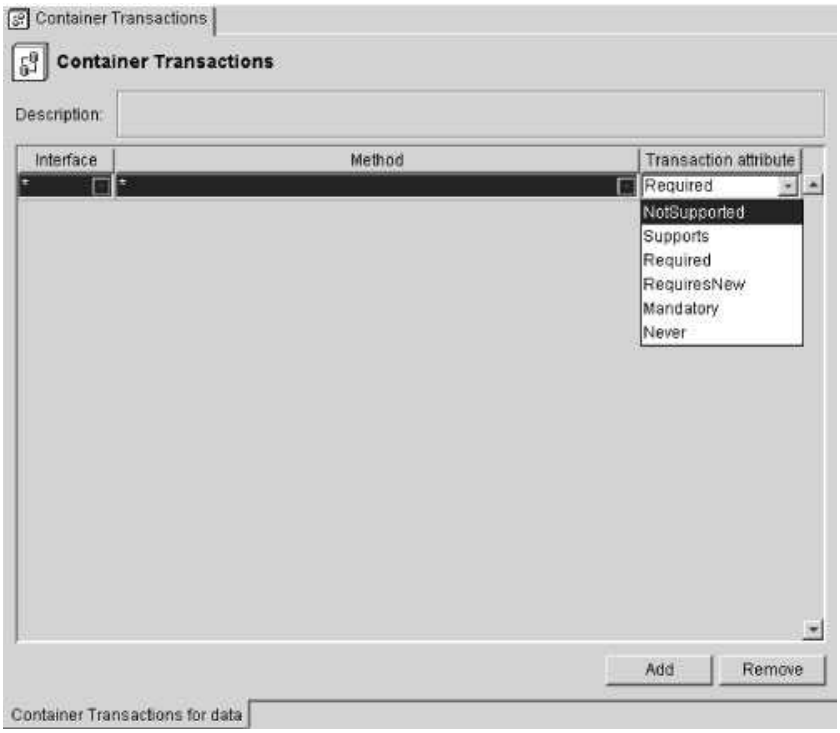
To add a container transaction:

1 In the navigation panel of the DDEditor, click on the enterprise bean so that all of its components are shown in the hierarchical tree.

2 In the navigation panel, click on Container Transactions for that enterprise bean.

The Properties panel appears for the transaction.

Figure 8.6 New container transaction



1 Enter a Description for the transaction.

2 Use the menu in the Interface column to choose either the home, remote, local, or localHome interface for the bean, or use * to specify all.

3 Use the menu in the Method column to select methods. This menu lists all methods for the interface specified in the previous column. Choose a specific method, or use * to specify all methods.

4 For each Interface and Method combination, select a Transaction Attribute from the menu.

Transactions supported include:

Attribute	Syntax	Description
Supports	TX_SUPPORTS	The bean is invoked with the same Transaction Policy as the client, if it has one. Otherwise, no transaction context is set.
Not Supported	TX_NOT_SUPPORTED	The bean is invoked without a transaction context.
Never	TX_BEAN_NEVER	An enterprise bean never invokes the User Transaction interface (javax.jts).
Required	TX_REQUIRED	The bean is invoked with the same Transaction Policy as the client, if it has one. Otherwise, the Container starts a new transaction before invoking a method on the bean. The Container commits the transaction upon returning from the method.
Requires New	TX_REQUIRES_NEW	The bean requires a new transaction. The bean is always invoked in a new transaction.
Mandatory	TX_MANDATORY	The bean requires a transactional context. The bean is invoked with the same Transaction Policy as the client, if it has one. Otherwise, an exception (javax.transaction.TransactionRequiredException) is thrown to the client.

Adding security roles and method permissions

The DDEditor enables you to create or edit security roles in the deployment descriptor. After Security Roles are created, you can then associate methods in the enterprise bean's home and remote interface with these roles, thus defining the security view of the application.

About Security Roles

Security roles are encountered in EARs, WARs and EJB JARs. They are intended to be logical roles that can be used in a variety of security environments. The advantage to using security roles is not having to hard-code specific identities into the beans since each environment has its own list of identities. This also makes it possible for you to modify access control without recompiling bean code. During deployment security roles are mapped to specific user groups and/or user accounts defined in the operating environment.

This section describes how to use the DDEditor to create security roles and assign enterprise bean method permissions to the roles. Defining security roles in the deployment descriptor is optional.

Creating a security role

To create a security role in the deployment descriptor:

1. In the navigation panel of the DDEditor, open the EAR, WAR or EJB JAR to which you want to assign roles.
2. Right-click the Security Roles folder and select New Role from the context menu.

   A dialog box appears.
3. Enter the name of the new Security Role and click OK.
4. The new Security Role appears in the Navigation pane.

To create a security role:

1. Select the Security Role in the Navigation pane.

   The Properties panel appears for the role. The information includes:

   - **Deployment role.** The standard `security-role-ref` element defines an alias to the `security-role` element allowing applications to refer to roles hard-coded in the bean code. Using role-refs allows deployers to change roles by adjusting the role-refs rather than having to update application code.

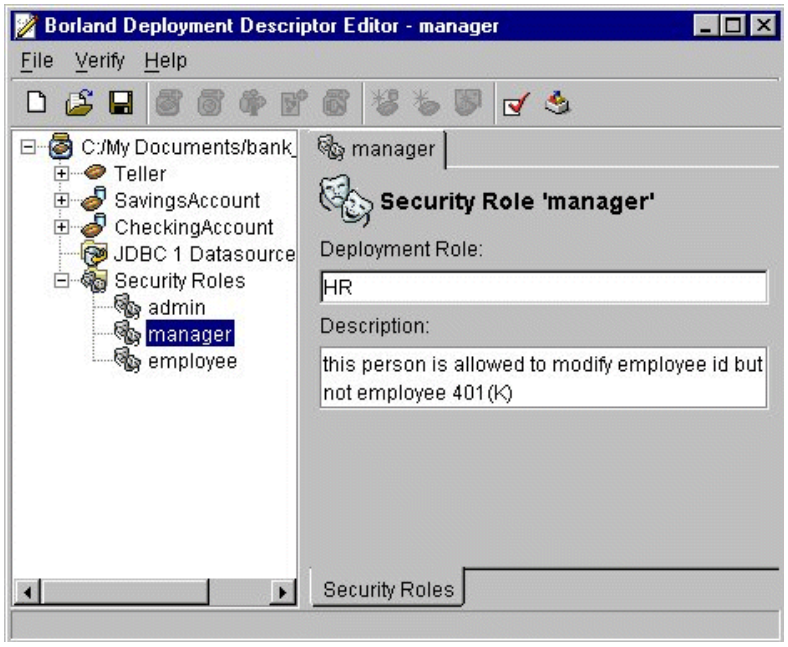
     Similarly, the `deployment-role` element is another alias for the `security-role` element. It allows users to maintain roles in the deployment descriptor independently of deployment roles that are defined in the authorization domain. Again, this allows deployers to map security roles, by using the `deployment-role` element, to existing roles defined in the appropriate authorization domain without changing the standard deployment descriptor.

     Note

     It is not necessary to use the `deployment-role` element to bind roles to the authorization domain. If there is no `deployment-role` defined for a `security-role`, the `security-role` is assumed to be defined in the authorization domain.

     The `deployment-role` element is a Borland-specific element and appears in the Vendor XML tab.
   - **Description.** This is an optional field, but enables the bean deployer to better understand the intent of the role.
2. Enter a Deployment Role.
3. Enter a Description for the role.

Figure 8.7 Security roles properties panel



Assigning method permissions

Once Security Roles are defined, you can specify which methods in the home and remote interface of an enterprise bean the Security Role is allowed to invoke. This is done using method permissions.

If the application assembler or bean developer defines security roles for the enterprise bean in the JAR file, they can specify the methods of the home and component interfaces that each security role is allowed to invoke. However, you are not required to associate a security role with methods in a bean's home or remote interface. Therefore, any security roles defined in the deployment descriptor are not allowed to invoke these methods unless associated with security roles in the bean's interface(s).

Some things to remember about method permissions:

- Each `method-permission` element includes a list of one or more security roles (only if there is “by Role” access) and a list of one or more methods. All listed security roles are allowed to invoke all listed methods. Each security role in the list is identified by the `role-name` element, and each method (or set of methods) is identified by the `method-name` element.
- A security role or method may appear in multiple `method-permission` elements.
- You can specify that some methods should not be checked for authorization prior to invocation by the Container. This is done by using the `unchecked` element instead of a role name in the `method-permission` element which indicates that a method won't be checked for authorization.
- If the method permission specifies both the `unchecked` element for a given method AND one or more security roles (if there are conflicting permissions set), the method won't be checked for authorization.
- You can use the `exclude-list`element to indicate the set of methods that will not be called. In this case, you should configure the enterprise bean's security such that no access is permitted to any method contained in the `exclude-list`.
- If a given method is specified in both the `exclude-list` element and in the method permission, you should configure the enterprise bean's security such that no access is permitted to the method.
- If you have multiple methods with the same name but that take different parameters, you can set permissions to distinguish between the two using the `method-params` element. This must be edited manually, directly in the XML.

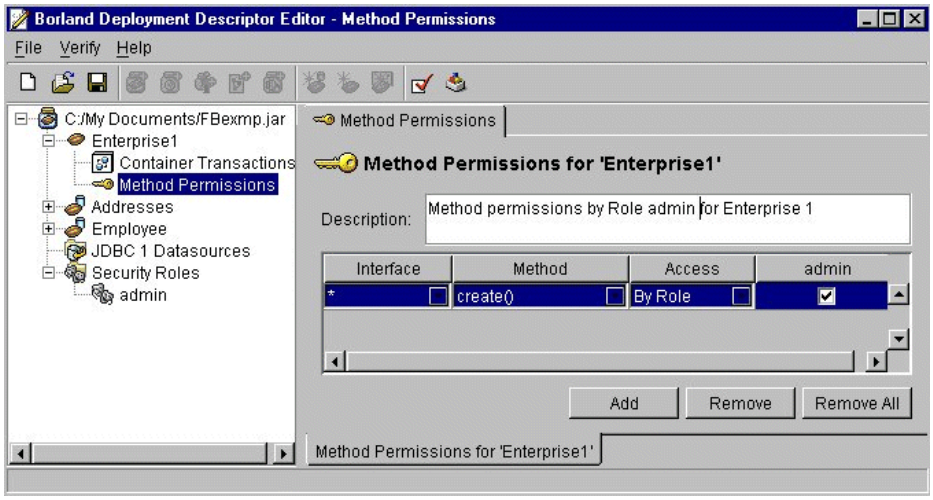
Note

Multiple methods with the same name but with different parameters are listed in the Method pull-down menu. For example, in the `SavingsAccount` bean of the `bank_beans`example, there are two `create` methods listed in the Method pull-down menu, `create` and `create(String, float)`.

To assign method permissions:

1. In the Navigation pane in the DDEditor, open the bean to which you want to assign method permissions.
2. Double-click Method Permissions from the bean's hierarchical menu.
3. In the Properties panel, enter a description. This is optional.
4. Click Add.
5. Select an interface in the Interface column to choose either the local home, local or remote interface for the bean, or select * to specify all.
6. Select a method from the Methods pull-down menu. This menu lists all methods for the interface(s) specified in the previous column. Choose a specific method, or select * to specify all methods.
7. Select a permissions level from the Access pull-down menu. You can choose from:
   - **By Role.** Choose one or more security role to specify which roles are allowed to invoke your selected method(s).
   - **Unchecked.** This specifies that some methods will not be checked for authorization prior to invocation by the Container. This element is used instead of a `role-name` element in the method permission.
   - **Excluded.** This is used to specify the `exclude-list` element, which specifies methods that will not be called.
8. Select which role you want associated with the method.
9. Continue to specify beans and methods as desired.

Figure 8.8 Specifying Method Permissions



Adding CMP 1.1 information

For entity beans, set persistence type to CMP 1.1 in the General tab to see the CMP node in the Navigation pane. To add container-managed persistence information:

Note

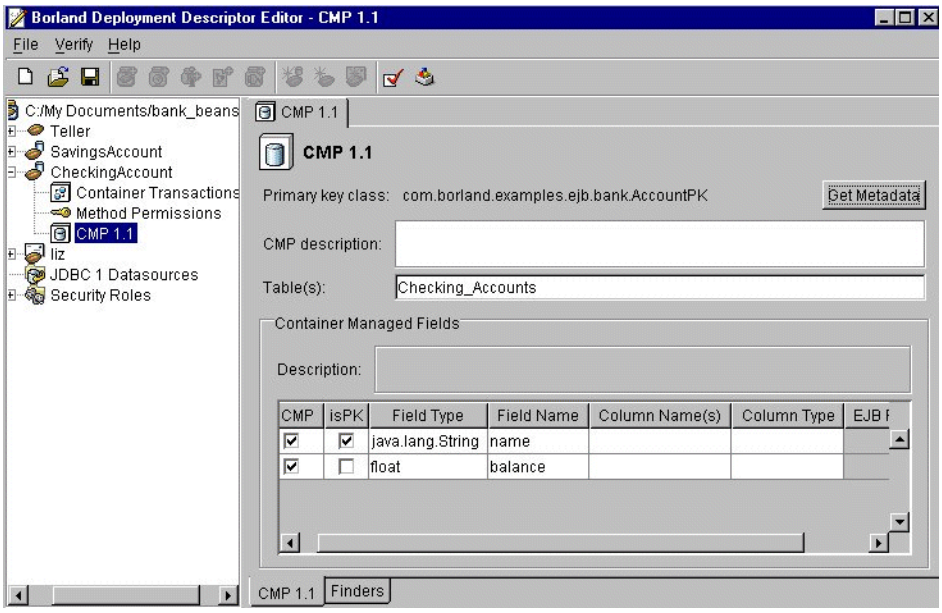
If you change the CMP version (1.1, 2.0), all previous CMP changes will be lost.

1. In the Navigation panel of the DDEditor, expand the enterprise bean so that all of its components are shown in the hierarchical tree.
2. Select the CMP 1.1 element to display its properties in the “CMP 1.1” pane.

Note

You can work with multiple tables and multiple columns.

Figure 8.9 CMP 1.1 fields



For beans with container-managed persistence (1.1), the following additional information is included to enable you to map fields in the bean to columns in a database table:

- **Get Metadata:** Queries the database to get a list of column names and types shown in a “Container Managed Fields” table.
- **CMP Description:** This is an optional field that describes the container-managed persistence.
- **Table(s):** The name of the database table(s) referenced by the bean.

Container Managed Fields

- **Description:** This is an optional field that describes a container-managed field.
- **isCMP:** A check mark indicates the field is container-managed.
- **isPK:** A check mark in the box indicates that this is a primary key. It also indicates the type of primary key class matches the field's class.
- **Field Type:** The data type of the field. If Type is a collection or subclass of EJBObject, there is an EJB Reference associated with it. In other words, Type refers to a foreign key. Collection foreign keys can match any EJB reference. Otherwise, the type must match the home interface of the EJB reference.
- **Field name:** Name of field in an entity bean.
- **Column name(s):** You can map compound fields in the bean (for example, location.street) to columns in the database table. You can either map the root field (for example, location) or the sub fields (for example, location.street), but not both.
- **Column type:** The column's type, for example, `CHAR(1)NOT NULL`.
- **EJB Reference:** If the field type is an EJB class, a menu appears with a list of EJB References to select. These references are set in the EJB References panel. “None” is a valid entry.

  The panel displays the primary key class name, which you can't change. In the CMP Description field, you can enter optional text describing the bean.

  The DDEditor uses JDBC to obtain metadata on existing tables. You can hook up entity beans to existing tables. For example, you might purchase a third-party enterprise bean and want to use it with a table in your database. To populate both the Column Name and Column Type fields, click the Get Metadata button and the metadata is retrieved.

Finders panel

The Finders panel only appears for EJB entity beans with container-managed persistence. It specifies the `WHERE` clause used by the CMP bean to execute finder methods defined by the bean.

Information includes:

- **Method:** The finder method name and a list of all arguments, for example, `findAccountsLargerThan(float balance)`.
- **Where Clause:** Specifies a SQL `WHERE` clause used by the Container to retrieve records from the database, for example, `balance > :balance`.
- **Load State:** A check mark enables the Container to preload all Container-managed fields whenever a `FIND` operation occurs.

Adding CMP 2.0 information

For information on CMP 2.0 refer to the section on the “The EJB Designer”. For information on CMP 2.0 in the AppServer, refer to Using Borland AppServer Properties for CMP 2.x in the *AppServer Developer's Guide*.

Adding a new datasource

The DDEditor enables you to specify a new data source for Entity beans and Containers and set the isolation level for the data transactions.

To add a new datasource:

1 Select an EJB JAR file in the Navigation pane and open it.

2 Right-click the JDBC 1 Datasources folder and select New Datasource from the context menu.

A dialog box appears.

3 Enter the JNDI name of the new data source and click OK. A new data source appears in the Navigation pane.

4 Double-click the datasource.

The Properties panel for the data source appears.

5 Enter information for the new data source. The information includes:

- **URL.** The URL is the location of the datasource.
- **User name and password.** These may be required to access the source.
- **Driver class name.** Class name of the JDBC driver. Clicking on ellipsespellipsesn a window with listed drivers.
- **Test Connection button.** This tests the connection to the datasource.

Note

A connection to the database will fail if the driver is not on your classpath. You must add the driver to the classpath using the Set Classpath command.

The data source is defined by a data source name, the URL location of the data source, and (if required) a user name and password to access the source. The panel also includes the class name of the JDBC driver and JDBC properties.

- Choose an Isolation Level from the pop-up menu. See the following section for details.

Isolation levels

Isolation level refers to the degree to which multiple, interleaved transactions are prevented from interfering with each other in a multiuser database. Possible transaction violations include:

- **Dirty Read:** Transaction t1 modifies a row. Transaction t2 then reads the row. Now t1 performs a rollback and t2 has seen a row that never really existed.
- **Non-Repeatable Read:** Transaction t1 retrieves a row. Then transaction t2 updates this row and t1 retrieves the same row again. Transaction t1 has now retrieved the same row twice and has seen two different values for it.
- **Phantoms:** Transaction t1 reads a set of rows that satisfy certain search conditions. Then transaction t2 inserts one or more rows that satisfy the same search condition. If transaction t1 repeats the read, it will see rows that did not exist previously. These rows are called phantoms.

The transaction isolation levels set in the deployment descriptor are defined based on the permitted violations defined above.

Attribute	Syntax	Description
Uncommitted	TRANSACTION_READ_UNCOMMITTED	Allows all three violations.
Committed	TRANSACTION_READ_COMMITTED	Allows Non-Repeatable Reads and Phantoms, but does not allow a Dirty Read.
Repeatable	TRANSACTION_REPEATABLE_READ	Allows Phantoms, but not the other two violations.
Serializable	TRANSACTION_SERIALIZABLE	Does not allow any of the three violations.

The EJB Designer

The DDEditor's implementation of the EJB Designer is primarily to facilitate the (O/R) mapping of CMP 2.0 entity beans' persistence schemas to a database, and for setting Borland-specific entity bean CMP properties for deployment. Many properties are disabled in the DDEditor's implementation of the EJB Designer as they require code change and must be changed in the source.

To use the EJB Designer effectively, you should have a schema already imported into the DDEditor in order to edit the CMP 2.0 properties. If you do not have it set up already, or do not have the vendor-specific XML imported, you will have to create and edit the table properties manually.

For more information on the EJB Designer, refer to the white paper, “DDEditor 6.7 Support for configuring EJB CMP 2.X Deployment Descriptor” at http://support.borland.com/entry.jspa?externalID=4311&categoryID=391.

Borland-specific deployment descriptors

For the Borland-specific DTD information, see the DTD documentation in the online Help.

9

Using the JNDI Browser

You can use the Management Console's JNDI Browser to manage JNDI. JNDI allows applications to look up objects, such as EJBs, datasources and JMS connection factories and destinations by name. JNDI defines serial provider interfaces to existing naming services such as CosNaming, LDAP, DNS, file system, and RMI registry. The serial providers have bindings, which relate names to objects and provide lookup capabilities to objects by name. This allows components to locate each other throughout the application.

Each naming service provider can be accessed using the factory icons in the left-hand column of the JNDI Browser. Each naming service in the JNDI Browser can be created, configured, or deleted by using the context menu and toolbar icons for each provider node in the navigation tree. Each service in the tree provides a view of pre-configured general properties in the work pane. The root context and properties are configured here. In addition, for each service, you can create a new provider or a set of default providers.

Java Resource Objects are typically deployed into the CosNaming namespace. You can view JRO properties by selecting a Java Resource Object from the tree and clicking the JRO Properties Tab. This Tab also displays the properties of the Java Resource Definition Object associated with the container-instantiated JRO. The Object Type field in the work pane will tell you whether or not an object selected in the tree is a JRO.

Setting the Classpath

In order to show JRO information in the JNDI Browser's work pane, class libraries may need to be made available to it. To add libraries to the Console's classpath:

1. Choose File|set classpath. The Classpath Editor appears.
2. You can add archives or paths (for exploded archives) to the Console's classpath. Click the corresponding button to add either an archive or a path to an exploded archive. A file-system browser appears.
3. Choose the archive or path from the file browser.
4. Continue Adding archives and/or paths until you are finished.
5. Click OK. The classpath is updated.

Note

You may need to click the Refresh button after configuring the classpath.

Creating Service Provider Views

You can create a set of default providers, or you can create a new provider that you configure during creation.

Note

CosNaming and Serial have default providers supplied.

To create provider defaults:

1. Select the appropriate factory from the factory icons.
2. Right-click the factory node in the navigation tree and select Create Default Providers. The default provider appears in a node below the factory.

To create a new provider:

1. Select the appropriate factory from the factory icons.
2. Right-click the factory node in the navigation tree and select New <factory> Provider. The provider properties configuration dialog opens.
3. Configure the provider. Refer to the appropriate configuration section for more information about configuring each provider in “Configuring Service Provider Views”.

Configuring Service Provider Views

Each provider has a select set of properties that must be configured, see the appropriate section below for information about configuring a particular provider.

CosNaming

The CosNaming Factory points to a running server's user port (Smart Agent port) on the Borland AppServer (AppServer). Expanding the default VisiBroker provider tree should reveal the default VisiBroker root context entry with the default Smart Agent port (14000).

To configure the provider:

1. Click the provider node in the tree. The CosNaming Provider Properties opens in the right pane.
2. You can configure the following properties:
   - Display name
   - Use java.naming.provider.url as display name
   - Root context
   - Use VisiBroker CORBA ORB
   - Smart Agent port
   - Use default port
   - Set an identity for security checks
1. You can also configure additional properties by clicking the Add button next to the properties table in the lower half of the CosNaming Provider Properties dialog.

Additionally, if this provider is to be used as a AppServer provider, you must add a property in order for AppServer Partitions to find the naming service. To configure the provider to be a AppServer Naming Service:

1. Right-click the provider node in the tree, and select Configure. The CosNaming Provider Properties dialog opens.
2. In the properties section of the dialog, click the Add button. The Add JNDI Property dialog appears.
3. From the drop-down list, select the property `SVCnameroot`.
4. Enter `namingservice` in the Value field.
5. Click OK. The Add JNDI Property dialog is dismissed.
6. Click OK.

Creating a subcontext

In the CosNaming Factory you can create a subcontext under the provider. JNDI entries are placed in a hierarchical order under subcontexts, similar to a file/subdirectory system. Using subcontexts provides a way to organize and group the entries.

To create a subcontext, right-click the context folder under the provider and select Create subcontext.

Note

If you delete a node in the CosNaming view, the entry is no longer accessible. Redeploying the archive restores the deleted (or renamed) serial entries. In most cases you have to restart the Partition to restore the redeployed archive.

Serial

Serial is Borland's proprietary naming service for AppServer. In BAS 6.7, JROs that were previously serviced by the Serial provider are now serviced by CosNaming. If you have a previous version of AppServer and/or simply wish to continue using the Serial namespace as before, you can enable the Serial provider view and continue to use the provider by setting a property for the Partition hosting the naming service.

To enable the Serial provider view for the JNDI Browser:

1. Select File|BES Serial view
2. The Serial icon appears in the provider list.

To enable the Serial provider for JROs:

1. Open a Borland Management Console on the host local to the Partition hosting the Serial.
2. Click the Installation button. The Local View appears in the Content pane.
3. Expand the node for the local agent.
4. Locate and expand the local agent's Configurations node.
5. Select the Partition you want to host the Serial.
6. In the Content pane, select the Files tab. A list of configuration and properties files appears in the Structure pane.
7. Select the `partition_server.config` file from the Structure pane. The file is opened for editing in the Content Pane.
8. Add the following line at the end of the file:

```
vmparam -DuseSerialForResRefs=true
```

9. Click the Save Icon in the Content Pane.

10 Restart the Partition.

To disable the Serial, change the value of `useSerialForResRefs` to `false` and restart the Partition.

To configure the provider:

1 Right-click the provider node in the tree, and select Configure. The Serial Provider Properties dialog opens.

2 You can configure the following properties:

- Display name
- Use java.naming.provider.url as display name
- Root context
- Use VisiBroker CORBA ORB
- Smart Agent port
- Use default port
- Set an identity for security checks

1 You can also configure additional properties by clicking the Add button next to the properties table in the lower half of the Serial Provider Properties dialog.

Additionally, if this provider is to be used as a AppServer provider, you must add a property in order for AppServer Partitions to find the naming service. To configure the provider to be a AppServer Naming Service:

1 Right-click the provider node in the tree, and select Configure. The Serial Provider Properties dialog opens.

2 In the properties section of the dialog, click the Add button. The Add JNDI Property dialog appears.

3 From the drop-down list, select the property `SVCnameroot`.

4 Enter `namingservice` in the Value field.

5 Click OK. The Add JNDI Property dialog is dismissed.

6 Click OK.

LDAP

The LDAP naming service points to an LDAP server such as `ldap://directory.company.com`. You need to specify the root context, for example, `o=borland`.

To configure the provider:

1 Right-click the provider node in the tree, and select Configure. The LDAP Provider Properties dialog opens.

2 You can configure the following properties:

- Display name
- Use java.naming.provider.url as display name
- Root context

1 You can also configure additional properties by clicking the Add button next to the properties table in the lower half of the LDAP Provider Properties dialog.

File System

This naming service is like the Windows File Explorer with a short cut to a specific directory. You need to specify a display name and a file directory, which will appear in the tree.

Note

If you physically delete a node in the File System view, it is deleted from the machine. However, if you delete a provider, only the JNDI provider definition is deleted, not the files and directories it points to.

Modules (EARs, JARs, etc.) that appear in a directory tree are editable by right-clicking on them, which launches the Application Assembly Tool. For more information on the Application Assembly Tool, see "Using the Archive Tool."

The naming definition `.binding` files appear in the tree as JMS services and JDBC datasources.

To configure the provider:

1. Right-click the provider node (Drive C:\ for example) in the tree, and select Configure. The File System Provider Properties dialog opens.
2. You can configure the following properties:
   - Display name
   - Use root directory as display name
   - Root drive and directory

RMI Registry

When you point the browser to your RMI service, for example `rmi://<your_host_name>:1099)`, you should see your RMI registry entries. Make sure the port you choose has an RMI server running on it. Otherwise, you will get an error.

To configure the provider:

1. Right-click the provider node in the tree, and select Configure. The RMI Registry Provider Properties dialog opens.
2. You can configure the following properties:
   - Display name
   - Use java.naming.provider.url as display name
   - Root context
1. You can also configure additional properties by clicking the Add button next to the properties table in the lower half of the RMI Registry Provider Properties dialog.

Network DNS

The default Network DNS naming service assumes a local machine as a DNS server, such as `dns://dnshost:53/domain_name.com`.

To configure the provider:

1. Right-click the provider node in the tree, and select Configure. The Network DNS Provider Properties dialog opens.
2. You can configure the following properties:
   - Display name
   - Use java.naming.provider.url as display name
   - Root context

1 You can also configure additional properties by clicking the Add button next to the properties table in the lower half of the Network DNS Provider Properties dialog.

Deleting providers

To delete providers:

1 Right-click on the provider node to open the context menu.

2 Click Delete. A confirmation dialog opens.

3 Click Yes to delete this provider node.

10

Using the Archive Tool

The Borland AppServer (AppServer) includes an Archive Tool that lets you perform a variety of tasks on archives that will be deployed on one or more servers, such as:

- Adding files to an archive
- Editing archive files
- Removing archived files
- Creating archive files
- Verifying archive files

The tool can open a variety of archive files including:

- Enterprise Application Archives (EARs)
- WEB Application Archives (WARs)
- Resource Adapter Archives (RARs)
- EJB JARs
- Client Jars
- Zip files
- Datasource Archive (DAR), a Borland proprietary archive format that stores JNDI definitions

The Archive Tool lets you create a variety of archive files including:

- Enterprise Application Archives (EARs)
- EJB Jars
- WEB Application Archives (WARs)
- Client Jars
- Connector modules
- Library files
- J2EE version 1.2 archives (EARs, EJB, Web, and client-side Jars)

The tool also lets you extract files from an archive and verify archive files at the assembly level.

The Archive Tool has a number of practical uses. It not only enables you to view and edit existing files in the archive; but lets you quickly assemble archives. For example, if you have forgotten to add an image file to a JAR file, you can use the Archive Tool to add the image file to the JAR without having to rebuild the JAR.

Starting the Archive Tool

The Archive Tool can only be started from the Console:

1. Start the Console.
2. From the Tools menu, select Assembly Tool, or click the Assembly Tool icon in the toolbar.

With the Archive Tool open, you can begin to add files to an (as yet unnamed) archive. Each archive, when created, contains a manifest file which is automatically generated by the Archive Tool.

Opening an archive

To open an archive:

1. Open the Archive Tool.
2. Select the Open option from the File menu.
3. Navigate to the archive file that you want to use, select it, and click OK.

The archive opens and its contents are displayed.

Performing actions on an archive

Once the archive is created, you can perform a variety of actions on it and the files that it contains.

Adding files to an archive

Files can be added to a new or existing archive. To add files to an archive:

1. Open the Archive Tool.
2. Open an archive, then select Add from the Module menu.
3. Select the files to add to the archive.
   - To add one or more files, click Add Files.
   - To add the contents of an entire directory, click Add Directory.
1. Navigate to the file or directory that you want to add, select it, and click OK.
2. Choose either the Relative path (the default) or Absolute path radio button. When constructing J2EE archives, you should use relative paths only.
3. Leave the Compress entries box checked (the default) unless you do not want to compress the files you are adding to the archive.
4. If adding a directory, leave the Include subdirectories box checked (the default) if you also want to recursively add the directories and files contained in the directory you are adding.
5. When you have finished adding entries, click OK.

Editing text files in archives

Once a text file has been added to an archive, it can be edited without the need to extract it from the archive. To edit a text file:

1. Open an archive.
2. Double-click on an editable file in the archive to open it in a text editor window.
3. Edit the file as needed, then click Save.

Removing files from an archive

To remove files from an archive:

1. Select the files to be removed.
2. Select Remove from the Module menu.

Saving an archive

To save an archive file, select Save from the File menu.

To save an archive that you have created with a different name:

1. Select the Save As option from the File menu.
2. Specify the name and the location of the file to be saved.
3. Click Save.

Editing archive files

To edit any type of file in an archive:

1. Open the archive.
2. Select the file and click the View file contents icon in the toolbar.
3. Enter the required changes.
4. Click Save.

Creating archives

To create a new archive file:

1. Select New from the File menu.
2. On the New Module Wizard, select the type of archive you want to create and the J2EE version that you want to use, and click Next.

   The New Module Wizard asks you to supply additional details that apply to the type of archive you have selected. The details you are asked to supply vary depending on the archive type.
3. When you are done supplying the details, click Finish.

   The New Module Wizard creates the fundamental foundation files required for the archive type selected and opens it as an untitled archive. If you make unsaved changes to an archive file, the Archive Tool prompts you to save it before it opens the new archive.

Extracting modules

The Application Assembly tools lets you uncompress one or more files in an archive.

To extract modules in an archive:

1 Open an archive.

2 You can either:

- Leave all the modules unselected if you want to extract them all.
- Click on one or more modules if you want to extract only selected modules.

1 Select Extract from the Module menu.

2 In the Output directory field, enter the location where you want to store the extracted files or use the Browse button to find a location, and click OK.

Verifying archives

The Application Assembly tools lets you check an archive file for correctness and consistency, and you can verify that all of the elements required for deploying your application are in place. After verifying archives individually and ensuring there are no errors, the assembly level verification involves verifying other resources that will be built into an application. For example, the Archive Tool will verify the existence and correctness of URIs (Uniform Resource Identifiers), but not EJB or JNDI links.

Supported archive types

Supported archive types are: EAR, WAR, JNDI, RAR, Client Jars, DAR, and ZIP.

The typical archive verification process includes the following checks:

- A pass over the XML code, checking for correct XML syntax.
- Verification of the semantics of the standard and proprietary XML descriptors, and the compliance with the required descriptors for each supported archive type.

Verification always occurs in a hierarchical fashion, starting with the top module. If verification is correct, verification continues recursively working through submodules and finally checking for inter-archive links.

Verifying an archive

To verify an archive:

1. Open the archive.
2. Select Verify from the Module menu.
3. In the Verify Module dialog, choose the verification Role level:
   - **Developer:** This is the lowest verification level. All XML is checked for syntax as well as standard and proprietary keywords relevant to the current archive type. Consistency across the archive is checked, but no external resources are verified at this level.
   - **Assembler:** Once the archives are individually verified and are correct, other resources built into an application will start to be verified. For example, this level will verify the existence and correctness of URIs, but not EJB or JNDI links.
   - **Deployer:** (the default) All checks are turned on. EJB and JNDI links are checked at this level as well as the operational environment in which the application is to be deployed.
1. Specify the additional options, if desired, then click OK. The Verifying dialog opens and displays the status of the verification process.

If errors and warnings are found, they are displayed in the Verifying dialog.

11

Using the License Manager

You can use the Management Console's License Manager to add, register, and remove your Borland product licenses. This section describes how to use the License Manager GUI features.

Borland AppServer uses **node-locked licenses.** These licenses are locked to the system on which they are applied and activated. You cannot copy these licenses to another system, and they cannot be accessed from a AppServer product running on another system. If you reinstall the software on the same system, you will need to reactivate the license.

Starting the License Manager

The License Manager can be started from the Management Console:

1 Start the Management Console.

2 From the Tools menu, select License Manager.

The License Manager can also be started by running an executable:

Windows

– To launch the License Manager on Windows, run the `lmadmw` executable found in the `<install_dir>\bin` directory.

UNIX

– To launch the License Manager on UNIX, run the `lmadm` executable found in the `<install_dir>/bin` directory.

Viewing license information

The License Manager displays the licenses you have applied to products on the local machine. To see more detailed information about each license, click on the license node in the left-hand navigation tree. The content pane on the right displays product details and operations that you can perform on the selected node.

Adding licenses

In some cases, you will need to apply multiple licenses to one installation. This will happen if you purchase optional products that can be added onto a base product. Applying a second, third, etc license to the same product installation is similar to the first.

Before you can add a new node-locked product license you will need the following:

- **The Serial Number and Key, or the activation file for each license.** These will arrive in an email from Borland.
- **A Borland Developer Network (BDN) account.** If you don't have an account, you will be able to create it during the registration/activation process.
- **Internet access.** This is required if you want to use the Direct or Web page methods for activating your license. This can be from any system—not necessarily the system on which your product is installed.

To add a product license, do the following steps:

1. Select Add from the Serial menu. The Add Serial Number dialog opens.
2. Enter the Serial Number and license Key, and click OK. The new serial number will appear in red letters in a list of Unregistered serial numbers in the License Manager navigation tree.
3. Select the New serial number, and select Register from the Serial menu. The Borland Product Registration Wizard steps you through the registration process. For more information about using the wizard see Borland product registration wizard in the *AppServer Installation Guide*.

Importing licenses

In some cases, you will need to import a license from a file. To import a product license, do the following steps:

1. Select Import from the License menu. The Import License dialog opens.
2. Locate the license file using the Import License navigation tool, and click OK.

12

Using Optimizeit Profiler and ServerTrace

Borland Optimizeit Profiler and ServerTrace are integrated with the Management Console for use with application servers (such as Borland AppServer) configured as Managed Objects.

To use Optimizeit Profiler or ServerTrace in the Management Console you must do the following steps:

1 Install and configure the local Profiler/ServerTrace client.

2 Configure the Profiler/ServerTrace server-side settings.

See "Using Profiler/ServerTrace features in the Management Console" for information about using the Optimizeit Profiler and ServerTrace features.

Note

This integration supports Optimizeit ServerTrace version 3 onwards and Optimizeit Profiler 6.0 onwards.

Install and configure the local Profiler/ServerTrace client

To configure the local Profiler/ServerTrace client:

1 Install Optimizeit Profiler and/or ServerTrace on the machine on which the Management Console is running.

Note

If you are using the Management Console to manage a remote agent, you must also install Profiler/ServerTrace on the machine on which the agent is running.

2 From the Console menu in the Borland Management Console, select Preferences, then select the Tools tab.

3 Enter the absolute (fully qualified) path to the client executables installed on the local machine (the machine on which the Management Console is running).

   Optimizeit Profiler requires a path to Optimizeit.exe and EditFilter.exe. Optimizeit ServerTrace requires a path to ServerTrace.exe. Enter the paths explicitly, or click the Browse button to locate the executables.

4 Click OK when you are finished.

Note

You may receive the following error message when configuring Profiler on more than one BAS Partition: “The port 1470 is already used, the audit system cannot start.” If you see this error, use the Audit System Selector utility to explicitly assign a different port number so that the next Partition chooses the new port. To change the port, edit the following audit-port entry in the `optimizeit.xml` file for the partition:

```
<audit-port>
   <port-range begin="1473" end="1483" />
</audit-port>
```

Configure the Profiler/ServerTrace server-side settings

To configure the Profiler/ServerTrace server-side settings:

1 Select the Management Console Hubs view if it is not already active.

2 Expand the navigation tree to the node that represents the AppServer Partition (or other supported application server) on which you want to use Profiler/ServerTrace.

3 Right click on the Partition node, select Optimizeit, then select Configure.

4 In the Mode drop-down list select the appropriate start mode.

   – **Normal—Non Optimizeit:** The default start mode where Profiler and ServerTrace do not start with the Partition.

   – **Profiler—Start with Optimizeit Profiler:** Select this start mode to run the Partition with Optimizeit Profiler.

   – **ServerTrace—Start with Optimizeit ServerTrace:** Select this start mode to run the Partition with Optimizeit ServerTrace.

1 Select either the Profiler or ServerTrace tab, depending on which mode you selected.

2 Enter the absolute (fully qualified) path to the Optimizeit Profiler or ServerTrace installation directory on the remote machine (the machine on which the Partition running) in the ServerTrace home or Profiler home field. For example:

```
C:\Borland\Optimizeit\ServerTrace3\
```

3 Click OK to deploy the Optimizeit Profiler/ServerTrace archive.

4 Right click on the Partition node, select Optimizeit, then select Configure.

5 To generate the ServerTrace EJB file, select EJB File in the files menu, and click Generate. Once you generate this file you can edit it.

6 Edit the Profiler/ServerTrace Master Configuration, Filter, Action, or EJB files by selecting the file in the Files list and clicking Edit.

   See the Optimizeit Profiler or ServerTrace documentation for information about how to configure each of these files.

Note

When ServerTrace is started, it also starts an SNMP agent. This SNMP agent has two port settings. These port settings are specified in the ServerTrace Master configuration file. There could be conflicts between ports used by ServerTrace and other systems on the same host. So, be sure to set the ports to some non-used ports. Default SNMP ports are 161 for the agent and 162 for the traps. Edit the following SNMP and trapPort entry in the `optimizeit.xml` file to some unused port numbers, for example:

```
<snmp agentPort="161" trapPort="162" confPath="adm/servertrace/straceSNMP" />
```

7 Click OK when you are finished.

Note

If the Partition is running when you click OK, the Management Console will prompt you to restart the Partition. If you choose Yes, the Partition will restart, but it will run more slowly.

Using Profiler/ServerTrace features in the Management Console

This section describes how to access and use the Optimizeit Profiler and ServerTrace features integrated with the Management Console. Refer to your Optimizeit Profiler and ServerTrace documentation for information about terminology and features. The following features are supported:

- Hibernate and Wakeup
- Generate Snapshot
- Attach
- Clear Statistics
- View Snapshot

Accessing Profiler/ServerTrace features

To access the integrated Profiler/ServerTrace features:

1 Right-click the Partition node (or other supported application server node) and select Optimizeit.

2 Select the appropriate feature in the sub-menu.

Attaching a partition process to the Profiler

To attach a partition process to the Optimizeit Profiler:

1 Right-click on the partition name in the left pane and select Optimizeit->Configure….

2 Select Profiler—Start with Optimizeit Profiler from the Mode drop-down menu.

3 Enter the path to the Optimizeit Profiler executable in the Profiler home text box.

4 Click on yes when you get a prompt to restart the partition.

5 In the Management Console, go to Console->Preferences.

6 Click on the Tools tab to bring it forward.

7 Right-click on the partition node in the left pane and select Optimizeit -> Attach.

8 In the Edit settings dialog, click the Remote Application radio button.

9 Enter the name of the machine that is hosting the partition you want to attach and the port number.

10 Click the Attach now button.

Hibernate and Wakeup

The Hibernate feature allows you to put ServerTrace into hibernate mode where it is still collecting information on the server but it stops reporting. This option is enabled if the Partition is running and the Partition is enabled for ServerTrace.

To put ServerTrace in hibernate mode, select Hibernate in the Optimizeit sub-menu. To return ServerTrace from hibernation, select Wakeup in the sub-menu.

Generate Snapshot

Generate Snapshot allows you to generate a snapshot of the server. This option is enabled if the Partition is running and the Partition is enabled for ServerTrace.

To generate a snapshot:

1. Select Generate Snapshot from the Partition's Optimizeit right-click menu.
2. Enter a reason for the snapshot, or any other information that could identify this snapshot.

   The text in the Reason field is incorporated into the filename for the snapshot.
3. Optionally enter a comment for the snapshot. This comment will be included in the meta information for the snapshot.

Attach

Use the Attach option to open the Profiler or ServerTrace client with a current view of the server. This option is enabled if the Partition is running and is enabled for Optimizeit Profiler or ServerTrace.

Clear Statistics

Clear Statistics allows you to clear the data on the ServerTrace audit system. When a new snapshot is taken it does not contain old information. It does not delete previously generated snapshots. This option is enabled if the Partition is running and is enabled for ServerTrace.

View Snapshot

Use View Snapshot to open a generated snapshot in the ServerTrace client. This option is enabled if the Partition is running and is enabled for ServerTrace.

To view a snapshot:

1. Select View Snapshot from the Partition's Optimizeit right-click menu.
2. Select a snapshot from the list, and click OK.
3. To save a snapshot locally, do the following additional steps:
   a. Check Save to local file.
   b. Enter the path (or use the Browse feature) to the directory where the snapshot should be saved.
   c. Click OK.

13

Using the JMX console

A JMX console, MC4J Management Console, is provided as a 3rd party management option for JMX-enabled objects such as the Borland AppServer (AppServer) Partition. To start the JMX console right-click on a Partition (or other J2EE server) and select Launch JMX Console.

Note

If you are using an external JDk instead of the default JDK (such as in the case of HP-UX), you may encounter a problem while launching MC4J from the Management Console. To avoid this problem, edit the following line in the mc4j.config file in the <install_dir>/bin directory to point to the right JDK:

```
javahome $var(installRoot)/jdk/jdk1.4.2
```

The MC4J Management Console provides a tree-structure view of the MBeans associated with the AppServer Partition. See `http://mc4j.sourceforge.net/guide/index.html` for the MC4J User Guide.

Launching the JMX console standalone

The MC4J Management Console can also be run standalone (that is, without running the Borland Management Console). For AppServer there are two ways to accomplish this:

- Using the JMX Service URL
- Using the Borland AppServer Smart Agent

Using the JMX Service URL

1 Copy the contents of the jmxservice.url file (say in Notepad) and copy the contents into the clip board. The jmxservice.url file is located in

```
<install_dir>/var/domains/<domain_name>/configurations/<config_name>/mos/
   <mo_name>/
```

To locate jmxservice.url for a AppServer partition, the Partition must be run after it is added to a configuration. Also <domain_name> is **base** and <mo_name> is ***Partition** (such as **PetstorePartition**). For example:

```
<install_dir>/var/domains/base/configurations/j2eeSample/mos/
PetstorePartition/
```

2 Launch MC4J at the command prompt:

```
prompt% mc4j.exe
```

This executable is located in the <install_dir>/bin/ directory.

3 In the Connection Explorer, right-click the MC4J Connections node and select Connect to Server. A wizard will come up.

4 Choose BorlandServer from the drop-down list.

5 Enter a name for the connection.

6 In the Server URL field, paste the contents of jmxservice.url file that you copied in step 1.

7 If your management domain is secure, supply the login and password in the respective Principle and Credentials fields. Please note that by default the AppServer management domain is secure, and the login and password are `admin` and `admin`.

8 Click Next to go to the Customize classpath step, and add the following JARs from `<install_dir>/lib/`

- asrt.jar
- common.jar
- dom4j.jar
- jaas.jar
- jafa.jar
- jsse.jar
- lm.jar
- log4j.jar
- mail.jar
- sanct4.jar
- vbejb.jar
- vbjorb.jar
- vbsec.jar
- xercesImpl.jar
- xmlParserAPIs.jar
- xmlrt.jar

1 Click Finish. The connection will appear as a new node in the Connection Explorer.

Using the Borland AppServer Smart Agent

1. Edit <install_dir>/bin/mc4j.config file and set the property `vmprop bes.no_osagent=false`.
2. Using the command-line, set the `osagent` port to your management port, and use `osfind` to discover the name of your JMX agent (for example `ojms_JOHNDOE_openjms_JmxAgent`).
3. Start <install_dir>/bin/mc4j.exe (from the same shell where you have the right `OSAGENT_PORT` set).
4. In the Connection Explorer, right-click the MC4J Connections node and select Connect to Server. A wizard will come up.
5. Choose BorlandServer from the drop-down list.
6. Enter the name for the connection from step 2.
7. You do not need to make any changes in the Server URL text field as it is not used.
8. If your management domain is secure, supply the login and password in the respective Principle and Credentials fields. Please note that by default the AppServer management domain is secure, and the login and password are `admin` and `admin`.
9. Click Next to go to the Customize classpath step, and add the following JARs from `<install_dir>/lib/`
   - asrt.jar
   - common.jar
   - dom4j.jar
   - jaas.jar
   - jafa.jar
   - jsse.jar
   - lm.jar
   - log4j.jar
   - mail.jar
   - sanct4.jar
   - vbejb.jar
   - vbjorb.jar
   - vbsec.jar
   - xercesImpl.jar
   - xmlParserAPIs.jar
   - xmlrt.jar
1. Click Finish. The connection will appear as a new node in the Connection Explorer.

Using the HTTP adaptor to monitor JMX-enabled objects

Alternatively, if you have enabled the HTTP adaptor and XSLT processor in the Partition's JMX Agent configuration, you can use the Web browser to monitor AppServer. The default URL for the HTTP adaptor is `http://localhost:8082`.

See JMX support in Partitions in the *AppServer Developer's Guide* for more information about the MBean instrumentation provided for the AppServer Partition.

14

Using Partitions

This section discusses how to use the Management Console to work with Partitions for Borland AppServer (AppServer). It describes the following tasks:

- Creating, cloning, and deleting Partitions
- Deploying modules to a Partition
- Configuring an existing Partition
- Viewing Partition information
- Tuning Partitions for Performance
- Dumping the Partition stack trace to a log file
- Enabling a Partition to run with Optimizeit Profiler or ServerTrace

Creating, cloning, and deleting Partitions

Using the Management Console, you can create, clone, and delete Partitions for your applications. A Partition can be created from a template or cloned from an existing Partition. Partitions are represented in the Navigation Pane of the Management Console's Hubs View as child nodes of their parent configurations.

Creating a new Partition

To create a new Partition:

1. Select the Configuration to which you want the new Partition to belong in the Navigation Pane.
2. Right-click the Configuration and select Add Managed Object.

   The Managed Object Template Gallery opens.
3. From the AppServer or OpenJMS Partition categories, choose from the following Partition templates:
   - **AppServer 6.7 Partition:** Produces a managed Partition using the default partition.config for AppServer 6.7.
   - **Standard Partition:** Produces a managed Partition.

- **Explicitly Pathed Partition:** Produces a path to an existing Partition. Use this template to migrate existing Partitions that you want under the management of the current configuration.
- **JBuilder Partition:** Produces an unmanaged Partition. **The JBuilder Partition is for use with JBuilder debugging of local servers. It is automatically created by JBuilder when used with AppServer for debugging purposes. You should not use the JBuilder Partition otherwise.**
- **Partition with Embedded OpenJMS:** Produces a Partition with embedded OpenJMS.

1 Select a Partition template and click Add.

The Add From Template dialog box appears.

Note

The information that appears on the Add From Template dialog box depends upon which template you chose in the previous step.

2 Enter the required information (shown in bold) in the dialog.

Different properties appear in this dialog, depending on which template you selected. Some of the common properties are:

- **Name:** Give the Partition a unique name in the Name field. This name will be used as the value of the string-replacement variable `${mo.name}` throughout the rest of the dialogs, and will be used as the display name of the Partition. If you want a display name different from the actual Partition name, change the value of the Display Name field.
- **Management Agent:** Select the host on which to create the Partition by selecting a valid Management Agent from the drop-down list. If you want to use the current Management Agent, represented by the string replacement variable `${hub.name}`, you do not need to change this field.
- **Display Name:** You can optionally enter a friendly name to be displayed in the Management Console. The string replacement variable, `${mo.name}`, indicates that the value in the Name field will be displayed.
- **Smart Agent Port:** You can modify the default osagent (Smart Agent) port number by entering a valid port number in this field.
- **HTTP Connector Port:** You can modify the default HTTP connector port number in this field.
- **Data Directory:** For an explicitly pathed Partition you must enter the path to an existing Partition.

1 Check the Show hidden properties check box to display some additional Partition configuration properties.

2 When you are finished, click OK.

Cloning an existing Partition

To clone an existing Partition:

1. Select the Partition you want to clone from the Navigation Pane.
2. Right-click the Partition and select Clone.

   The Clone Managed Object dialog appears.
3. Select the Configuration for which you want to target the new, cloned Partition from the Target Configuration drop-down list.
4. Enter the name you want to provide for the cloned Partition in the New Name field.

   This name will be used as the value of the string-replacement variable `${mo.name}` throughout the rest of the dialogs, and will be used as the display name of the cloned Partition. If you want a display name different from the actual Partition name, change the value of the New Display Name field.
5. Enter a Description, if desired. This is optional.
6. The Data Directory, `${config.path}/mos/${mo.name}`, identifies the location of the Partition relative to its Agent.
7. Choose a management agent to host the cloned Partition from the Target Agent drop-down list.
8. Choose the group to which the cloned Partition should belong from the Target Group drop-down list.
9. When you are finished, click OK.

Deleting a Partition

To delete a Partition:

1. Select the Partition you want to delete from the Navigation Pane.
2. Right-click the Partition and select Remove.

Deploying modules and libraries to a Partition

To deploy modules to a Partition:

1. Select the Partition to which you want to deploy in the Navigation Pane.
2. Right-click the Partition and select Deploy modules.

   The Module and Library Deployment Wizard appears.
3. To add a module, click Add.

   The Add J2EE Module dialog appears.

   a. Navigate to the module you wish to deploy to the Partition, and click OK.

   b. Repeat this step until all the application modules to be deployed have been added.

      If you make an error, you can highlight the module in the list and click Remove to delete it.

4 Select any additional options using the check boxes. The options are:
   - **Restart partitions on deploy (cold deploy):** Executes a “cold” deploy, and will restart the Partition after the deployment operation is complete. Leave this unchecked if you wish to “hot” deploy to a running Partition without restarting.
   - **Verify deployment descriptors:** Runs the verification tool which checks to make sure that all deployment descriptors have been constructed properly, including Borland-specific descriptors. This option is recommended.
   - **Generate stubs:** Check this box if you want your application stubs generated at deploy-time. This option is recommended.

1 Click Advanced Options to configure advanced properties for this deployment. See “Advanced options for deployment” for more information.

2 Click Next to continue.

   Step 2 of the wizard appears.

3 Select the Partitions that will be the deployment targets for your modules.

   The available Partitions appear automatically in the list. If no Partitions appear initially, click Refresh List. *Shift-click* or *Ctrl-click* to select multiple Partitions.

4 When you are done, click Finish.

5 While the module is deployed you can observe the progress in the Deploying Modules dialog.

6 When you are done, click Close.

Advanced options for deployment

In Step 1 of the Deployment Wizard, you can choose to set Advanced Options for the Stub Generator and Verifier tools. To work with advanced settings:

1 In Step 1 of the Deployment Wizard, click Advanced Options.

   The Advanced Deployment Options dialog appears.

2 On the Stub Generator tab, you can set the following options:
   - **Generate stubs:** Check this box to enable stub generation on deployment.
   - **Classpath:** Select a Classpath from the drop-down list or click Edit to add archives to the list for addition to the Classpath.
   - **Edit:** Click Edit to open the Classpath Editor, where you can add and remove archives and paths using the editor.
   - **Java2IIOP arguments:** Include a sequence of `java2iiop` command-line arguments in this field. Click More Info for a list of valid command-line arguments, or see “Programmer tools for Java” in the *VisiBorker for Java Developer’s Gude.*
   - **More Info:** Click More Info to view the Stub Generator compiler flag usage information.
   - **Javac arguments:** Include a sequence of `javac` command-line arguments in this field.

1 On the Verifier tab, you can choose the level of verification using the following check boxes:
   - **Verify deployment descriptors:** Check this box to verify all descriptors, both Borland and standard.
   - **Show all warnings:** Check this box to receive logged information of all potential problems with your descriptors.
   - **Use strict (pedantic) checks:** Perform pedantic checks on all descriptors.

1 When you are finished, click OK.

Hosting additional modules

You can also host “pre-deployed” modules that will not reside on the Partition footprint. Additionally, you have the option of hosting a path (that is, an application that is not converted to an archive form) called an “exploded archive.”

To have a Partition host an archive:

1. Select the Partition to which you want to deploy in the Navigation Pane.
2. Right-click the Partition and select Host additional module.

   The Host Additional Module dialog appears.
3. Choose the Select File option to host an archive, or choose the Select Directory option to host an exploded archive.
4. Click Browse to locate the archive or exploded archive you want to host.
5. Optionally, you can provide a descriptive Module name for the hosted module.
6. When you are finished, click OK.

Configuring Partitions

You can set Partition properties, including all those that can be viewed in the Content Pane when a Partition is selected in the Navigation Pane. You can specify general information about the Partition, Partition properties, statistics gathering settings, JMX Agent settings, log settings, JMX client settings, time rules, and advanced options for management.

To edit a Partition's property settings:

1. Select the Partition in the Navigation Pane and right-click it.
2. Select Properties.

   The Partition Properties dialog appears.
3. Use the tabs described in the following sections to edit the settings.
4. When you are finished making changes, click OK.

General properties

The properties on the General tab allow you to specify some general information about the Partition.

You can edit the following options:

- **Display name:** Enter the name of this Partition as displayed in the Management Console.
- **Data directory:** Enter the location of the Partition's footprint, relative to its Agent.
- **Version:** A version number created and tracked by the Management System.
- **Vendor:** The managed object vendor. The standard partition vendor is Borland Software Corporation.
- **Description:** An optional description for your Partition.

Note

According to the specification for application module (EAR), the classpath entries of the submodule that is specified in the manifest file should be added to the application classpath. This feature is not used in most cases. If you want to use it, please specify the VM property `enable.add.classpath.entries` into the corresponding configuration file, for example, partition.config or iastool.config.

Partition Settings properties

The properties on the Partition Settings tab allow you to specify command-line arguments for the Partition and set up JPDA debugging.

You can set the following options:

- **Arguments:** You can either enter a space-separated list of command-line arguments for the Partition executable, or click the Edit button to store them in a list.
- **Enable JPDA remote debugging:** Flags whether or not debugging is enabled on the Partition.
- **JPDA debugging transport address:** The port used by the JPDA debugger to connect to the Partition. Leave this field blank for random port assignment.
- **Suspend partition until debugger attaches:** This is a flag that, when checked, does not mark the Partition as Running until the debugger successfully attaches.

Statistics properties

The properties on the Statistics tab allow you to gather statistical information which is displayed in the statistics tabs throughout the Management Console. Statistics gathering is enabled by default. Disable statistics gathering to improve the performance of the Partition.

When enabled you can configure the following statistics gathering settings:

- **Enable Agent Statistics:** Check this box to enable the Partition's statistics agent.

  You can set the logging level using the Statistics level drop-down list. Additionally you can set the polling interval by entering a value in the Snapshot period field.

- **Enable Agent Statistics Reaping:** Check this box to periodically delete stored statistics information and preserve disk space, known as *reaping*.

  You determine how long to keep statistics by entering a value in Reap older than. You can set how often the statistics logs are reaped by entering a value in Reap period.

JMX Agent properties

The properties on the JMX Agent tab allow you to configure some aspects of the JMX MBean Server, the RMI-IIOP and HTTP adaptors, and the MLet service implemented for the Partition.

You can edit the following options:

- **Enable JMX:** Check this box to enable the JMX MBean Server.

  The MBean Server is an interface and a factory object defined by the agent specification level of the JMX. This option must be enabled to launch the JMX Console (see “Using the JMX console”).

- **Enable HTTP Adaptor:** Check this box to enable the HTTP adaptor.

  The HTTP adaptor is the adaptor for the HTTP protocol through which the Partition can be managed through any HTML 3.2 compliant browser or application.

  You can configure the port number at which the HTTP adaptor is listening (default 8082) and enable the XSLT processor in this configuration tab. If you are using a Web browser to monitor the Partition, the XSLT processor must be enabled to transform the HTTP adaptor output from raw XML to HTML. To configure the host name (default `localhost`) you must edit partition.xml (see “Partition XML reference” in *the AppServer Developer's Guide* for more details).

- **Enable RMI-IIOP adaptor:** Check this box to enable the RMI-IIOP adaptor.

The RMI-IIOP adaptor is based on the JMX client framework, which helps managing applications to communicate with the MBean Server through RMI.

– **Enable mlet service:** Check this box to enable the MLet service.

The MLet service is useful for loading MBean classes and resources inside an MBean Server's JVM from a remote host and registering the MBeans in a single action.

For more information about configuring the JMX agent properties see “Partition XML reference” in *AppServer Developer's Guide*. For information about using the JMX console, a JMX client available to monitor MBeans associated with the Partition, see “Using the JMX console.”

JDK properties

The properties on the JDK tab allow you to set JDK properties on the Partition.

You can edit the following options:

– **Select the JDK to be used by this partition:** Select the appropriate JDK from the list by clicking on it.

– **Heap and Thread Stack Sizes:** Enter a value in the Initial heap size field to modify the starting heap size.

You can control the maximum heap size by entering a value in the Maximum heap size field. Use the Java thread stack size field to control how threads are managed within the VM.

– **Java VM Type:** Select the Java VM type by selecting the appropriate radio button. The values are Server, Client, or Other (which you can customize from the drop-down list).

Advanced JDK options

Click Advanced Configuration to open the `partition_server.config` file in an editing window. Add entries to the file as needed. When you are finished, click OK.

Performance Tuning Hints

Click Performance Tuning Hints to get hints on optimizing the performance of your Partition. When you are finished, click OK. For more information about performance tuning, see “Tuning the Partition for performance”.

VisiBroker properties

The properties on the VisiBroker tab allow you to tweak some of the VisiBroker ORB properties used by the Partition.

You can edit the following options:

– **Select a server connection manager:** Select a server connection manager from the drop-down list.

– **Listener port:** Set the server connection manager listener port.

– **Connection Pool:** Set the maximum number of allowable connections and the maximum connection idle time in these fields.

– **Thread Dispatcher Pool:** Set the range of allowable threads and the maximum thread idle time in these fields.

Advanced VisiBroker options

Click Advanced to open the `vbroker.properties` file in an editing window. Edit the file as needed. When you are finished, click OK.

Security properties

The properties on the Security tab allow you to set security information for your Partition.

To enable security, select a Security profile from the drop-down list. To enable SSL on this Partition select the ssl_enabled option. Additionally, if you selected `ssl_enabled` you can configure the following settings:

- **SSL Listener Settings:** Set the SSL listener port and enable trust in a client connected through the port.
- **SSL Connection Pool:** Set the maximum number of allowable connections and the maximum connection idle time in these fields.
- **SSL Thread Dispatcher Pool:** Set the range of allowable threads and the maximum thread idle time in these fields.

Log Settings properties

The properties on the Log Settings tab allow you to set logging properties on the Partition.

You can edit the following options:

- **Partition log format:** Choose a log format (XML or text) from the drop-down list.
- **Trace Level Settings:** Select the trace level (minimal or verbose) for each of the services where trace is provided.

Time Rules properties

The Time Rules tab allows you to configure rules for when the Partition should be running and when it should be stopped.

You can edit the following properties:

- **Default state:** Indicate whether the default state of the Partition is Running or Stopped with this drop-down list.
- **Rules:** Configure the Managed Object Availability Timer rules for the Partition here.

Advanced properties

The properties on the Advanced tab allow you to configure advanced information about how the Partition is managed.

You can set the following two types of management information: Managed Object Settings, and Management Action Settings.

Managed Object Settings

You can edit the following Managed Object Settings:

- **Local restart:** When checked, instructs the local Agent to carry out restarts on the Partition rather than having a remote Hub send the request.
- **Escalate stop:** When checked, the stop operation on the Partition will be escalated to a kill if the stop operation times out.
- **Ping policy:** When set to Always, the Partition is always pinged for its status whether it is running or not. When set to Not-when-stopped, the Partition is only pinged when it is known to be in a Running state.
- **Ping strategy:** The AppServer Action Strategy to use for this operation.
- **Ping interval:** Amount of time in seconds between state checks on the Partition.
- **Max failure retries:** Maximum number of times the Managed Object will attempt to retry an operation, such as Start or Stop.
- **Failure retry interval:** Amount of time between Managed Object attempts to retry an operation.

Management Action Settings

This section contains three tabs for configuring starting, stopping, and killing the Partition. For each operation (Start, Stop, and Kill) you can edit the following:

- **Strategy:** the AppServer Action Strategy to use for this operation.
- **First ping delay:** Amount of time before any pinging begins. Leave this field blank for no delay before the first Ping operation.
- **Ping interval:** Amount of time between state checks on the Partition to determine the success of the operation.
- **Retry interval:** (Stop and Kill tabs only) Amount of time between retry attempts.
- **Max retries:** Maximum number of times the Managed Object will attempt to retry this operation if it fails the first time.
- **Timeout:** Amount of time to allow the operation to proceed without success.

Viewing Partition information

When you select a Partition in the Navigation Pane, a variety of information is displayed in the Content Pane on the right side of the Management Console. This section details the tabs that appear in the Content Pane and what information they display.

General tab

The General tab displays basic information about the Partition. It shows three different categories of information: General Properties, Partition Properties, Security Settings, and the Web Container Root Context.

General properties

- **Display Name:** The name of the Partition as displayed in the Management Console.
- **Name:** The logical name of the Partition.
- **Agent name:** The hosting Agent for the Partition.
- **Data directory:** The location of the Partition's footprint, relative to its Agent.
- **Version:** An optional version number for the Partition, provided by the user.
- **Vendor:** The Partition's vendor, provided by the user.
- **Description:** An optional description for the Partition, provided by the user.

Partition properties

- **Path of partition on server:** The physical location of the Partition's footprint.
- **Verify all modules when loaded:** Flags whether or not the verification tool runs at Partition startup.

Security properties

- **Security profile:** Flags whether or not basic security (authentication/authorization) is enabled on the Partition.
- **Secure transport enabled:** Flags whether or not RPCs implement SSL.

Web Container Root Context

This section of the General properties tab displays the Web Container's root context.

Properties tab

The Properties tab displays information about JPDA remote debugging and the Server Connection Manager settings.

Virtual Machine

- **Enable JPDA remote debugging:** Flags whether or not debugging is enabled on the Partition.
- **JPDA debugging transport address:** The port used by the JPDA debugger to connect to the Partition.
- **Suspend partition until debugger attaches:** Flag that, when true, does not mark the Partition as “Running” until the debugger successfully attaches.

Server Connection Manager Settings

- **Server connection manager name:** The server connection manager servicing the Partition.
- **Listener port:** The port on which the server connection manager listens for incoming connections.
- **Connection Pool:** Shows the range of minimum and maximum allowable connection to the Partition.
- **Dispatcher Pool:** Shows the range of minimum and maximum allowable threads within the Partition. Additionally displays the Maximum Thread Idle time.

XML tab

The XML tab displays the XML data block defining the Partition's management properties. The data block is excerpted from the `configuration.xml` file.

Class Loading tab

The Class Loading tab displays the Partition's classloading policy and the classloader hierarchy.

Logs tab

The Logs tab displays the log4j, stderr, and stdout logs. Use the drop-down list in the upper left corner of the tab to view the contents of each log. The log4j log has filtering capabilities.

Status tab

If the Partition's statistics agent is enabled, you can view “real-time” performance information about your Partition on the Status tab. To enable statistics gathering see “Statistics properties”.

JDBC Pool States tab

If the Partition's statistics agent is enabled, you can view “real-time” performance information about the JDBC pool on the JDBC Pool States tab. To enable statistics gathering see “Statistics properties”.

Tuning the Partition for performance

The Partition has a Performance Tuning Wizard that allows you to set particular preferences about a Partition's operation. You can also configure some advanced properties for performance tuning.

To tune a Partition:

1. Select the Partition you want to tune from the Navigation Pane.
2. Right-click the Partition and select Performance Tuning. The Performance Tuning Wizard Step 1 appears.

   In this step you can select the usage model for statistics tuning. The choices here are reflected in the statistics gathering settings in the next Wizard step. Choose from among the following models:

   - **Developer:** Enables a maximum level of statistics gathering.
   - **System test:** Enables a medium level of statistics gathering.
   - **Production:** Enables a minimum level of statistics gathering.
   - **Best performance:** Selecting this option disables statistics gathering, improving the performance of your Partition.
   - **Custom:** Selecting this option means that your desired settings do not match any of the predefined usage models listed above. You can customize any of the other usage models in which case your usage model will be described as *custom*.

1. Click Next when you are finished. The Performance Tuning Wizard Step 2 appears.

   In this step you can determine how the Partition collects statistics about its operation. Statistics information is written to disk and displayed in the Partition's Statistics tab in the Content Pane. For more information on Partition statistics, see “Partition XML reference” in the *AppServer Developer's Guide*. For maximum performance, disable statistics gathering. In this step you can:

- **Enable Agent Statistics:** Check this box to enable the Partition's statistics agent. You can set the logging level using the Statistics level drop-down list. Additionally you can set the polling interval by entering a value in the Snapshot period field.
- **Enable Agent Statistics Reaping:** Check this box to periodically delete stored statistics information and preserve disk space, known as *reaping*. You determine how long to keep statistics by entering a value in Reap older than. You can set how often the statistics logs are reaped by entering a value in Reap period.

1 To continue, click Next. Step 3 appears.

Having unneeded or unused deployed modules enabled in a Partition adds to Partition start up time, increases it's memory footprint, and adds to the time taken by the garbage collector. Choose the modules you want to disable by selecting them in the list. You can *Shift-click* or *Ctrl-click* to select multiple modules.

2 To continue, click Next. Step 4 appears.

You can disable those Partition services not needed by your applications. To disable a Partition service, uncheck its check box. To enable it, check the check box.

3 To continue, click Next. Step 5 appears.

You can improve performance by tuning the Java VM parameters appropriately for your application. In this step, you can configure the following settings:

- **Select the JDK to be used by this partition:** Select the JDK from the list by clicking on it.
- **Heap and Thread Stack Sizes:** Enter a value in the Initial heap size field to modify the starting heap size. You can control the maximum heap size by entering a value in the Maximum heap size field. Use the Java thread stack size field to control how threads are managed within the VM.
- **Java VM Type:** Select the appropriate option. Valid values are either Server, Client, or Other (which you can select from the drop-down list).

In addition you can do some advanced configuration (see “Advanced performance tuning options”) or look at the performance tuning hints.

1 When you are finished, click Next. Step 6 appears.

You can set some of the VisiBroker parameters which affect performance. In this step, you can configure the following settings:

- **Connection Pool:** Set the maximum number of allowable connections and the maximum connection idle time in the appropriate dialog boxes.
- **Dispatcher Pool:** Set the range of allowable threads in the Minimum number of threads and Maximum number of threads fields. Set the Maximum thread idle time in its field.
- **Security:** Flags whether or not basic security (authentication/authorization) is enabled on the Partition.

1 To continue, click Next. The last step appears.

In this step you can tune some J2EE specific properties. You can configure the following settings:

- **EJB Container Tuning:** Set the Session passivation timeout field. Check the check box to use PBV for intra-bean calls.
- **Message-driven bean thread pool:** Set the thread pool range by setting values in the Minimum number of threads and Maximum number of threads fields. Set the Maximum thread idle time in its field.
- **Web Container Tuning:** Select the HTTP Service Connector from the drop down list, and set the range of processors and connection timeout settings.

1 When you are done, click Finish.

Advanced performance tuning options

To access the advanced options for Performance Tuning:

1. In Step 4 of the Performance Tuning Wizard, click Advanced Configuration.
2. The `partition_server.config` file opens in an editing window.
3. Edit the file as needed. When you are finished, click OK.

Dumping the Partition stack trace to a log file.

You can dump the Partition's stack trace to a log file for troubleshooting.

To dump the Partition's stack trace to a log file:

1. Select the Partition in the Navigation Pane.
2. Right-click the Partition and select Dump stack to log.
3. The stack trace is dumped to the stdout.log, located in:

```
<install_dir>/var/domains/<domain-name>/configurations/<configuration-name>/
    mos/<partition-name>/adm/logs/<partition_name>.stdout.log
```

Enabling a Partition to run with Optimizeit Profiler or ServerTrace

See "Using Optimizeit Profiler and ServerTrace" for information about configuring and using the Borland Management Console integration with Optimizeit ServerTrace and Optimizeit Profiler.

Index

E

Index

F

G

H

I

J

L

M

N

O

P

Index

R

S

T

U

V

Index

W

X