



**BlueDragon,  
BEA WebLogic® Edition 6.2.1  
User Guide**

---

# BlueDragon, BEA WebLogic® Edition

## 6.2.1

### User Guide

---

Published April, 2006

Copyright © 1995-2006 BEA Systems, Inc. All Rights Reserved.

This software is protected by copyright, and may be protected by patent laws. No copying or other use of this software is permitted unless you have entered into a license agreement with BEA authorizing such use. This document is protected by copyright and may not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form, in whole or in part, without prior consent, in writing, from BEA Systems, Inc.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA SYSTEMS DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE DOCUMENT IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Copyright © 1995-2006 BEA Systems, Inc. All Rights Reserved. BEA, BEA JRockit, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop, Built on BEA, Jolt, JoltBeans, SteelThread, Top End, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA AquaLogic, BEA AquaLogic Data Services Platform, BEA AquaLogic Enterprise Security, BEA AquaLogic Service Bus, BEA AquaLogic Service Registry, BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA MessageQ, BEA WebLogic Commerce Server, BEA WebLogic Communications Platform, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Java Adapter for Mainframe, BEA WebLogic JDriver, BEA WebLogic Log Central, BEA WebLogic Network Gatekeeper, BEA WebLogic Personalization Server, BEA WebLogic Personal Messaging API, BEA WebLogic Platform, BEA WebLogic Portlets for Groupware Integration, BEA WebLogic Server Process Edition, BEA WebLogic SIP Server, BEA WebLogic WorkGroup Edition, Dev2Dev, Liquid Computing, and Think Liquid are trademarks of BEA Systems, Inc. BEA Mission Critical Support, BEA Mission Critical Support Continuum, and BEA SOA Self Assessment are service marks of BEA Systems, Inc.

All other names and marks are property of their respective owners.

Copyright © 1997-2006 New Atlanta Communications, LLC. All rights reserved.

100 Prospect Place • Alpharetta, Georgia 30005-5445

Phone 678.256.3011 • Fax 678.256.3012

<http://www.newatlanta.com>

BlueDragon is a trademark of New Atlanta Communications, LLC. ServletExec and JTurbo are registered trademarks of New Atlanta Communications, LLC in the United States. Java and Java-based marks are trademarks of Sun Microsystems, Inc. in the United States and other countries. ColdFusion is a registered trademark of Macromedia, Inc. in the United States and/or other countries, and its use in this document does not imply the sponsorship, affiliation, or endorsement of Macromedia, Inc. All other trademarks and registered trademarks herein are the property of their respective owners.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org>).

No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise without the prior written consent of New Atlanta Communications, LLC.

New Atlanta Communications, LLC makes no representations or warranties with respect to the contents of this document and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, New Atlanta Communications, LLC reserves the right to revise this document and to make changes from time to time in its content without being obligated to notify any person of such revisions or changes.

The Software described in this document is furnished under a Software License Agreement (“SLA”). The Software may be used or copied only in accordance with the terms of the SLA. It is against the law to copy the Software on tape, disk, or any other medium for any purpose other than that described in the SLA.

# Contents

<b>1. INTRODUCTION .....</b>	<b>1</b>
1.1 ABOUT THIS MANUAL .....	1
1.2 ABOUT THIS MANUAL .....	1
1.3 BLUEDRAGON PRODUCT CONFIGURATIONS.....	1
1.4 TECHNICAL SUPPORT .....	2
1.5 OTHER DOCUMENTATION.....	2
<b>2. GETTING STARTED.....</b>	<b>3</b>
2.1 INSTALLATION AND VERIFYING INSTALLATION .....	3
2.2 WEB SERVER ADAPTERS AND THE BUILT-IN WEB SERVER .....	3
2.3 USING AND CONFIGURING DATASOURCES .....	3
2.4 EXECUTING YOUR CFML APPLICATIONS .....	4
2.4.1 <i>When Working with a Web Server Adapter</i> .....	4
2.4.2 <i>When Working with the Built-in Web Server</i> .....	4
2.5 BLUEDRAGON WORK FILES .....	4
2.5.1 <i>BlueDragon.xml Configuration File</i> .....	4
2.5.2 <i>BlueDragon Log Files</i> .....	5
2.5.3 <i>Work Directory Locations</i> .....	6
2.5.4 <i>Viewing Log Files</i> .....	6
<b>3. ADMINISTRATION CONSOLE.....</b>	<b>7</b>
3.1 PRODUCT-SPECIFIC ADMIN CONSOLE INFORMATION .....	7
3.1.1 <i>BlueDragon Server and Server JX</i> .....	7
3.2 BLUEDRAGON FOR J2EE SERVERS .....	8
3.2.1 <i>BlueDragon for the Microsoft .NET Framework</i> .....	9
3.3 ADMINISTRATION CONSOLE ONLINE HELP .....	9
<b>4. DATASOURCES .....</b>	<b>11</b>
4.1 OVERVIEW .....	11
4.1.1 <i>About Database Drivers</i> .....	11
4.1.2 <i>Automatic Configuration of ODBC Datasources (Windows)</i> .....	12
4.1.3 <i>Configuring Access, SQL Server, and Other ODBC Datasources</i> .....	13
4.1.4 <i>Support for MySQL</i> .....	13
4.1.5 <i>Configuring Datasources Not Listed</i> .....	13
4.1.6 <i>Database Connection Pooling</i> .....	13
4.1.7 <i>J2EE Datasources</i> .....	14
4.2 CONFIGURING DATASOURCES.....	14
4.2.1 <i>Possible Timeout Errors When Verifying All Datasources</i> .....	15
4.2.2 <i>ODBC Datasource Auto-Configuration</i> .....	15
4.2.3 <i>Add Datasource Wizard – Step 1</i> .....	17
4.2.4 <i>Add Datasource Wizard – Step 2</i> .....	19
4.2.5 <i>Add Datasource Wizard – Step 3</i> .....	21
4.2.6 <i>Add Datasource Wizard – Step 4</i> .....	22
4.3 MODIFYING AND DELETING DATASOURCES .....	22
<b>5. UPGRADING FROM COLDFUSION .....</b>	<b>24</b>
5.1 ADMINISTRATION CONSOLE/CONFIGURATION DIFFERENCES .....	24
5.1.1 <i>Site-wide Error Handling Templates</i> .....	24
5.1.2 <i>Server Debugging Output</i> .....	24
5.1.3 <i>CFML Tag and Function Security</i> .....	25

5.1.4 Administration Console Mappings .....	25
5.1.5 WhiteSpace Compression .....	25
5.1.6 Administration Console Extended Features .....	25
5.1.7 CFX Custom Tags .....	26
5.1.8 File Cache/Template Cache .....	27
5.2 OTHER FEATURES RELATED TO COLD FUSION .....	28
5.2.1 CFENCODed CFML Templates .....	28
5.2.2 CFC and Web Service Browsing .....	28
5.2.3 Precompiled, Encrypted CFML Templates .....	29
5.2.4 Flash, Flash Remoting, and Other Flash Integration .....	31
5.2.5 Ajax Integration .....	32
5.2.6 RDS (For ColdFusion Studio, HomeSite+, and Dreamweaver) .....	33
5.2.7 Page Buffering Behavior .....	34
5.2.8 FuseBox, Mach II, and Model-Glue Support .....	35
<b>6. CFML/JAVA/J2EE INTEGRATION .....</b>	<b>36</b>
6.1 INTEGRATING CFML WITH JAVA AND J2EE .....	36
6.2 PLACEMENT OF JAVA CLASSES .....	37
6.3 SESSION AND APPLICATION SCOPE SHARING .....	37
6.4 LEARNING MORE ABOUT CFML/J2EE INTEGRATION .....	38
6.5 JAVA AND THE MICROSOFT .NET FRAMEWORK .....	39
<b>7. BLUEDRAGON SERVER FREE EDITION .....</b>	<b>40</b>



# 1. Introduction

**B**lueDragon is family of server-based products for the deployment of ColdFusion® Markup Language (CFML) applications for dynamic web publishing. CFML is a popular server-side, template-based markup language that boasts a rich feature set and renowned ease-of-use. BlueDragon provides a high-performance, reliable, standards-based environment for hosting CFML web applications, and enables the integration of CFML with the Microsoft .NET Framework and Java 2 Platform, Enterprise Edition (J2EE) technologies.

## 1.1 About This Manual

The *BlueDragon, BEA WebLogic Edition 6.2.1 User Guide* presents information of use to both CFML developers and BlueDragon administrators. It offers a brief discussion of the BlueDragon product configurations, their installation, and some configuration issues. Additionally, it offers information on using the BlueDragon administration console, with particular focus on configuring datasources. It also offers a discussion of certain issues relevant to those migrating from ColdFusion to BlueDragon, and finally it discusses some topics regarding integration of CFML with J2EE, available on both the Server JX and J2EE editions of BlueDragon.

## 1.2 About This Manual

The *BlueDragon, BEA WebLogic Edition 6.2.1 User Guide* presents information of use to both CFML developers and BlueDragon administrators. It offers information on using the BlueDragon administration console, with particular focus on configuring datasources. It also offers a discussion of certain issues relevant to those upgrading from ColdFusion to BlueDragon, and finally it discusses some topics regarding integration of CFML with J2EE, available on both the Server JX and J2EE editions of BlueDragon.

## 1.3 BlueDragon Product Configurations

BlueDragon is available in four product configurations. Details about these configurations—BlueDragon Server, BlueDragon Server JX, BlueDragon for J2EE Servers, and BlueDragon for the Microsoft .NET Framework—are provided in other related manuals, discussed in the “Additional Documentation” section below. Except where explicitly

noted, all references to “BlueDragon” in this document refer to all product configurations.

## 1.4 Technical Support

If you’re having difficulty installing or using BlueDragon, visit the self-help section of the New Atlanta web site for assistance:

[http://www.newatlanta.com/products/bluedragon/self\\_help/index.cfm](http://www.newatlanta.com/products/bluedragon/self_help/index.cfm)

Besides offering all the documentation, a FAQ, and feature request options, perhaps the most useful aspect of our self-help support is our active BlueDragon-Interest discussion list. New Atlanta engineers and customers are available to help solve both common and challenging problems. There’s also a searchable archive of past discussion list topics.

Details regarding paid support options, including online-, telephone-, and pager-based support, are available from the New Atlanta web site:

<http://www.newatlanta.com/biz/support/index.jsp>

For BEA Systems technical support, go to:

<http://support.bea.com>

## 1.5 Other Documentation

The other manuals available in the BlueDragon documentation library are:

- *BlueDragon 6.2.1 Server and Server JX Installation Guide*
- *BlueDragon, BEA WebLogic Edition 6.2.1 CFML Compatibility Guide*
- *BlueDragon, BEA WebLogic Edition 6.2.1 CFML Enhancements Guide*
- *BlueDragon, BEA WebLogic Edition Deploying CFML on J2EE Application Servers*
- *Deploying CFML on ASP.NET and the Microsoft .NET Framework*
- *Integrating CFML with ASP.NET and the Microsoft .NET Framework*

Each offers useful information that may be relevant to developers, installers, and administrators. BlueDragon, BEA WebLogic Edition documents are available at [edocs.bea.com](http://edocs.bea.com) and additional documents are available in PDF format from New Atlanta’s web site:

[http://www.newatlanta.com/products/bluedragon/self\\_help/docs/index.cfm](http://www.newatlanta.com/products/bluedragon/self_help/docs/index.cfm)

## 2. Getting Started

CFML developers interested in using BlueDragon, as well as administrators responsible for installing and configuring BlueDragon, should be aware that some aspects of working with BlueDragon may be discussed in more detail in other manuals in the BlueDragon documentation set. A few key points are reiterated here with reference to where to find additional detail.

### 2.1 Installation and Verifying Installation

Instructions for installing BlueDragon Server and verifying your installation can be found in the *BlueDragon 6.2.1 Installation Guide*. Instructions for installing BlueDragon/J2EE and verifying its installation can be found in the *Deploying CFML on J2EE Application Servers with BlueDragon* document.

On Windows, BlueDragon Server is installed as a service that is configured for automatic startup and is started by the installer. On Linux, BlueDragon Server is installed as a UNIX daemon and is started by the installer.

### 2.2 Web Server Adapters and the Built-in Web Server

The *BlueDragon 6.2.1 Installation Guide* also contains a thorough discussion regarding installing Web server adapters (for web servers including IIS, Apache, Netscape Enterprise Server, and iPlanet), as well the built-in web server that is installed with BlueDragon.

When using the built-in web server, the `wwwroot` subdirectory of the BlueDragon Server installation directory is the document root. Place any `.cfm`, `.jsp`, `.html`, `.gif` files, etc., in the `wwwroot` directory and serve them as you would normally for any web server.

### 2.3 Using and Configuring Datasources

The process of using and configuring datasources in BlueDragon is in some ways the same and in some ways different than you may be familiar with from past experience using ColdFusion. See Section 4 for a complete discussion of the topic.



## 2.4 Executing Your CFML Applications

With BlueDragon installed and verified, the next step will likely be to execute your existing CFML applications. How you do this will depend on whether you're using the built-in web server or have implemented a web server adapter.

### 2.4.1 When Working with a Web Server Adapter

If you're using a web server adapter, the location of your CFML files will be driven by your web server and/or virtual directory mappings for that web server. For instance, if you're using IIS, you would typically place your CFML templates in a subdirectory of C:\inetpub\wwwroot. Or, you may create a virtual directory in IIS and then point that virtual directory (alias) to any directory accessible to the web server.

Indeed, if you already have CFML templates in the web server root or a virtual directory from working with ColdFusion, and you implement the BlueDragon web server adapter for that web server (thus choosing to have BlueDragon process your CFML templates instead of ColdFusion), you will be able to simply execute those same CFML applications from their existing location. See the *BlueDragon 6.2.1 Installation Guide*, in the Section "Running Alongside ColdFusion and Servlet/JSP Engines," for information on configuring the environment when you already have ColdFusion installed and integrated with your web server.

### 2.4.2 When Working with the Built-in Web Server

If you choose to use BlueDragon's built-in web server instead (such as for testing or development), you will want to place your CFML templates in the the wwwroot subdirectory of the BlueDragon installation directory. Move or copy/paste any .cfm, .jsp, .html, .gif files, etc., into the wwwroot directory or a subdirectory and serve them as you would normally for any web server. (Note that the BlueDragon Server FREE edition does not support execution of JSP's.) The built-in web server does not currently support creation of virtual directories.

## 2.5 BlueDragon Work Files

While working with BlueDragon, it may be useful to understand some of the work files used in association with it. In particular, it can be helpful to view the logs of activity for the BlueDragon runtime environment.

### 2.5.1 BlueDragon.xml Configuration File

Whenever changes are made to the BlueDragon admin console (see section 3), the configuration information is written to a `bluedragon.xml` file. This file is located in the `config` subdirectory of the BlueDragon installation directory. See section 2.5.3 for more information on the location of such work files in each edition.

A new `bluedragon.xml` file is created each time a change is made to the admin console, and the previous copy is saved under a new file name as `bluedragon.xml.bak.n`. As of BlueDragon 6.2.1, the latest backup is `.bak.1` and the oldest would be `.bak.10`. No

more than 10 copies are kept (in prior releases, the oldest was `.bak.1` and the newest was named higher than the greatest number currently found, and there was no limit to how many were kept).

Note that if you change the configuration file manually, you must then restart the BlueDragon server (or web application in J2EE or .NET) to cause the changed values to be loaded into the BlueDragon runtime.

## 2.5.2 BlueDragon Log Files

There are multiple log files created during the execution of BlueDragon templates.

### 2.5.2.1 BlueDragon.Log file

Information about the execution of the BlueDragon engine can be found in `bluedragon.log`, which is located within the `work` subdirectory of the BlueDragon installation directory. See section 2.5.3 for more information on the location of the `work` directory in each edition.

As of BlueDragon 6.2.1, a new `bluedragon.log` file is created each time the BlueDragon engine is started, and up to 10 previous files are kept (prior released appended to the file on subsequent runs with no limit to its growth).

Note that there is a mechanism available to view the `bluedragon.log` file, in the BlueDragon administration console. See `General>view logfile`.

### 2.5.2.2 BlueDragon Runtime Error logs

Unlike ColdFusion, BlueDragon creates a log file for every runtime and compile-time error, in the `work\temp\rtelogs` subdirectory of the BlueDragon `work` directory. The location of the `work` directory will vary for the three product editions, Server/Server JX, J2EE, and .NET. See section 2.5.3 for more information.

The runtime error file created for each error contains the same HTML used to render the error page shown to the end user, including (usually) the dump of all variable scopes, the file trace, the tag trace, and the date/time of the error.

The advantage to creating a file for each error is that if an error occurs in production, you can see the error page without needing to ask the end user to print the page or create error handling processes with `CFERROR`. (You can, of course, still use `CFERROR` to create your own error handling process.)

Note that this log of error pages grows without limit. There is currently no rotation or limiting mechanism in place. You may want to delete older files occasionally. Note that if an error page includes a dump of a large number of variables, the file could be large (100-200 KB).

You can, however, completely disable the creation of the error log files in the BlueDragon administration console. See the `Error Logging` option in the console's `Application>Settings` section.

### 2.5.2.3 BlueDragon Server.log files

In the Server and Server JX editions only, the `BlueDragon Server.log` files are created and are located within the `logs` subdirectory of the BlueDragon installation directory (the file name is called `BlueDragon Server JX.log` on BlueDragon Server JX, and on Linux/OS X the words in the log file names are separated with underscores). These log files track both startup and execution information for the BlueDragon Server, as well as requests made against the BlueDragon built-in web server.

Each time the server is restarted, a new log file is created and the log from the previous run is named `BlueDragon Server.log.1` (and all previous logs are renamed to preserve chronological sequence). Additionally, if the log file ever reaches 100KB in size, it will be rotated/renamed in the same way. Only 9 of these backup log files are kept, by default, as controlled in `logging.properties` in the `config` subdirectory of the BlueDragon installation.

### 2.5.3 Work Directory Locations

Some of the logs discussed above are created in the BlueDragon `work` directory (or in BlueDragon Server and Server JX there is a `logs` directory under `work`, as well). The `config` directory (where `bluedragon.xml` is kept) is a sibling to the `work` directory.

In Server and Server JX, the `work` directory is located in the BlueDragon installation directory.

In BlueDragon for J2EE Servers, the `work` directory is located within the `WEB-INF/bluedragon/` subdirectory of your J2EE web application.

In BlueDragon for the Microsoft .NET Framework, there may be multiple work directories depending on the installation options chosen, which may cause there to be multiple admin consoles, with work files for each. See the documentation, *Deploying CFML on ASP.NET and the Microsoft .NET Framework*, for more information.

### 2.5.4 Viewing Log Files

Note that there is now a mechanism available to view the `BlueDragon.log` file, in the BlueDragon administration console. See `General>view logfile`.

Additional information on logs is presented in the *BlueDragon 6.2.1 Installation Guide*, Section “Viewing BlueDragon Logs”.

## 3. Administration Console

BlueDragon offers an administration console to support configuration and setup of the runtime environment for CFML pages.

### 3.1 Product-Specific Admin Console Information

The following sections discuss issues specific to each of the three editions of BlueDragon (Server/Server JX, J2EE, and .NET), including the URL to use to access it for each product.

#### 3.1.1 BlueDragon Server and Server JX

For BlueDragon Server and Server JX, the administration console is accessible only via the built-in web server; it is not accessible via web servers on which you've installed a web server adapter.

By default upon installation, the BlueDragon built-in web server will only accept requests from browsers running on the same computer (local IP address only). This can be changed during installation or in the BlueDragon administration console. Further discussion of setting and/or changing this default is found in the *BlueDragon 6.2.1 Installation Guide*, in the section “Changing the Default Localhost Limitation for the Built-in Web Server”.

To access the BlueDragon Server (or Server JX) administration console, enter the following URL in a web browser (if you specified a port other than 8080 for the built-in web server during installation, use that port number in the URL):

<http://localhost:8080/bluedragon/admin.cfm>

After entering this URL you will be prompted to log in to the BlueDragon administration console using the password you provided to the installer. See Figure 1, below. *Note that you must have cookie support enabled in your browser to log in to the BlueDragon administration console.* If you don't have cookies enabled, you won't be able to log in.

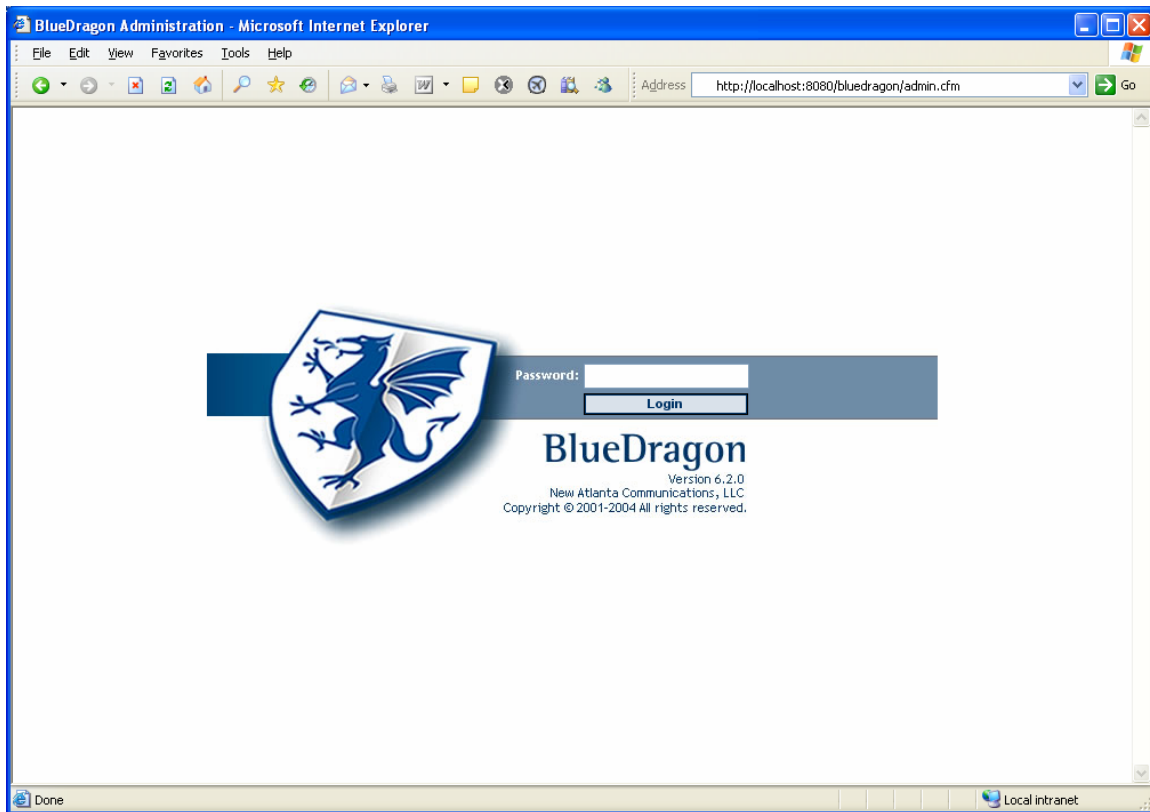


Figure 1. Administration Console Login

### 3.2 BlueDragon for J2EE Servers

For BlueDragon/J2EE, there is separate administration console for each J2EE web application that contains BlueDragon. In development mode (the default mode of BlueDragon/J2EE before you purchase a license key), BlueDragon/J2EE will only accept requests from browsers running on the same computer (local IP address only).

After deploying your webapp on the J2EE server, access its BlueDragon administration console via the following URL:

[http://localhost/<context\\_path>/bluedragon/admin.cfm](http://localhost/<context_path>/bluedragon/admin.cfm)

In the above URL, *<context\_path>* is the URL context path that you specified for the webapp when you deployed it on the J2EE server. After entering this URL you will be prompted to log in to the BlueDragon administration console. See Figure 1, above. By default there is no password assigned to the administration console; you can log in by simply clicking the “Password Login” button. After logging in, you can set the administration console password from the License and Security page.

*Note that you must have cookie support enabled in your browser to log in to the BlueDragon administration console. If you don't have cookies enabled, you won't be able to log in.*

After logging in, the BlueDragon administration console will appear similar to Figure 2, below. (Figure 2 shows the administration console home page for BlueDragon Server JX; the home pages for BlueDragon Server and BlueDragon/J2EE will appear slightly different). Click the menu items in the left frame to access the various BlueDragon administration functions.

### 3.2.1 BlueDragon for the Microsoft .NET Framework

BlueDragon for the Microsoft .NET Framework offers multiple installation options, and it's possible to have multiple BlueDragon admin consoles. See the documentation, *Deploying CFML on ASP.NET and the Microsoft .NET Framework*, for more information.

## 3.3 Administration Console Online Help

Each page of the administration console contains a help icon that links to detailed help for that feature. For example, Figure 3, below, shows the help window for the *License and Security* page. Refer to the help pages for detailed instructions on using the BlueDragon administration console.

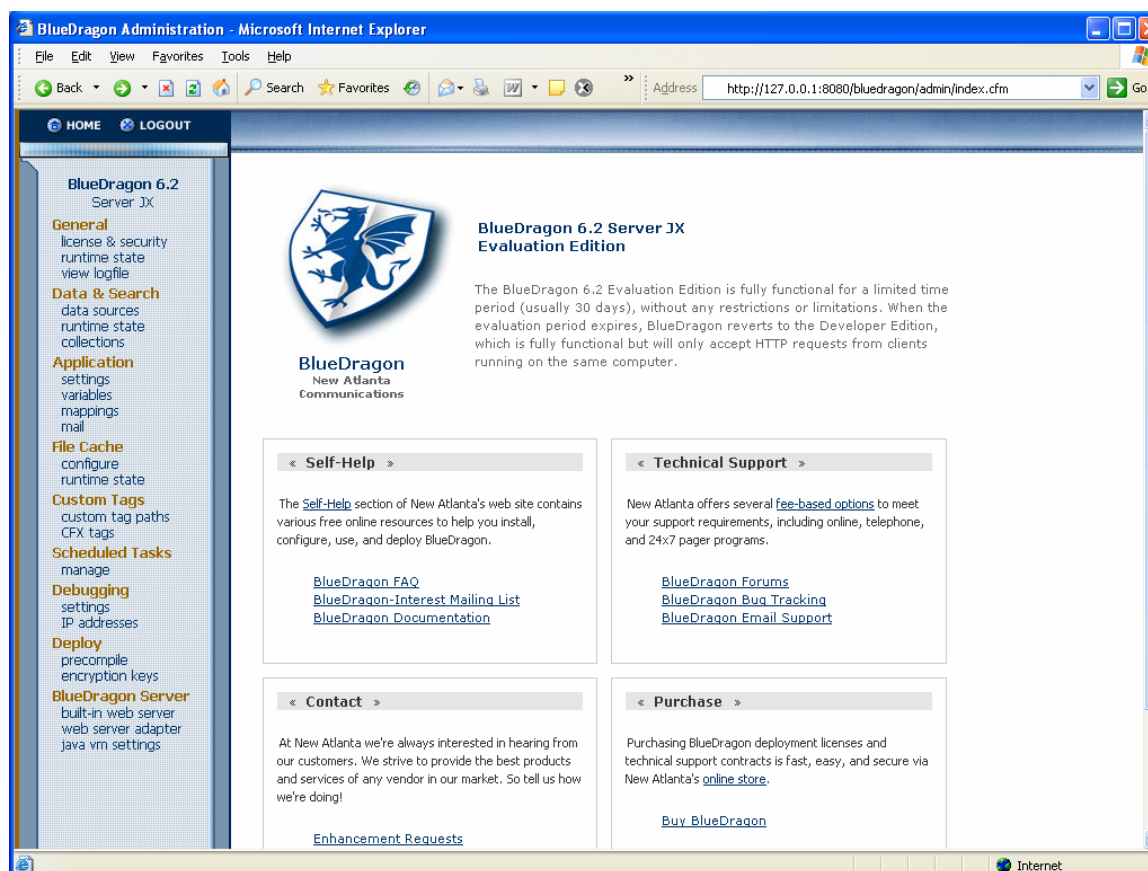


Figure 2. BlueDragon Administration Console

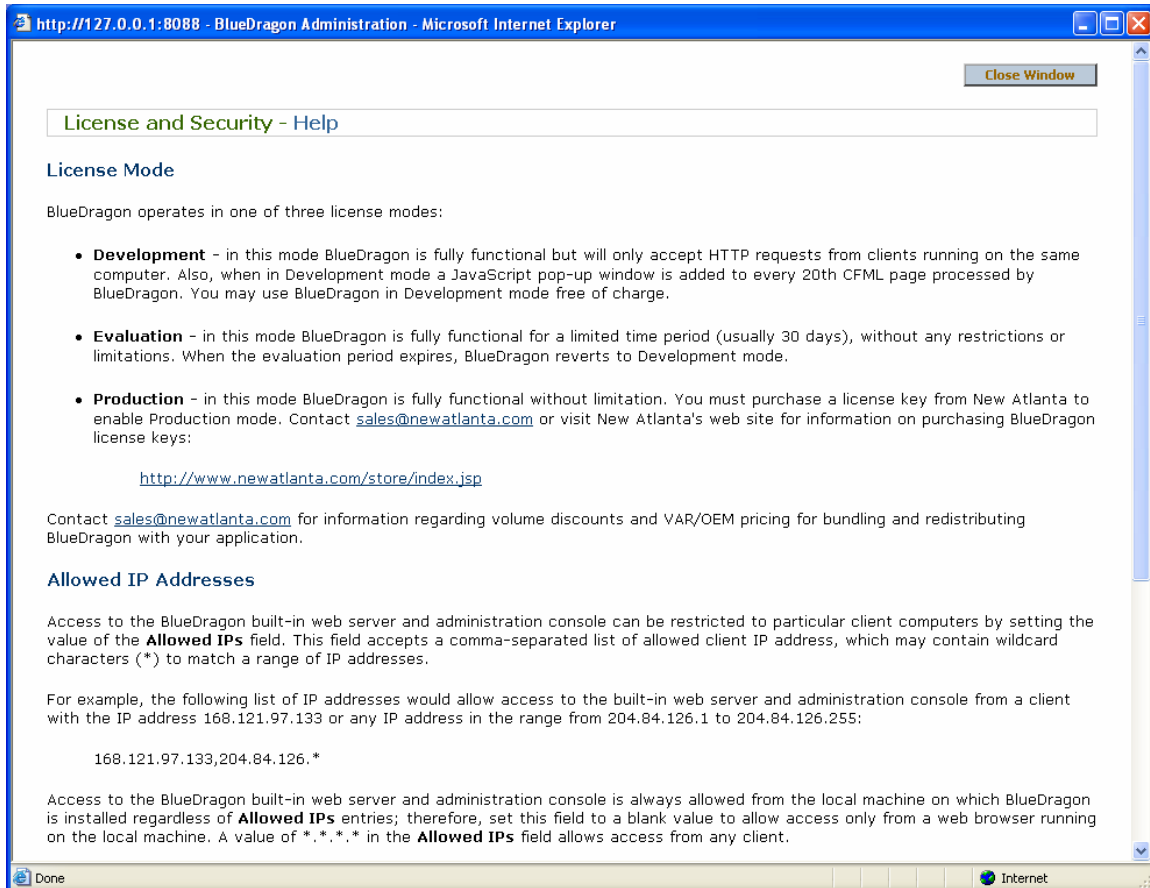


Figure 3. License and Security Help

# 4

## 4. Datasources

### *Database Access via CFML*

**I**n order to use CFML tags that interact with databases, it's generally necessary to configure named datasources via the BlueDragon administration console.

There are two exceptions to the general rule of needing to configure datasources in the administration console. See Sections 4.2.1 and 4.1.7 below for information on auto-configuration of ODBC datasources on Windows on all editions of BlueDragon, and the option of using J2EE datasources in BlueDragon/J2EE, both of which do not require definition of a datasource in the BlueDragon administration console.

### 4.1 Overview

This section describes the concepts of database drivers, connection pooling, auto-configuration of ODBC datasources, and using J2EE datasources (on the J2EE edition of BlueDragon). The final section of this chapter discusses configuring datasources with the BlueDragon administration console.

After a datasource is configured (or is detected as an ODBC datasource on Windows or as a J2EE datasource in the BlueDragon/J2EE edition), its name can be specified as a value for the `datasource` attribute for tags such as `CFQUERY`, `CFINSERT`, `CFUPDATE`, and `CFSTOREDPROC`.

#### 4.1.1 About Database Drivers

Database access in the Java-based editions of BlueDragon is based on the standard Java JDBC™ API. In order to access a specific vendor's database product (such as Microsoft SQL Server or Oracle), a JDBC driver for that product generally must be installed in BlueDragon (either those provided with BlueDragon, or those that you arrange to install yourself). BlueDragon provides built-in drivers and on some editions supports addition of other ones.



For additional information about JDBC, see:

<http://java.sun.com/products/jdbc/>

Database access in the .NET edition of BlueDragon is based on the standard ADO.NET libraries provided in the Microsoft .NET Framework.

BlueDragon also supports ODBC datasources when BlueDragon editions are deployed on Microsoft Windows, using the standard Java JDBC-ODBC Bridge on Java editions or ADO.NET ODBC support on the .NET edition.

On BlueDragon Server JX and the J2EE edition, drivers for Microsoft SQL Server (JTurbo®, New Atlanta's J2EE-certified JDBC driver) Oracle and PostgreSQL databases are bundled with BlueDragon. MySQL is also supported, as discussed in section 4.1.4. Additional drivers can be added as discussed in section 4.1.5. Finally, ODBC drivers are also supported when these editions are deployed on Microsoft Windows, as discussed in the next section.

The BlueDragon Server (Free Edition) supports only ODBC datasources on Windows, and only MySQL and PostgreSQL on Linux and OS X.

When working with ODBC datasources on Windows, or J2EE datasources in BlueDragon/J2EE, you do not need to install a specific database driver in BlueDragon.

### **4.1.2 Automatic Configuration of ODBC Datasources (Windows)**

On Windows platforms, when BlueDragon is installed it automatically finds any existing ODBC datasources (as configured by either the Windows ODBC Datasource Administrator or ColdFusion 5 or earlier) and imports the configuration information into BlueDragon. They are listed in the BlueDragon administration console under Datasources>Configure.

Note that when an ODBC datasource is auto-configured this way, the BlueDragon engine is not able to preserve any username, password, or server name that may be defined for that datasource in the Windows ODBC Datasource Administrator; therefore, you must manually add those values to the datasource definition in the BlueDragon administration console.

See Section 4.2 for more information on using the administrator console to create additional datasources as well as using the “ODBC Refresh” button to refresh the list of automatically configured ODBC datasources.

#### **4.1.2.1 Preventing Auto-Configuration of ODBC Dataources**

While this feature of auto-loading ODBC datasources is generally considered a productivity enhancement, it may be desirable in some situations to prevent the process. To do so, simply delete the `ODBCNativeLib.dll` from the BlueDragon installation.

On the Server or Server JX editions, this file is found in [bluedragon-install]\lib directory. Restart the server after deleting the file for the change to take effect.

On the J2EE edition, it's found in [webapp]\WEB-INF\bin\ directory. Redeploy or restart the web application after deleting the file for the change to take effect.

It's not currently possible to stop this behavior in the .NET edition.

### **4.1.3 Configuring Access, SQL Server, and Other ODBC Datasources**

The converse of the previous section is that there is no option in the BlueDragon administration console for adding ODBC datasources such as Access. And while the Server JX and J2EE editions of BlueDragon provide a driver for SQL Server (and other databases like Oracle), the free Server edition does not. Does this mean you can't use an Access datasource in BlueDragon? Or a SQL Server database in the free Server edition? No. It's just that you must instead create the datasource outside of BlueDragon as described in section 4.1.2 and then use the ODBC Refresh option also described there to see the datasource.

### **4.1.4 Support for MySQL**

BlueDragon supports MySQL using either JDBC or ODBC. In the Server JX and J2EE editions, the administration console option for adding a datasource shows an available MySQL driver in the list of choices of drivers offered, but note that the driver is not installed with BlueDragon. See "Support for MySQL" in the *BlueDragon 6.2.1 Installation Guide* for more information on preparing the environment to support MySQL, using either JDBC or ODBC drivers.

### **4.1.5 Configuring Datasources Not Listed**

If you'd like to use a database other than those listed in the BlueDragon administration console, you may be able to, if you have the needed JDBC driver (and are running other than BlueDragon Server FREE edition). For more information, see the BlueDragon User's Guide as well as FAQ 215 on New Atlanta's web site:

[http://www.newatlanta.com/biz/c/products/bluedragon/self\\_help/faq/home](http://www.newatlanta.com/biz/c/products/bluedragon/self_help/faq/home)

Recall as well that on Windows, you may use any datasource for which you can configure an ODBC datasource, as discussed in section 4.2.2.

### **4.1.6 Database Connection Pooling**

BlueDragon implements a database connection pooling mechanism that provides efficient access to datasources. For each datasource, you can configure the maximum number of connections in the pool and a timeout value that determines when idle connections are released from the pool.

### 4.1.7 J2EE Datasources

Note that on the J2EE edition, if you choose to use J2EE datasources, then connection pooling and other beneficial features are provided by the J2EE server and its datasource administration.

When using BlueDragon for J2EE, you can either use the datasource configuration capability discussed above, or you can instead use the datasource configuration capability provided by your J2EE application server. In that case, you do not need to configure the datasource in the BlueDragon administration console.

Once configured, you can use the datasource name (technically the JNDI name, in J2EE parlance) in tags like CFQUERY, et cetera. BlueDragon will search first to see if there is a datasource of that name in its configuration, and if none is found, it will then look for the JNDI name in the J2EE environment.

Defining datasources in the J2EE environment may give you additional benefits, such as deployment, clustering, replication, and more. Consult your J2EE server documentation.

## 4.2 Configuring Datasources

The remainder of this chapter discusses datasource configuration in the BlueDragon administration console.

Click the configure link under the `Datasources` heading in the BlueDragon administration console menu to access the datasource configuration pages. The screen will appear similar to Figure 4, below. Note that the status for all datasources is initially listed as `not verified`.

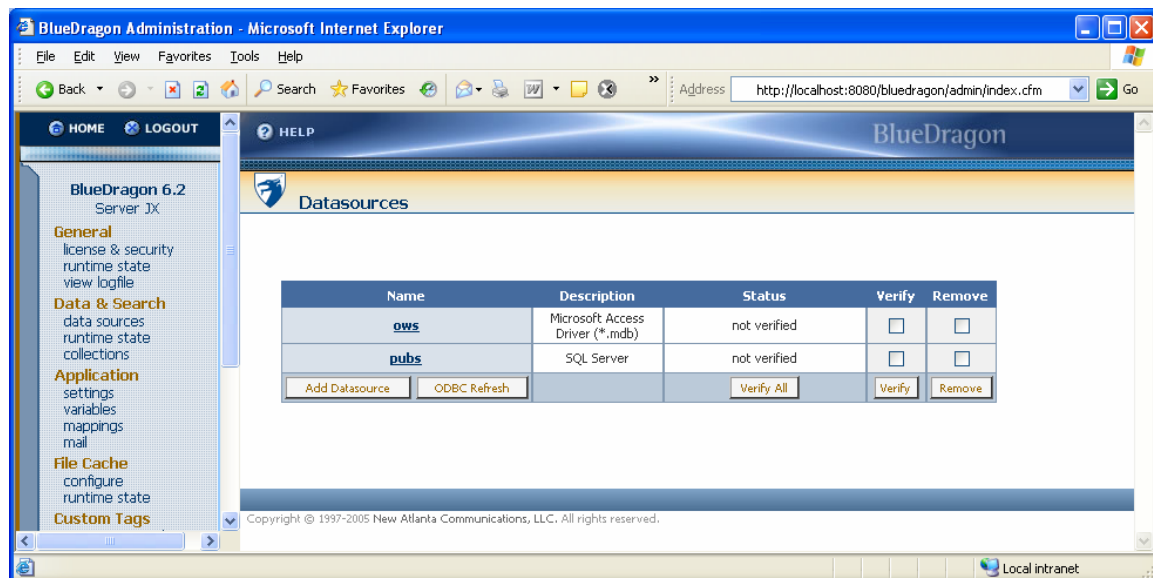
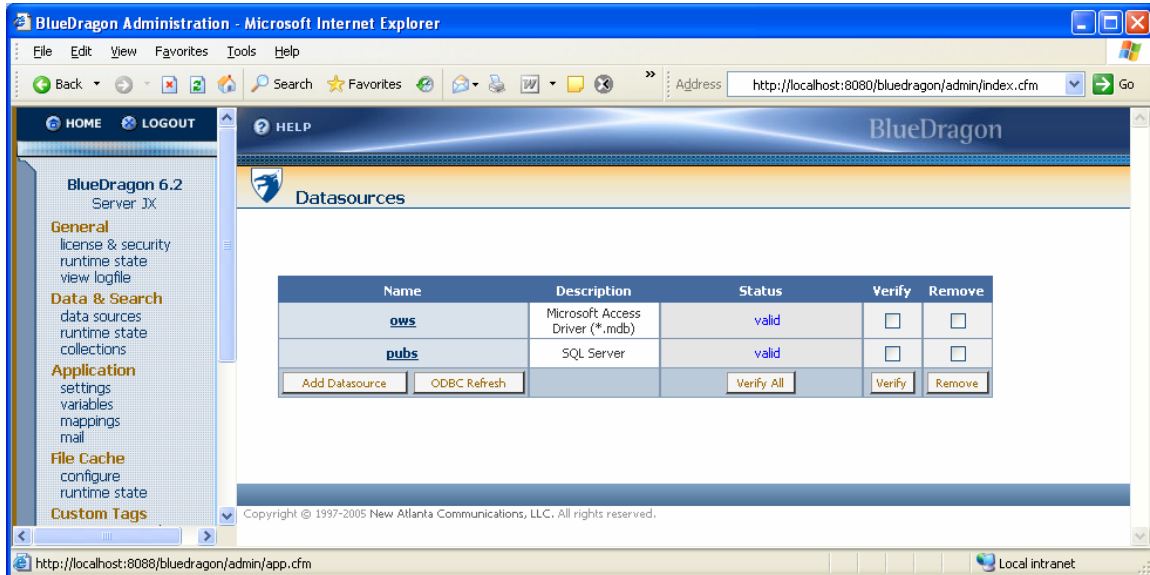


Figure 4. Datasource Configuration (not verified)

You can verify BlueDragon's ability to connect to configured datasources, whether a single datasource (using the `Verify` checkbox and `Verify` button) or all datasources with the `Verify All` button. Figure 5 shows a sample screen following verification of all datasources.



**Figure 5. Datasource Configuration (verified)**

#### 4.2.1 Possible Timeout Errors When Verifying All Datasources

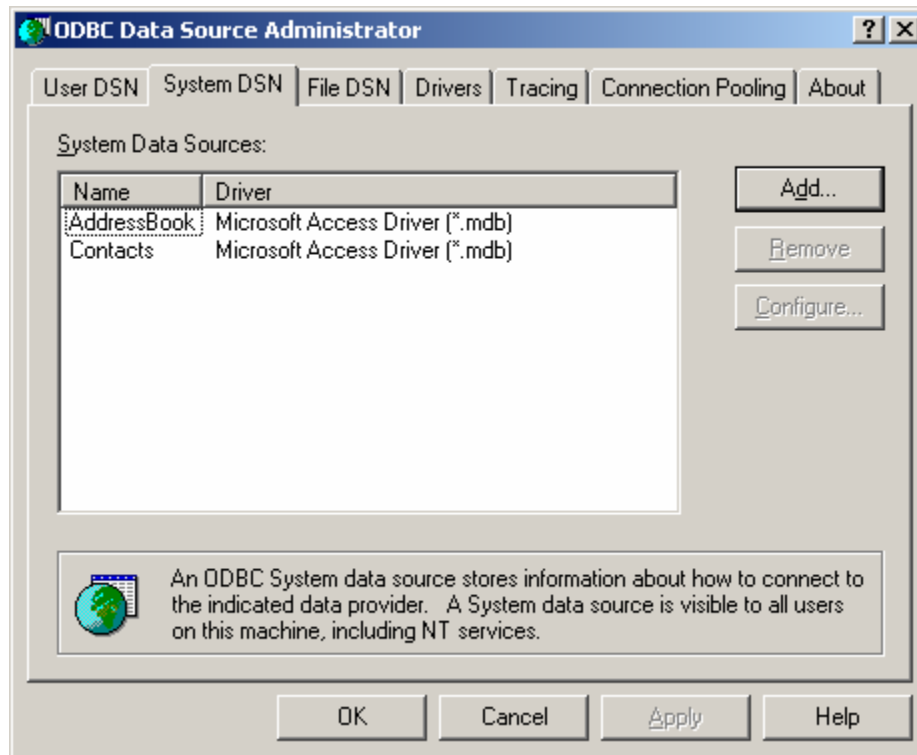
Beware that if you have a large number of datasources, using `Verify All` could take an excessive amount of time and could possibly timeout. The default timeout for failed connections is two minutes. Therefore, it could take several minutes to verify all datasources depending on how many are configured and how many fail.

Further, on the .NET edition, there is also an imposed timeout for processing of CFML pages, typically set in the .NET configuration files, which might also prevent completion of an attempt to verify all datasources.

If you have more than a few datasources to verify, simply verify a few at a time. If all datasources are valid, verification should take only a few seconds.

#### 4.2.2 ODBC Datasource Auto-Configuration

On Windows, BlueDragon automatically detects all datasources configured as System DSNs in the Windows ODBC Data Source Administrator and makes them available for use in CFML tags. This is discussed further in section 4.1.2, including how to prevent this behavior. For example, Figure 6 below shows two ODBC datasources configured for Microsoft Access databases.

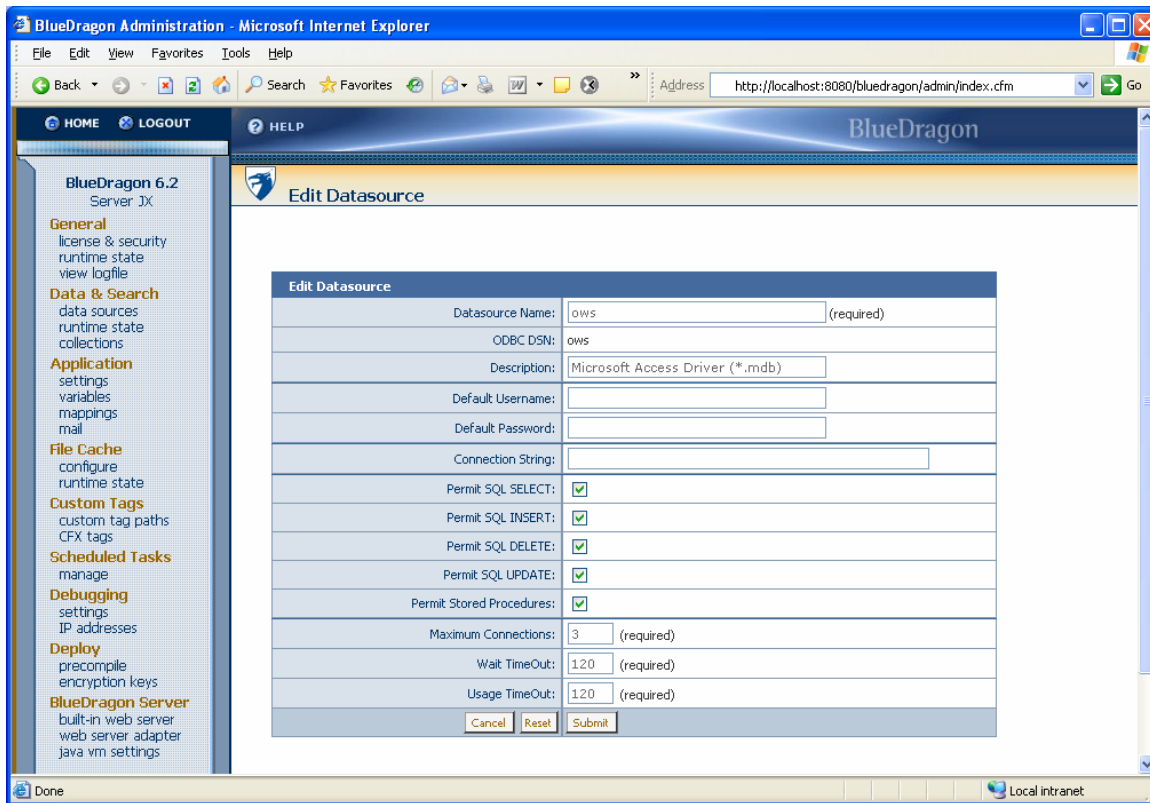


**Figure 6. ODBC Datasources**

These datasources are automatically detected and configured by BlueDragon as illustrated in Figure 4, above.

If you add a new ODBC datasource via the Windows ODBC Data Source Administrator while BlueDragon is running, use the ODBC Refresh button to have BlueDragon re-detect and auto-configure the new ODBC datasource, also shown in Figure 4, above.

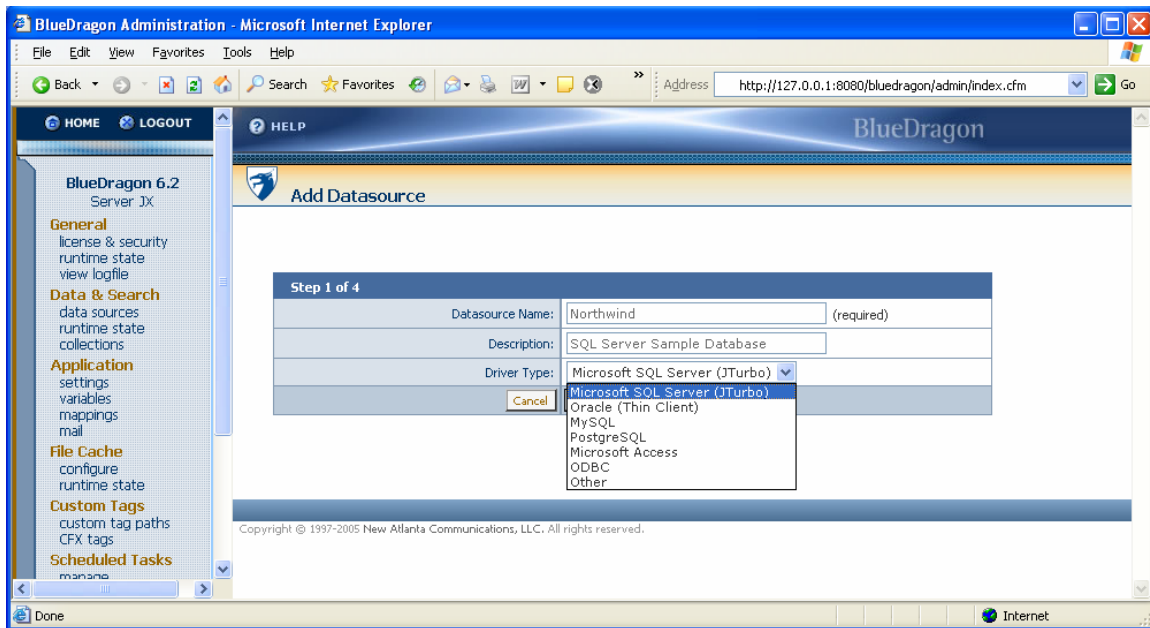
After an ODBC datasource is auto-configured by BlueDragon, you may edit the datasource to change the Datasource Name used by BlueDragon to identify the datasource in CFML tags such as `CFQUERY`, or the Description associated with the datasource. You may also edit the Default Username and Password, and in fact if those were defined for the datasource in the Windows ODBC configuration, you must re-enter those manually here. Finally, you may edit the CFML permissions, connection pooling configuration, and other details for the datasource, as illustrated in Figure 7, below.



**Figure 7. Editing an ODBC Datasource**

### 4.2.3 Add Datasource Wizard – Step 1

Click the Add Data Source button to start the Add Data Source wizard. Step 1 of the Add Datasource wizard is illustrated in Figure 8, below. Enter values for the form fields, described in the following sections.



**Figure 8. Add Data Source Wizard – Step 1**

**Datasource Name.** This required field is the name that will be specified as the value for the datasource attribute for CFML tags such as `CFQUERY`, `CFINSERT`, `CFUPDATE`, and `CFSTOREDPROC`. For example, if you configure a Datasource Name of “MyCompany”, this could be used in a `CFQUERY` tag as follows:

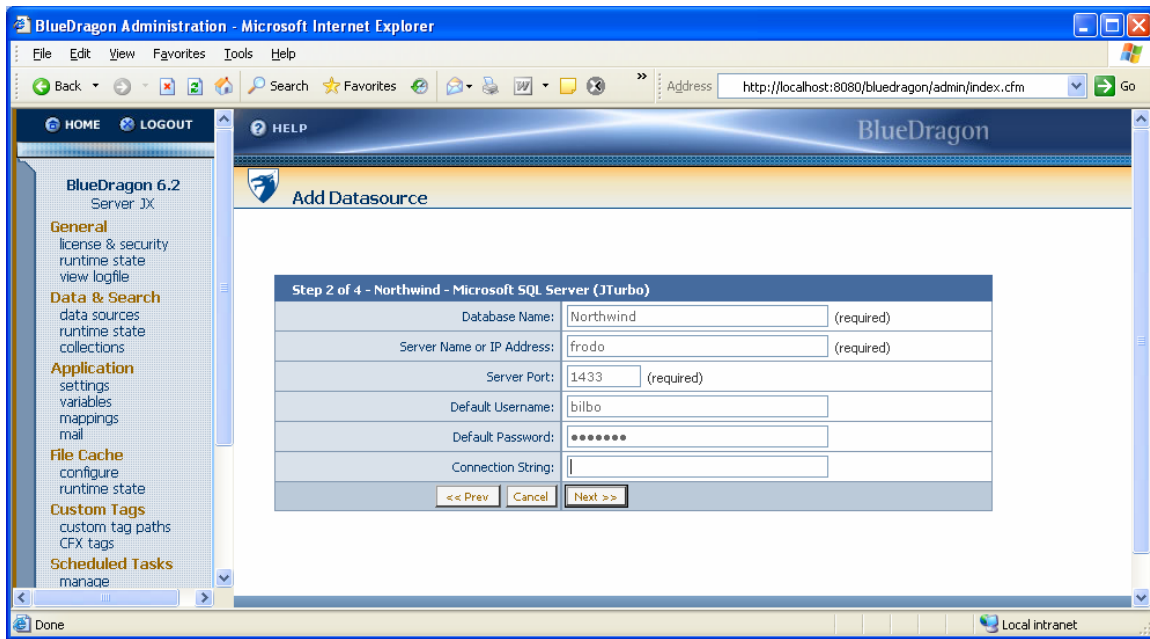
```
<CFQUERY DATASOURCE="MyCompany" NAME="employees">
SELECT * FROM Employees
ORDER BY HireDate
</CFQUERY>
```

**Description.** This field is for the convenience of the BlueDragon administration console. It can be used to enter a plain text description of the datasource. This field is optional and is not used by BlueDragon other than for display purposes.

**Driver Type.** As illustrated in Figure 8, above, BlueDragon is bundled with JDBC drivers for PostgreSQL and (on the Server JX and J2EE editions), Microsoft SQL Server, and Oracle databases. For more information on these and other database drivers supported by BlueDragon, see section 4.1.1. Select the appropriate driver for your database, or choose “Other” if you’ve installed and configured a different JDBC driver other than the ones provided (the “Other” option is not offered on the BlueDragon Server FREE edition).

After a datasource is created, you may not change the Driver Type.





**Figure 9. Add Data Source Wizard – Step 2**

#### 4.2.4 Add Datasource Wizard – Step 2

If you selected “Microsoft SQL Server (JTurbo),” “Oracle (Thin Client),” “PostgreSQL,” or “MySQL” as the Driver Type in Step 1, Step 2 of the Add Datasource Wizard appears as illustrated in Figure 9, above.

On Server JX or the J2EE edition, if you selected “Other” as the Driver Type in Step 1, the information to be provided in Step 2 is different than what’s shown in Figure 9. See the help panel for Step 2 for detailed instructions for completing the configuration for Driver Type of “Other.”

**Database Name.** This required field is the name of the database that will be accessed by this datasource. For example, Figure 10, below, illustrates a Microsoft SQL Server 2000 configuration. In this illustration, some valid Database Names are “Northwind” and “pubs” (these are sample databases that ship with SQL Server 2000).

**Server Name or IP Address.** Enter either the Windows computer name, the TCP/IP domain name, or the IP address of the server machine on which your database is installed. In Figure 9, “frodo” is the Windows name of the server computer.

**Server Port.** Based on the Driver Type selected in Step 1, the Add Datasource wizard will set the default for the port number. If you’ve configured your database to use a port other than the default, change the port number to the appropriate value.

**Default Username.** If you do not specify the `username` attribute in database tags such as `CFQUERY`, this is the username that will be used to log in to the database. This field can be left blank if your database does not require a username, or if you always specify the `username` attribute in database tags.



**Default Password.** If you do not specify the `password` attribute in database tags such as `CFQUERY`, this is the password that will be used to log in to the database. This field can be left blank if your database does not require a password, or if you always specify the `password` attribute in database tags.

**Connection String.** Allows you to specify additional parameters that you wish to be passed to the database.

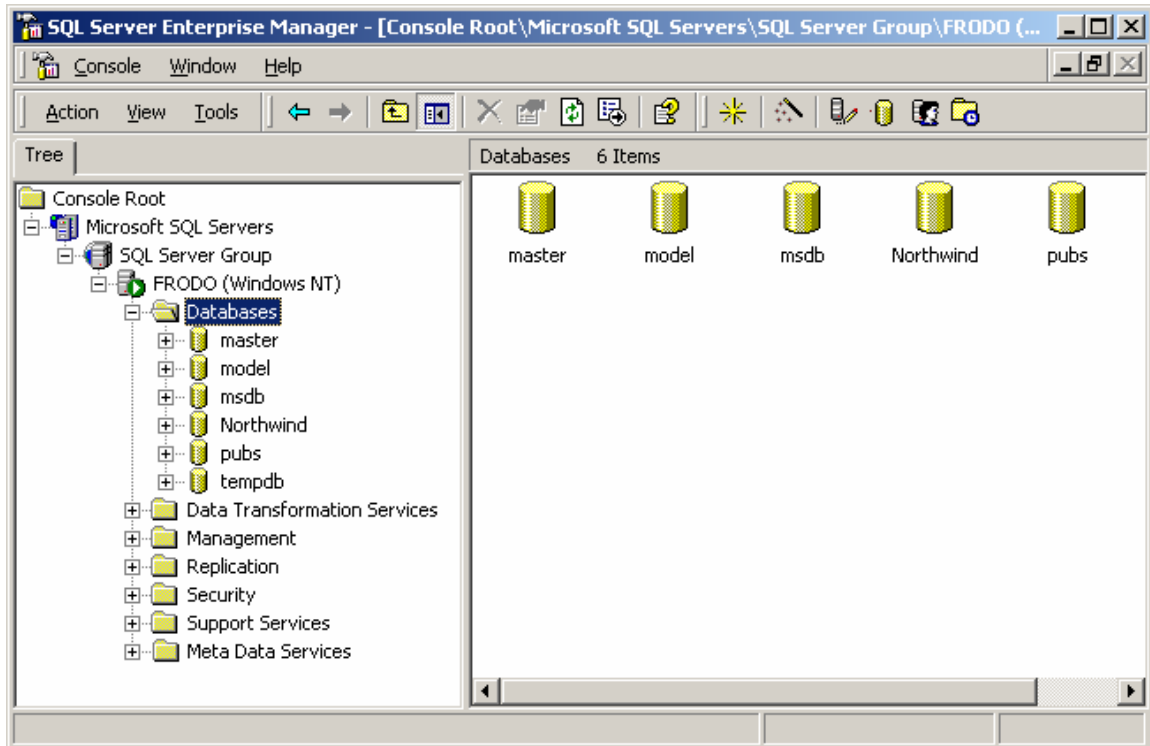
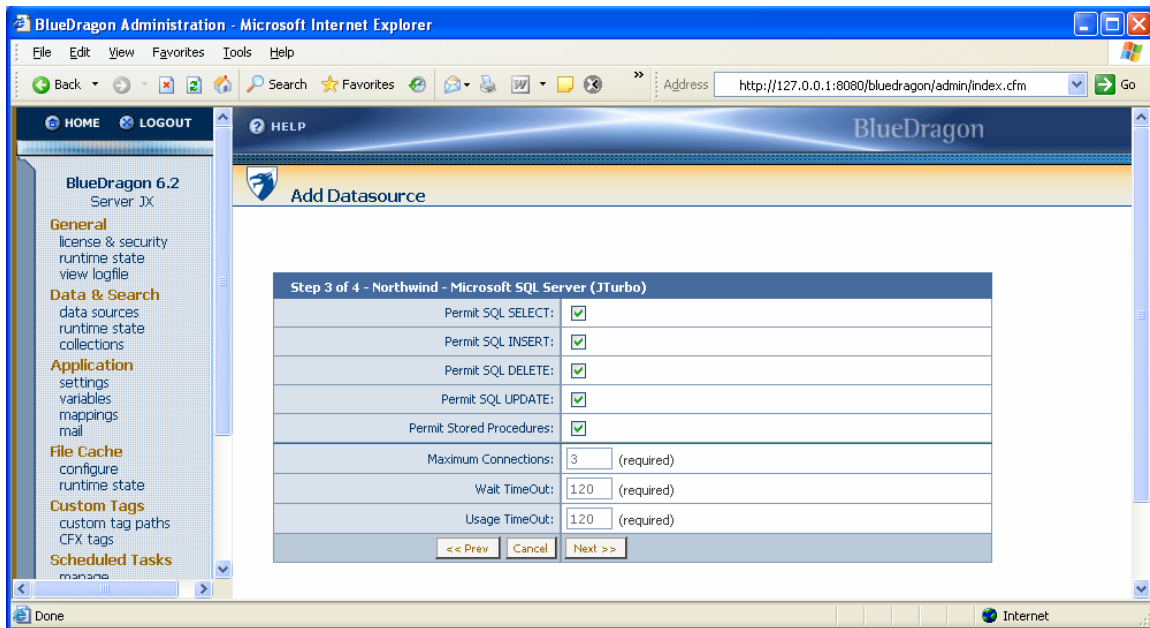


Figure 10. Microsoft SQL Server 2000 Databases



**Figure 11. Add Data Source Wizard – Step 3**

#### 4.2.5 Add Datasource Wizard – Step 3

Step 3 of the Add Data Source wizard allows you to set security and connection pooling parameters as illustrated in Figure 11, above.

**Permit SQL SELECT.** If checked, BlueDragon will allow SELECT statements to be executed on this datasource. If unchecked, SELECT statements are not allowed.

**Permit SQL INSERT.** If checked, BlueDragon will allow INSERT statements to be executed on this datasource. If unchecked, INSERT statements are not allowed.

**Permit SQL DELETE.** If checked, BlueDragon will allow DELETE statements to be executed on this datasource. If unchecked, DELETE statements are not allowed.

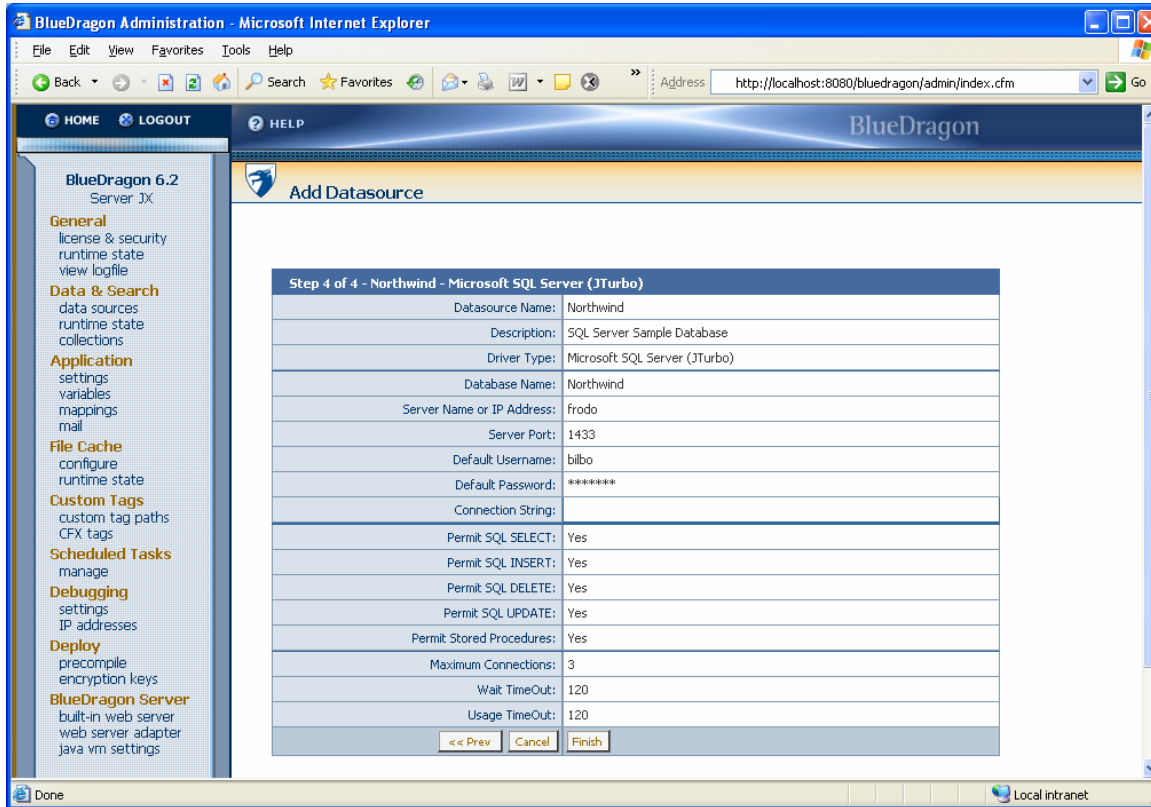
**Permit SQL UPDATE.** If checked, BlueDragon will allow UPDATE statements to be executed on this datasource. If unchecked, UPDATE statements are not allowed.

**Permit Stored Procedures.** If checked, BlueDragon will allow stored procedures to be executed on this datasource. If unchecked, stored procedures are not allowed.

**Maximum Connections.** The maximum number of connections that will be created and placed into the connection pool for this datasource. The default is 3.

**Idle Timeout.** The time in seconds after which idle connections will be released from the database connection pool for this datasource. The default is 120.

**Connection Timeout.** The timeout in seconds for creating new database connections for this datasource. If a new connection can't be created within this time an error is generated. The default is 120.



**Figure 12. Add Data Source Wizard – Step 4**

#### 4.2.6 Add Datasource Wizard – Step 4

As illustrated in Figure 12, above, step 4 of the Add Data Source wizard allows you to review all of the configuration options before adding a new datasource. Click the **Prev** button to go back and make any changes to the configuration information. Click the **Finish** button to add the new data source. Click the **Cancel** button to discard all entries and terminate the Add Data Source wizard without adding the new datasource.

On clicking **Finish**, the datasource will be verified automatically and the results reported on the next page, as shown in Figure 13.

### 4.3 Modifying and Deleting Datasources

As illustrated in Figure 13 and Figure 14, below, a datasource can be edited by clicking on the datasource name from the Configure Datasources page of the BlueDragon administration console. To delete a datasource, select the checkbox in the **Remove** column, then click the **Remove** button. Multiple datasources can be deleted by selecting multiple checkboxes.

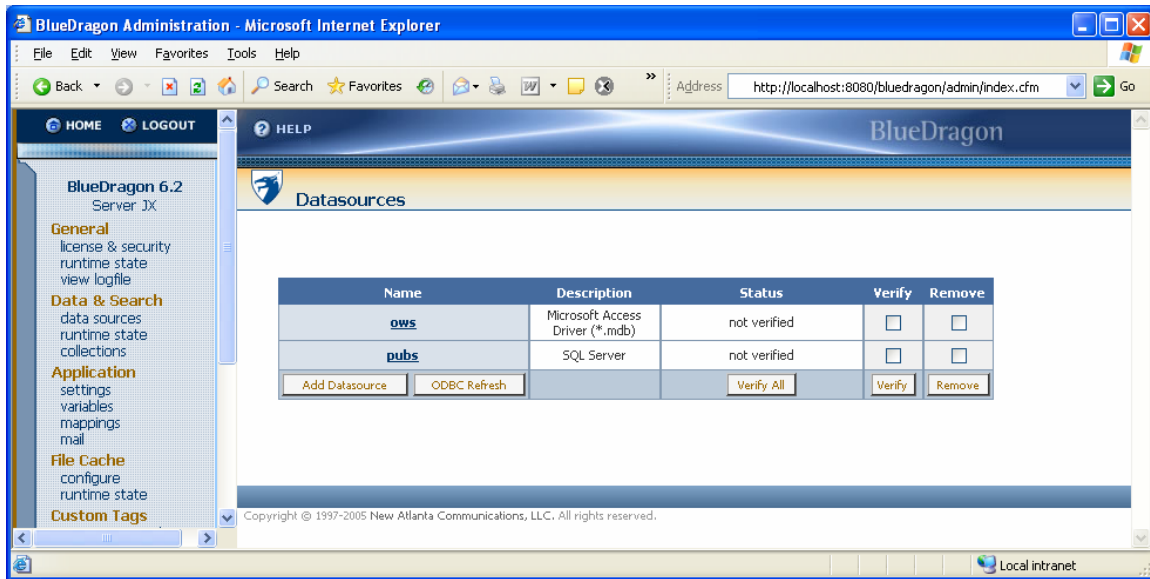


Figure 13. Managing Data Sources

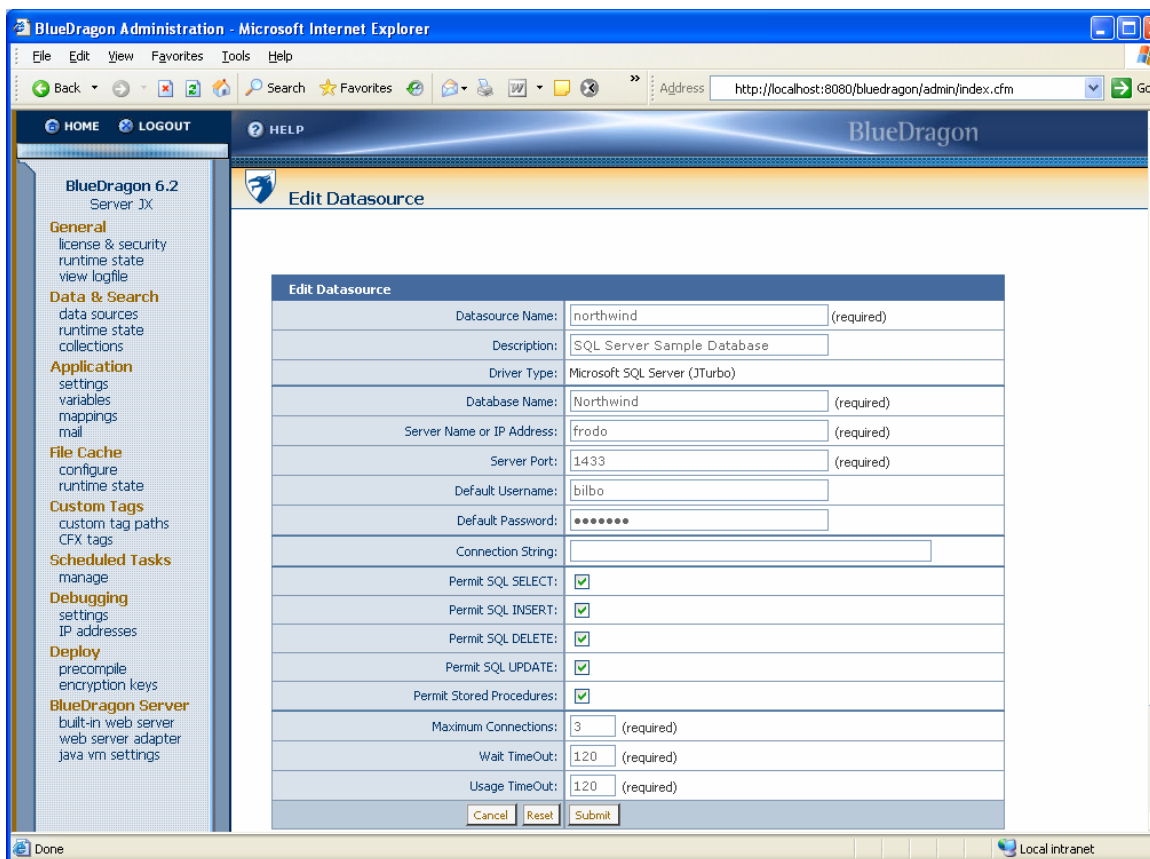


Figure 14 . Editing Data Sources

## 5. Upgrading from ColdFusion

**B**lueDragon offers a CFML runtime environment very similar to ColdFusion Server. Where the language support for tags and functions differ, the details (and available work-arounds) are offered in the *BlueDragon 6.2.1 CFML Compatibility Guide*. This section discusses other aspects of working with ColdFusion that differ in ways that current ColdFusion users may not expect.

### 5.1 Administration Console/Configuration Differences

There are various aspects of working with the administration console and its default configuration settings that are slightly different in BlueDragon (some limitations and some enhancements).

#### 5.1.1 Site-wide Error Handling Templates

The BlueDragon administration console supports the definition of a site-wide error handling template as well as one to handle missing templates, as in ColdFusion. Note that BlueDragon allows the path to be specified either as full physical paths or as a relative from the application root directory.

#### 5.1.2 Server Debugging Output

The BlueDragon administration console supports the enabling of server debugging information that can be output at the bottom of each CFML template, as in ColdFusion. You should be aware of some differences from CF's implementation of this feature.

First, like CFMX (and an improvement over CF5), BlueDragon lets you select whether to display variable values (as of the end of the request) for any or all of the available scopes (including application, session, server, and more). Unlike CFMX, BlueDragon allows selection of the `variables` scope.

The BlueDragon administration console supports the enabling of display of template execution time information that can be output at the bottom of each CFML template, as in ColdFusion. Note that while the ColdFusion MX administrator settings for debugging output offers a Report Execution Time format called “tree”, BlueDragon supports only the equivalent to CFMX's “summary” format. Note also that while the ColdFusion MX administrator settings for debugging output offers a format called “docakable”, BlueDragon supports only the equivalent to CFMX's “classic” format.

Finally, while both ColdFusion and BlueDragon allow specification of IP addresses that should receive debugging information, BlueDragon allows you to specify wildcards (the “\*” symbol) to indicate a range of addresses, such as 200.129.3.\*.

### 5.1.3 CFML Tag and Function Security

In ColdFusion, you can limit whether developers can use certain tags such as `CFFILE`, `CFDIRECTORY`, `CFREGISTRY`, and more. BlueDragon does not provide a mechanism to limit tags this way. Instead, we recommend that you deploy multiple independent instances of BlueDragon and use operating system or application server security controls to limit what resources that instance is allowed to access.

### 5.1.4 Administration Console Mappings

The BlueDragon administration console supports mappings (for use with `CFINCLUDE` and `CFMODULE`, for instance). There are some aspects about this feature that differ from ColdFusion.

In the Mappings section of the BlueDragon administration console, the Server and Server JX editions implicitly map “/” to the web server document root directory for either the built-in web server, or the external web server if an adapter is installed. BlueDragon/J2EE implicitly maps “/” to the J2EE web application root directory. You may override these implicit mappings.

Be aware of this distinction about how the implicit mapping changes depending on whether an external web server adapter has been configured. Code running in the BlueDragon wwwroot may not function as expected with the implicit root mapping pointing to code in the web server’s docroot.

Separately, when giving the full path in Linux and Mac OS X, BlueDragon assumes that the path provided is relative to the web root unless you prepend it with a \$, in which case it will be considered relative to the server root. For example, a mapping of `/mysite/` to `/home/mysite/www/` should instead be written as `$/home/mysite/www/`.

### 5.1.5 WhiteSpace Compression

BlueDragon’s whitespace compression is more aggressive than ColdFusion’s, striking a balance between compression (to reduce bandwidth) and adverse impact on runs of HTML code that may require preserving whitespace as coded. See the discussion of “WhiteSpace Compression” in the “Miscellaneous” section of the *CFML Compatibility Guide* for additional details.

### 5.1.6 Administration Console Extended Features

There are various features in the BlueDragon administration console that are not found in the ColdFusion MX administrator. Some of the key features to explore include:

- `General>runtime state` – This page, one of several “runtime state” pages in the administration console, offers several useful pieces of information including total

BlueDragon uptime, total hits and bytes served, mails sent, datasources in use, total applications and sessions, and more.

- `Data&Search>runtime state` – This page offers more details about all datasources in use.
- `Application>settings` – There is a new Default Response Buffer Size setting. This is discussed further in section 5.2.8. There is also a new Default charset setting, allowing you to change the default character set used for CFML page processing.
- `File Cache>runtime state` – This page offers more details about all CFML pages in the (in memory) template cache. Note that, additionally, this page offers a Flush Cache button allowing you to flush the template cache if you choose to turn off trusted cache in order to update code when using the trusted cache feature. (In ColdFusion, you’d need to restart the server to flush the cache when updating code after disabling trusted cache.)
- `Debugging>settings` – Note the available Enable Assertions option. See the discussion of the available CFASSERT tag and assert function, unique to BlueDragon, in the *BlueDragon 6.2.1 CFML Enhancements Guide*.
- `Debugging>IP addresses` – Note that BlueDragon permits wildcard values in the IP addresses designated to see debugging information, if enabled. See section 5.1.2 for more information.
- `Deploy>precompile` – This option processes CFML templates to use the newly available precompiled, encrypted templates feature in BlueDragon. See section 5.2.3 for more information.
- `General>license&security>Allowed IP Addresses` – For more information, see the *BlueDragon 6.2.1 Installation Guide*, in the section “Changing the Default Localhost Limitation for the Built-in Web Server”.

Note that the administration console offers online help on nearly every page, offering more details on these and other features.

### 5.1.7 CFX Custom Tags

BlueDragon supports using CFX custom tags. For more information on installing and registering them, including where to place their supporting files, see the online help in the BlueDragon administration console. Note especially the help provided on the page displayed after clicking “Register CFX”.

When deploying a CFX class on the Java editions of BlueDragon, be sure to place any supporting class files in the classpath as well.

Note that it's not necessary to register CFX custom tags if their underlying code implementation uses no package name (in Java) or namespace (in .NET). Packages and namespaces are helpful, though, for organizing related classes and avoiding clashes among similarly named classes.

### 5.1.8 File Cache/Template Cache

Where ColdFusion (5 and MX) defines a "template cache" as a place to hold templates in memory once rendered from source code, BlueDragon has the same notion but refers to this as the "file cache". In both engines, a template once rendered from source will remain in the cache until the server (or J2EE or .NET web app) is restarted.

The cache size, specified in the Admin Console, indicates how many of these cached templates to keep. It defaults to 60 but that number may need to change for your application, depending on how many CFML templates your application uses. One entry is used for each template (CFM or CFC file) requested.

It's very important to understand that this is not caching the OUTPUT of the page but rather the rendering of the template from source into its internal objects. One cached instance of the template is shared among all users in the application.

As in ColdFusion, once the file cache is full (for instance, you set it to 60 and 60 templates have been requested), then the next request for a template not yet cached will force the engine to flush the oldest (least recently used) entry in the cache to make room. Naturally, if you set this file cache size too low, thrashing in the cache could occur as room is made for files only to soon have the flushed file requested again.

BlueDragon offers a couple of advantages in managing the template cache. First, our Admin Console can offer a detailed report on the status of the file cache. (See the `File Cache>Runtime State` link, which lists each page in the template cache as well as how many times it was requested.) There it also a higher-level report simply indicating both the current file cache size and how many files are in the cache currently (See the `General>Runtime state` link.) These reports can help you determine the optimal size for your environment.

You can set the size of the file cache in the `File Cache>configure` link. As with ColdFusion's "trusted cache" setting, the available `trust cache` option there controls whether files in the template cache should be updated once loaded if the source code of their corresponding CFML template has changed. The default is `no`, which makes sense in development.

In production, there can be a slight performance improvement by setting the `trust cache` option to `yes`. Note that, unlike ColdFusion, BlueDragon offers an available `Flush Cache` button on the `File Cache>Runtime State` page. This will immediately flush the template cache so that all previously cached templates are removed.

Finally, note as well that the *BlueDragon 6.2.1 Server and Server JX Installation Guide* discusses an available `cacherealpath` configuration option. For those two editions of



BlueDragon, when running in a multi-homed environment (multiple web sites served by one instance of BlueDragon), the setting of that value can have an important impact on the file cache.

## 5.2 Other Features Related to ColdFusion

There are a few other ColdFusion features, not having to do with tags or functions, which are not supported in BlueDragon. In each case, however, there are alternatives or work-arounds allowing similar functionality.

### 5.2.1 CFENCODed CFML Templates

BlueDragon cannot currently execute CFML templates that have been encoded with ColdFusion's CFENCODE utility (previously known as CFCRYPT), which can be a challenge if custom tag or application providers do not provide source code. New Atlanta is considering adding that support in a future update.

Customers interested in an alternative mechanism for protecting their CFML source code when executed on BlueDragon should consider the available precompiled, encrypted templates feature, discussed in section 5.2.3.

### 5.2.2 CFC and Web Service Browsing

While BlueDragon does support the use of both CFCs and web services (consuming and publishing them), there are some aspects of processing these files from a web browser that differ from ColdFusion MX.

First, ColdFusion MX offers a CFC browsing tool, where you can request a URL naming the CFC (such as: `http://servername/cfcname.cfc`) to be shown a formatted screen displaying all the available methods and properties. The Java editions of BlueDragon do not support that feature. A browser request for a URL such as that will return a 403 status code (forbidden). On BlueDragon.NET, however, you will be shown the traditional .NET web service browser (as would show when browsing an .asmx file).

As an alternative, to be able to see what a CFC's methods and properties are, you can invoke a CFC without invoking a method (such as with `CFOBJECT`) and then `CFDUMP` the result, which will show the properties and methods of the CFC. An example is:

```
<cfobject component="cfcname" name="somevar">
<cfdump var="#somevar#">
```

It would be trivial to create a tool that accepted a CFC name and performed this action, returning the result to the user.

Second, note that it is possible to execute a web service method via a URL request. Both BlueDragon and ColdFusion MX can accept browser request offering a URL that request the execution of a given web service method (a method of a CFC exposed with `ACCESS="Remote"`). An example is:

```
http://servername/cfcname.cfc?method=methodname
```

While ColdFusion will return any resulting data in WDDX format, BlueDragon will display it as pure XML (in the form of a SOAP packet). Note that this difference has no effect on processing of web services using `CFINVOKE/CFOBJECT/createObject`.

### 5.2.3 Precompiled, Encrypted CFML Templates

BlueDragon adds an enhanced mechanism to protect the intellectual property in your CFML templates, by way of a new precompiled, encrypted templates feature. BlueDragon offers the option of either precompiling or additionally encrypting the templates.

#### 5.2.3.1 Benefits of Precompiled Templates

Both features offer many benefits over ColdFusion's "encoded templates" feature, which simply encodes templates to make them no longer human readable.

- These precompiled/encrypted templates are not only more secure, they are also better performing since they contain a precompiled version of the code
- Only the BlueDragon runtime can process the precompiled/encrypted templates, and there is no mechanism to return the templates to their original CFML form
- There is no service that can be offered (by New Atlanta or any third party) to recover or "decode" our precompiled/encrypted templates
- Only BlueDragon offers a mechanism to set an expiration date for protected templates. Users cannot continue to use the precompiled/encrypted templates after the expiration date
- Only BlueDragon offers a mechanism (in the form of the encrypted templates option) to prevent code being executed without a valid license key

#### 5.2.3.2 About Encrypted Templates

To extend that last point, if you choose to encrypt templates in addition to precompiling them, this adds another measure of security in preventing the code being run by unauthorized users (whereas precompiling already prevents the code being viewed or returned to source). With encrypted templates, only a BlueDragon engine having the decryption key can process the templates. The following points explain this feature:

- When encrypting templates, you have the choice of using a default key or a custom encryption key
- There is a default key provided in all BlueDragon installations that can process templates encrypted with the default key

- Vendors can create different keys to encrypt their code, such that only BlueDragon installations that have implemented the decryption key (by way of the administration console) can process templates they've encrypted
- Several decryption keys can be specified in the administration console to support code offered by multiple vendors/providers of encrypted code
- This encryption mechanism not only prevents users seeing the CFML code, it also prevents the code from running on a BlueDragon installation that lacks the decryption key (assuming the code was encrypted with other than the default encryption key)
- The use of multiple custom encryption keys can also be used to control specific directories within a web application. A vendor could distribute all parts of an application but know that only those users who've implemented the decryption key for code in specific directories will be able to run that code

### 5.2.3.3 Creating Precompiled and Encrypted Templates

You can create precompiled, encrypted templates from your CFML source code using either a command line tool or an option in the BlueDragon Administration Console, in the `Deploy>precompile` section.

Note that the BlueDragon Server FREE edition can not be used either to create or deploy/execute precompiled/encrypted templates.

The command line tool is offered with the Server JX edition in the `bin` directory as `precompile.bat` on Windows and `precompile.sh` on Linux/OS X. In the .NET edition, it's available as a `precompile.exe` file, found in the `precompiler` directory of the BlueDragon.NET installation. The command line tool prompts you to enter the needed arguments (and it can perform encryption as well, which is offered as a prompted option.)

For those interested in automating the precompilation process, note that while it's not possible to pass the arguments on the `precompile` command itself, you can use redirection to pass the contents of a file to the `precompile` command, such as with:

```
precompile.bat < somefile.txt
```

BlueDragon automatically creates a `precompile.log` file logging the results of the precompilation, which is placed in the BlueDragon `work` directory. In Server JX, this location would typically be `C:\BlueDragon_Server_JX_62\work`. In the J2EE and .NET editions, the work directory depends on how you installed BlueDragon. See their respective documents, *Deploying CFML on J2EE Application Servers* and *Deploying CFML on ASP.NET and the Microsoft .NET Framework* for more information on locating the work directory.

When using the precompilation feature, note that while the process creates new copies of the precompiled CFML files in a named destination directory, it does not copy other files

from your web application, such as HTML, GIF, JPG, and other file types. You must manage that process manually.

Note that in the Server JX and J2EE editions of BlueDragon, the encrypted templates feature requires support for the Java Cryptography Extension (JCE), which is included with JDK 1.4. Therefore, that JDK is required to use encrypted templates (and it is included in the Server and Server JX installation). See the *BlueDragon 6.2.1 Installation Guide* for more details on system requirements.

Finally, note that currently it is not possible for code precompiled or encrypted in one version of BlueDragon to run in another version (either a later version, or between the Java and .NET editions of BlueDragon). You simply need to re-run the precompilation/encryption step against your source code. A future release of BlueDragon may eliminate this challenge.

### 5.2.4 Flash, Flash Remoting, and Other Flash Integration

While BlueDragon does not currently provide built-in support for server-generated Flash components or a Flash Remoting engine, it is still possible to integrate BlueDragon with Flash components you may create, whether using the Flash development environment, Flex or CFMX 7, Flash Remoting, or third party and open source alternatives to these solutions.

Flex and Flash Remoting are proprietary Macromedia technologies, the former enabling XML-driven server-side creation of Flash components and the latter enabling execution of and communication with server-side objects from Flash clients.

Laszlo is an open source alternative to Flex available from Laszlo Systems. Third party alternatives to Flash Remoting include the open source project OpenAMF project and WebOrb (formerly FlashOrb) from Midnight Coders, which works not only on J2EE application servers but .NET as well. For more information about each of these respectively, see:

- <http://www.laszlosystems.com>
- <http://www.openamf.org>
- <http://www.themidnightcoders.com/>

All these tools support communication with web services, and BlueDragon supports execution of CFCs as web services. In fact, WebOrb has been extended to be able to call CFCs directly, without using web services (and it supports both Flash Remoting and Ajax-style communication to the server). See the WebOrb site (<http://www.themidnightcoders.com/>) for specific details on integrating it with CFCs on BlueDragon.

As for users of Flash Remoting (and OpenAMF), consider the following line of ActionScript in a Flash client using Flash Remoting to call a CFC directly:

```
Svc = Conn.getService("myCFC", this);
```

This could easily be changed to make that same request as a web service:

```
Svc = Conn.getService("http://[myserver]/myCFC.cfc?wsdl", this);
```

While web services may not be as compressed as Remoting's Action Messaging Format (AMF), they are certainly a more open standard than AMF.

And while Macromedia's support of CFMX in Flash Remoting performs extra processing on CFQUERY resultsets returned to automatically handle them as native recordset objects in the client, it's also possible to use CFML on BlueDragon to turn such resultsets into more standard datatypes that can be processed by any client (such as an array of structures). See the CFLib site (<http://www.cflib.org>) for information on available user defined functions to perform such actions.

Note as well that Flash 7 and Flash MX 2004 enable calling of web services directly from the Flash client without need of Flash Remoting to act as a proxy.

Additionally, even CFML templates can be called from Flash Remoting and its alternatives, using LoadVariables, LoadVars, LoadXML, etc. See various Flash resources for more information on these capabilities, which work with any back-end web application, including BlueDragon. An excellent resource to help you make connections using these various approaches is the new book from noted CFML author Nate Weiss, *Flash MX 2004 Professional for Server Geeks*, from New Riders. The book covers use of CFML (much of it applying to BlueDragon as well), in addition to also showing code examples in J2EE and ASP.NET.

Note that besides simply integrating with an existing deployment of Flex, Laszlo, FlashOrb, OpenAMF, and others, it's also possible to directly embed any of these tools into a BlueDragon deployment, just as may be described about doing so with ColdFusion. With respect to the J2EE edition of BlueDragon in particular, most of these tools are J2EE web applications as well, and where their (or Macromedia's) documentation discusses merging these web applications into ColdFusion's J2EE deployment, the same steps apply to merging them into BlueDragon's.

We are investigating embedding a third-party Flash Remoting and/or server-side Flash generation engine directly into the BlueDragon product, in a future release.

### 5.2.5 Ajax Integration

It's entirely possible for Ajax-based client applications to call upon CFML pages (or CFCs) to render results for display. Again, the third party product WebOrb (discussed in the previous section) has specific direct support for calling CFCs on BlueDragon using Ajax (and without using web services). See <http://www.themidnightcoders.com/> for more information.

Following are some tips which will help you as you develop CFML pages to serve to Ajax clients. First, be aware that you'll generally want to disable debugging, since most Ajax clients expect specific data, not the random HTML shown in debugging output. You can disable debugging with `<CFSETTING SHOWDEBUGOUTPUT="no">`.

Indeed, if the page expects XML (and not HTML), you need to set the content-type for the page, using `<CFCONTENT TYPE="text/xml">`.

Finally, for similar reasons, note that if the CFML page being requested by the Ajax client fails, it will generally return the standard HTML-based error page that BlueDragon created. If it's expecting XML or only some specific datatype, that could make the client fail to respond properly (or even reflect the error). Be sure to use error handling (such as via the `CFERROR` tag) to better control how errors are reported to Ajax clients.

### 5.2.6 CFML Editors Supported

BlueDragon CFML pages can be edited using any of the many editors available, including Macromedia's tools such as ColdFusion Studio, HomeSite, HomeSite+, and Dreamweaver MX. It can also be used with the CFEclipse tool. Another alternative is PrimalCode, from Sapien Technologies (<http://www.sapien.com>).

An advantage of both Dreamweaver MX and PrimalCode is that they support not only CFML but also ASP.NET, including specific support for building ASP.NET pages and adding ASP.NET controls to an aspx page, which can be helpful for users of BlueDragon.NET. (The other Macromedia editors can also edit ASP.NET pages, and include some color coding, but they don't offer the same level of support for adding ASP.NET controls to an aspx page.)

With regard to ASP.NET integration (discussed further in the manual, *Integrating CFML with ASP.NET and the Microsoft .NET Framework*), note that it's also possible to edit CFML pages in traditional ASP.NET tools, such as Visual Studio and WebMatrix. These tools, however, do not have specific CFML tag support and syntax highlighting.

### 5.2.7 RDS (For ColdFusion Studio, HomeSite+, and Dreamweaver)

BlueDragon does not include support for RDS, a proprietary Macromedia technology for communication with the ColdFusion server environment from Macromedia development environments including ColdFusion Studio, Dreamweaver MX, and HomeSite+. While you can configure those tools to use ColdFusion to obtain information about datasources and file systems on the ColdFusion server, you cannot configure them to obtain that sort of information from BlueDragon.

It may be worth noting, however, that if you have both ColdFusion and BlueDragon installed on a single machine, you can continue to use the connection via ColdFusion to obtain that information even while editing files for use with BlueDragon. (And you can obtain a single-IP developer edition of ColdFusion MX for free, which should suffice for the needs of using it as a connection for RDS purposes, even if deployed on a production box, and even if configured to use only the built-in web server. CFMX's developer

edition supports a request from both the localhost and another single IP address, which can change with a restart of CFMX.)

Additionally, there are third-party products that can provide many of the same features usually enabled via RDS connectivity in CF Studio, Dreamweaver, and HomeSite+.

One example is WinSQL, from Synametrics Corporation (whose Lite edition is free). It's available from:

<http://www.synametrics.com/SynametricsWebApp/WinSQL.jsp>

Another is Aqua Data Studio, from Aquafold Corporation (who license it freely personal or educational use, and offer their older edition free even for commercial or government use). It's available from:

<http://www.aquafold.com/>

Each offers features similar to the database features made available via RDS, as database query and administration tools that allows developers to easily create, edit, and execute SQL scripts, as well as browse and modify database structures. They also offer additional features over those familiar with using the RDS-based database features in Macromedia's development environments.

Further, users of specific databases may find that the vendors or third parties offer tools that satisfy the needs for graphical interfaces to managing and querying the databases.

### 5.2.8 Page Buffering Behavior

Like ColdFusion, BlueDragon now automatically buffers the complete generation of a page's output and doesn't send it to the browser until page completion (unless you use `CFFLUSH` to change that behavior).

BlueDragon previously followed the approach used by JSPs and servlets in the J2EE model, buffering only a certain amount of generated content before flushing the buffers. This had benefits for performance and scalability (discussed later in this section), but it introduced compatibility issues, so BlueDragon now defaults to buffering the full page output like ColdFusion.

You can alter the page buffering behavior for performance reasons, either on a template basis (using `CFFLUSH`, as in ColdFusion) or server-wide via an option in the BlueDragon administration console (in the section `Application>settings`, and its option `Default Response Buffer Size`. The default is now set to `Buffer Entire Page`, and this can be unchecked and an alternative setting in Kbytes specified to indicate that pages should be flushed automatically after generating that much output. (If you choose other than the `Buffer Entire Page`, BlueDragon offers a new option on `CFFLUSH`, with the attribute `Interval="Page"` to change the behavior on a single page to revert to the behavior of buffering the entire page, as discussed in the *BlueDragon 6.2.1 CFML Enhancements Guide*.)

Why might you want to change the behavior to not buffer entire pages (either on a page-by-page basis or server-wide)? First, by not waiting for the entire page to be processed, the page will likely be presented to the user sooner. Second, when a very large number of concurrently requested pages are being rendered in the server's memory, much more total memory will be used when buffering the complete generation of all pages to their completion.

Both BlueDragon and ColdFusion offer the `CFFLUSH` tag to flush a page at a particular point or when a specified `Interval` is reached (BlueDragon uniquely offers the `Page` value for that attribute because it offers the option to change the buffer size server-wide as discussed above.)

There is a negative impact of flushing the buffer before page completion, as is discussed in the ColdFusion documentation about `CFFLUSH`. The negatives apply as well to choosing BlueDragon's server-wide setting to not buffer the entire page. Any tags that write to the page headers (such as `CFCOOKIE`, `CFHEADER`, `CFCONTENT`, and `CFLOCATION`) cannot be executed after a page flush.

Consider this impact carefully before changing the server-wide setting to do other than complete page buffering.

### 5.2.9 FuseBox, Mach II, and Model-Glue Support

BlueDragon supports the Fusebox, Mach II, and Model-Glue frameworks.

Due to a bug in BlueDragon, in some builds of Fusebox, you must change Line 255 of `fusebox40.parser.cfm` from this:

```
fb_.parsedfile = fb_.parsedfile & repeatString(fb_.indentBlock,
min(fb_.maxIndentLevel, fb_.indentLevel)) &
'<CFSCRIPT>getPageContext().forward("' &
fb_.fuseQ[fb_.i].xmlAttributes.url & "');" </CFSCRIPT>';
```

to this:

```
fb_.parsedfile = fb_.parsedfile & repeatString(fb_.indentBlock,
min(fb_.maxIndentLevel, fb_.indentLevel)) & '<CFSET
getPageContext().forward("' & fb_.fuseQ[fb_.i].xmlAttributes.url &
'">';
```

Note that the `CFSCRIPT` tag was replaced with `CFSET`. This workaround is compatible with CFMX 6.1. Future implementations of FuseBox have been expected to implement this workaround in the corefile as downloaded.



## 6. CFML/Java/J2EE Integration

**B**lueDragon enables easy and powerful integration with native J2EE components, including accessing the methods and properties of Java classes, JavaBeans, EJBs, and more.

In addition, CFML pages can integrate with JSPs and servlets, including sharing of session, application, and request variables, as well as including/forwarding between them. While ColdFusion MX requires the Enterprise edition for similar features, these capabilities are possible with BlueDragon Server JX and BlueDragon/J2EE. BlueDragon makes this sort of integration a compelling addition to the other benefits it offers to CFML shops.

You can learn more about CFML/J2EE integration, its benefits and the various means of doing so, in section 6.4 below.

### 6.1 Integrating CFML with Java and J2EE

There are several ways to integrate with Java:

- call a Java class (custom or built-in object) with `CFOBJECT/createObject`
- call a java class as a CFX, assuming you have the Java source code (this allows the Java to be extended to read/write certain CFML variable scopes as exposed by the CFX interface)
- call a java servlet from CFML with `CFServlet`
- integrate CFML and JSP/servlets (sharing session/application/request scopes)
- call an EJB with `CFOBJECT/CreateObject`
- and more

That first point, about calling a java class, applies to both classes you might write (or someone in your organization may write) and classes available in the built-in java (and J2EE) libraries. Once you instantiate such a class (with `CFOBJECT/CreateObject`), you

can then access the class' properties and methods within CFML. A simple example is this:

```
<CFSCRIPT>
function GetHostAddress(host) {
    iaddrClass=CreateObject("java", "java.net.InetAddress");
    address=iaddrClass.getByName(host);
    return address.getHostAddress();
}
</CFSCRIPT>

<CFOUTPUT>#gethostaddress("www.newatlanta.com")#</CFOUTPUT>
```

## 6.2 Placement of Java Classes

If you will be trying to call custom java classes (ones you write, or from third parties, that are not built-in to Java/J2EE), you need to place the class file (or jar, or java archive, file) in a location recognized as in the classpath for BlueDragon.

In the case of BlueDragon Server JX, you can place such files in any location. After that, you must add the location (full path and file name) to the `Classpath` field offered in the BlueDragon Admin console's `BlueDragon Server>java vm settings` page.

The BlueDragon Server FREE edition does not provide the `BlueDragon Server>java vm settings` page in the Admin console. It is still possible to call built-in java classes, but it's not possible to declare a location for custom java classes.

In the case of BlueDragon/J2EE, you can place such files either the web application's `classes` directory (in the case of a Java class) or `lib` directory (in the case of a jar file,) or in any available shared directory for that J2EE server. There is no need (and no option) to extend the classpath in the BlueDragon/J2EE Admin console.

## 6.3 Session and Application Scope Sharing

In order to share session variables with JSP pages, the "J2EE Sessions" option of the BlueDragon administration console must be selected. (Application and request scopes are always shared).

With regard to sharing variables, it's important to distinguish between CFML applications with (and without) a `NAME` declared on `CFAPPLICATION`.

For named `CFAPPLICATIONS`, the application and session variables are stored in HashMaps (from the JSP perspective) that are retrieved using the application name, just like CFMX. For unnamed `CFAPPLICATIONS`, the application and session variables are stored directly in the underlying J2EE application and session scopes, just like CFMX.

For both named and unnamed `CFAPPLICATIONS`, the underlying J2EE session timeout is set to the value of the `SESSIONTIMEOUT` attribute of the `CFAPPLICATION` tag. This means

the timeout for JSP-created session variables will be controlled by the `CFAPPLICATION` tag.

For named `CFAPPLICATIONS`, the application scope variables will timeout based on the value of the `APPLICATIONTIMEOUT` of the `CFAPPLICATION` tag. For unnamed `CFAPPLICATIONS`, the application scope variables will never timeout.

## 6.4 Learning More about CFML/J2EE Integration

You can learn more about CFML/J2EE integration from several resources. Here are several chapters from books, entire books, articles, and article series that focus on the topic:

- Chapter 33 of the CFMX 6.1 manual, "Developing ColdFusion MX Applications". The chapter is titled, "Integrating J2EE and Java Elements in CFML Applications" and is available online at [livedocs.macromedia.com](http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/java.htm), specifically <http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/java.htm>
- The book, "Reality ColdFusion MX J2EE Integration", by Ben Forta et al.
- The book, "Java for ColdFusion Developers", by Eben Hewitt
- The chapter, "28: Integrating ColdFusion MX and Java", in the book, "ColdFusion MX Bible" from Wiley (by Churvis, Churvis, Helms, and Arehart)
- The CFDJ article, "Getting Started Integrating CFML with Java & .NET", at <http://cfdj.sys-con.com/read/86127.htm>
- The Macromedia DevNet article, "Using Java and J2EE Elements in ColdFusion MX Applications" at <http://www.macromedia.com/devnet/mx/coldfusion/articles/java.html>
- Another MM Devnet article, "ColdFusion MX and J2EE Hybrid Applications—A Case Study", at <http://www.macromedia.com/devnet/mx/coldfusion/j2ee/articles/hybrid.html>
- A set of other DevNet articles on CF/J2EE integration at [http://www.macromedia.com/devnet/mx/coldfusion/java\\_j2ee.html](http://www.macromedia.com/devnet/mx/coldfusion/java_j2ee.html)
- A CFDJ (ColdFusion Developer Journal) article, "ColdFusion MX/J2EE Hybrid Applications" at <http://cfdj.sys-con.com/read/42059.htm>
- An 8-part series of CFDJ articles on CF/Java integration by Guy Rish, at <http://cfdj.sys-con.com/author/113rish.htm> (from the CF 4.5 and 5 timeframe, but most concepts still apply)

## 6.5 Java and the Microsoft .NET Framework

Integration of Java classes in BlueDragon for the Microsoft .NET Framework requires an explanation where there may be some differences from the topics discussed in this chapter. Most important, the .NET framework supports J#, as an alternative to Java. If you can recompile your Java classes using Visual J#, then you should be able to call them with `CFOBJECT/createObject`. For more information on using CFXs in BlueDragon for the Microsoft .NET Framework, see the manual *Deploying CFML on the Microsoft .NET Framework*.

## 7. BlueDragon Server Free Edition

The following advanced features are available in the BlueDragon Server and BlueDragon/J2EE editions. These advanced features are **not** supported by the BlueDragon Server FREE edition:

- Java Servlets and JavaServer Pages (JSP)
  - CFINCLUDE of servlets/JSPs from CFML templates
  - CFFORWARD to servlets/JSPs from CFML templates
  - CFSERVLET tag
  - J2EE sessions
- Ability to define and extend the VM ClassPath for using Java classes
- Built-in JDBC drivers for Microsoft SQL Server and Oracle
  - Support for configuring additional drivers
  - (BlueDragon Server FREE edition supports ODBC on Windows, and MySQL and PostGreSQL only on Linux and OS X)
- Ability to create and deploy “Precompiled, Encrypted” templates
- Ability to process CFML pages in an SSL (HTTPS) request
- Web server adapters for iPlanet and SunONE web servers

To clarify, for the free Server edition, the only external web server supported on Windows is IIS, and the only one supported on Linux is Apache. This is in addition to the built-in web server supported on both Server and Server JX.

In addition to providing these advanced features, BlueDragon Server JX and BlueDragon/J2EE (and BlueDragon.NET) may be used to provide commercial web hosting services, may be redistributed to your customers, and may be deployed

commercially with the purchase of an appropriate license from New Atlanta. For additional information, please contact us via email: [sales@newatlanta.com](mailto:sales@newatlanta.com).