BlueDragon, BEA WebLogic® Edition 6.2.1
CFML Compatibility Guide

BlueDragon, BEA WebLogic® Edition

6.2.1
CFML Compatibility Guide

Published April, 2006

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

BlueDragon, BEA WebLogic® Edition 6.2.1
CFML Compatibility Guide

1 Introduction

BlueDragon is family of server-based products for deploying dynamic web applications developed using the ColdFusion® Markup Language (CFML). 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, WebLogic Edition 6.2.1 CFML Compatibility Guide* contains information about the compatibility of the CFML implementation in BlueDragon with the Macromedia ColdFusion 5.0 (CF5) and ColdFusion MX 6.1 (CFMX) implementations. Except where explicitly noted, all references to "ColdFusion" in this document refer to both CF5 and CFMX.

Developers currently working with ColdFusion 4.5 or earlier should be aware that there are differences among those older releases and CF5 and CFMX that are not generally documented in this manual. Refer to the Macromedia documentation for information on those differences.

BlueDragon enhancements to CFML that are not supported by ColdFusion are described in the *BlueDragon, WebLogic Edition 6.2.1 CFML Enhancements Guide.*

1.2 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, as listed in section 1.4, below. Except where explicitly noted, all references to "BlueDragon" in this document refer to all product configurations.

1.3 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

In the self-help section, you'll find documentation, FAQs, a feature request form, and a supportive mailing list staffed by both customers and New Atlanta engineers.

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

http://www.newatlanta.com/c/support/bluedragon/home

```
For BEA Systems technical support, go to:
```

http://support.bea.com

1.4 Additional Documentation

The other manuals available in the BlueDragon documentation library are:

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

2 ColdFusion Compatibility

2.1 ColdFusion Compatibility

The BlueDragon implementation of CFML is designed to be compatible with Macromedia CFMX 6.1. This document describes differences in syntax and semantics between the two implementations that are of interest to developers when upgrading existing CFML applications to BlueDragon from CF5 or CFMX.

This document is not a complete reference to CFML; for in-depth coverage of CFML, the following books are recommended:

ColdFusion MX Bible
by Adam Churvis, Hal Helms, and Charlie Arehart
http://www.wiley.com/WileyCDA/WileyTitle/productCd-0764546228.html

Programming ColdFusion MX
by Rob Brooks-Bilson
http://www.oreilly.com/catalog/coldfusion2/

2.2 Upgrading to BlueDragon from ColdFusion

This guide is intended for developers upgrading from either CF5 or CFMX; therefore, not all of the compatibility issues discussed here will apply to both audiences.

2.2.1 Upgrading from ColdFusion 5 and Earlier

If you're currently operating at CF5 or earlier, then differences in features introduced in CFMX won't apply to you, such as the issues discussed under CFARGUMENT, CFCOMPONENT, CFFUNCTION, CFIMPORT, and so on. Even some issues with tags existing in CF5 refer to changes made in CFMX, such as in CFCACHE and CFMAIL.

2.2.2 Upgrading from ColdFusion MX

If you're currently operating at ColdFusion MX 6 or 6.1, then differences in features that were supported in CF5 but are no longer supported in CFMX also will not apply. For instance, CFMX no longer supports CFAUTHENTICATE and CFIMPERSONATE. If you're operating at CFMX, you can't be using those tags so the incompatibility will not apply. Several other functions and tags are also deprecated in CFMX. Similarly, where we point out that the Java/J2EE editions of BlueDragon do not support "DSN-less connections" in CFINSERT, CFUPDATE, CFQUERY, etc., note that this feature is also not supported in CFMX.

2.2.3 Other Information on Upgrading from ColdFusion

While this document covers issues of CFML language compatibility, additional information about upgrading to BlueDragon from ColdFusion is provided in the *BlueDragon 6.2.1 User Guide*, in the Section "Upgrading from ColdFusion". Developers are strongly encouraged to review that material, and indeed all the information presented in the *BlueDragon 6.2.1 User Guide*.

2.3 Resolving CFML Compatibility Errors

BlueDragon offers several debugging tools and resources to aid your development and testing efforts, including enhancements over similar features in ColdFusion. While the *BlueDragon 6.2.1 CFML Enhancements Guide* offers detailed discussion of enhancements, the following are related specifically to resolving problems.

2.3.1 Debugging Output

BlueDragon offers optional debugging output at the end of each page (as offered in ColdFusion), which can be controlled in the Administration console and via the CFSETTING tag's SHOWDEBUGOUTPUT attribute. BlueDragon also supports tags such as CFDUMP, CFLOG, and CFTRACE.

One enhancement in the output of variable scopes (whether in debugging output, error page output, or CFDUMP) is that BlueDragon expands the values of arrays, structures, and other nested objects, providing more information to assist in debugging. BlueDragon also displays the variables scope by default in error pages.

2.3.2 Error Handling and Logging

When an error occurs in BlueDragon, the error can be handled as in ColdFusion with CFERROR, CFTRY/CFCATCH, try/catch (within CFSCRIPT), and the global error handler.

If an error is not handled, BlueDragon returns an error page to the browser. As an enhancement, BlueDragon also logs that entire error page to a log file as an HTML page (including the lines in error, all the variable scopes, etc.). For more information on the run-time error log (and other logs), including its location in the file system, see the technote:

> *How is logging in BlueDragon different than in ColdFusion?*
> http://www.newatlanta.com/products/bluedragon/self_help/tech_notes/logging_diff.cfm

As a further enhancement, as of BlueDragon 6.2 you can now optionally indicate in the Administration console that this error log file be created *even if errors are handled by your custom error templates*. Further, the name and location of that logfile is also now available as a new variable in the CFCATCH, CFERROR, and ERROR scopes, named ERRORLOGFILE, as described in the *BlueDragon 6.2.1 CFML Enhancements Guide.*

2.3.3 CFDEBUGGER Tag

As an additional enhancement, BlueDragon offers a special CFDEBUGGER tag that can log a trace of every line of CFML processed, writing this information to a logfile, which can be invaluable when resolving certain errors. See the *BlueDragon 6.2.1 CFML Enhancements Guide* for more information.

2.3.4 Seeking Additional Assistance

There are several resources available to help you solve problems while working with BlueDragon. See section 1.3, above for more information.

2.3.5 Recreating Problem Scenarios

When trying to resolve a seeming compatibility issue, it's helpful to try to reduce the code to a small, reproducible case. This helps you narrow the problem scope and sometimes may help you find and resolve the problem on your own. Should you end up passing it to the BlueDragon-Interest mailing list or New Atlanta engineers, it will aid others as well.

In particular, a useful tip when trying to debug problems within large `CFSCRIPT` blocks is to separate the block into smaller chunks by closing and re-opening the `CFSCRIPT` tag (where appropriate, as you cannot interrupt a block of code within statements such as `If`, `Switch`, `Loop`, or a `function` declaration.)

2.3.6 Updated Compatibility Information

While this guide documents all known compatibility issues at the time of its publication, additional items may arise before the publication of the next revision of this document, which will coincide with the release of the next new edition of BlueDragon.

Any important compatibility issues that arise in the interim will be documented in a technote offered in the self-help area of our web site:

http://www.newatlanta.com/products/bluedragon/self_help/technotes.cfm

2.3.7 Additional Compatibility Information in Bug Tracking System

There may be additional compatibility issues (including work-arounds) reported in the online bug tracking system:

http://www.newatlanta.com/c/support/bluedragon/bugtracking/home

3   Variables

3.1 CFML Keywords/Reserved Words

It is strongly recommend that you avoid using CFML reserved keywords as variable names. The following are recognized as reserved words in BlueDragon:

AND	CONTAINS	EQ	EQUAL	EQUALS
EQV	GT	GE	GTE	IMP
IS	LE	LT	LTE	MOD
NEQ	NOT	OR	XOR	

While this practice is not recommended, it is permissible to use these reserved words as variable names in BlueDragon if you prefix them with a variable scope name; for example, the following are permitted:

```
<CFSET variables.contains = "foo">
<CFSET session.contains = "foo">
```

Further, the following are recognized as reserved words within a CFSCRIPT block by BlueDragon:

BREAK	CASE	CATCH	CONTINUE	DEFAULT
DO	ELSE	FOR	FUNCTION	IF
IN	RETURN	SWITCH	TRY	VAR
WHILE				

Within the CFSCRIPT block, these can be referenced with a variable scope name as well.

The following can be used as variable names even without a scope prefix: DOES, CONTAIN, GREATER, THAN, LESS, IS, EQ, and EQUALS.

Finally, note that variable scope names (such as url, form, session, etc.) can be used as variable names, as long as they are prefixed with a scope name (such as variables.url). Note, however, that when referencing them later, you must use the prefix, otherwise the reference is presumed to refer to the structure created for the scope name (as in <CFDUMP var="#url#">.) Again, this practice is not recommended and you should avoid using scope names as variable names.

3.2 SERVER Variables

For BlueDragon, the variable `Server.ColdFusion.ProductName` returns the value "BlueDragon" and `Server.ColdFusion.ProductLevel` returns a value indicating the BlueDragon product installed (such as "Server JX"). For BlueDragon 6.2.1, the `ProductVersion` variable returns the value "6,2,1,xxx" where "xxx" is the internal build number; for example: 6,2,1,285.

See the *BlueDragon 6.2.1 CFML Enhancements Guide* for information on new server variables available in BlueDragon.

Finally, BlueDragon does not support the `Server.Coldfusion.Rootdir` variable.

3.3 Client and Cookie Variables after Flush

BlueDragon will display an error when you perform a certain unpermitted action that ColdFusion will simply ignore. As in ColdFusion, cookie variables (or client variables stored in cookies) cannot be created or edited once the page output has been flushed, whether by using `CFFLUSH` or when the size of the output generated has exceeded that specified in the BlueDragon Administration console for the response buffer size (if any).

If you do try to set a cookie (or a client variable stored in a cookie) after the page has been flushed, ColdFusion gives no error but does not send the cookie to the client (or set the value of the client variable); that is, it "fails silently." BlueDragon, on the other hand, will throw an exception in this case (since, as with ColdFusion, you cannot modify the HTTP headers to set cookies after performing a `CFFLUSH`).

For more on page flush (buffering) behavior in BlueDragon, see Section 6.6, below, and the *BlueDragon 6.2.1 CFML Enhancements Guide*.

3.4 Client Variable Processing

There are some differences between BlueDragon and ColdFusion in the underlying implementation of client variable processing with regard to using a database to store the client variables. These differences do not affect your CFML in any way, but may be of interest to server administrators.

First, where ColdFusion requires an administrator to configure a datasource to serve as a client variable repository, BlueDragon does not. If a datasource is specified in the `CFAPPLICATION ClientStorage` attribute (or in the administration console when setting a default client storage datasource, if desired), BlueDragon will store client variables there without need to prepare the datasource to hold client variables.

Note as well that BlueDragon will not create the tables needed within the datasource until a page is processed that references client variables.

Finally, ColdFusion stores client variables in database tables named `CDATA` and `CGLOBAL`. For the Java/J2EE editions of BlueDragon, the client database tables are named `BDDATA`

and `BDGLOBAL`; for the .NET edition of BlueDragon the client database tables are named `BDNETDATA` and `BDNETGLOBAL`.

Releases of BlueDragon older than 6.2 did not store client variables on a per-application basis; starting with BlueDragon 6.2, client variables are stored per application. The data table has a composite key of `CFID` (the `CFID:CFTOKEN` combination) and `APP`, which contains the application name. The individual data strings stored as client variables are held in a single column named `CFDATA` whose value is encoded and cannot be read except by referencing the data through the `client` variable scope in CFML.

If you had a release of BlueDragon prior to 6.2 installed and had created client variables in a given datasource using that older release, BlueDragon would have written client variables to a single table (`BDCLIENTDATA`). If present, that table will be used by BlueDragon 6.2.1 for backwards compatibility. To gain the new application-specific client variable functionality, you must either delete the `BDCLIENTDATA` table (losing the existing client variables stored there) or simply choose a new datasource to hold client variables (which also will effectively lose connection to the older client variables.)

3.5 CGI.Path_Info Variable

In some cases, ColdFusion incorrectly renders the `CGI.PATH_INFO` variable to have the same value as the `CGI.SCRIPT_NAME` variable. BlueDragon instead only assigns the value that is consistent with the "extra path info" as defined in the HTTP specification.

The "extra path info" is the information that follows the filename in a URL when separated by a "/" (as opposed to query string info, which is what follows a "?"). Whether the "extra path info" is available depends on the web server being used, so results may vary. For example, consider a request with the following form:

```
http://myserver/mypage.cfm/test/1/2
```

In BlueDragon, the value of path_info would be rendered (in most web servers) as "/`test/1/2`", which is useful for some application programming techniques. In ColdFusion, the value would instead be "/`mypage.cfm/test/1/2`". Many developers rely on this variable to get the template name (`mypage.cfm`) and then resort to extra work to strip out the template name.

Do not use the `CGI.PATH_INFO` variable where the file name of the script being processed is intended; instead, use `CGI.SCRIPT_NAME`, which is compatible with ColdFusion.

For more information on the HTTP definition of the CGI PATH_INFO variable, see:

http://www.w3.org/Daemon/User/CGI/Overview.html#Input

http://hoohoo.ncsa.uiuc.edu/cgi/env.html

4 CFML Tags

4.1 Unsupported Tags

The following CFML tags are not supported, and will generate run-time errors when rendered by BlueDragon:

Unsupported Tags		
Client-side Java	Extensibility	Web Applications
CFAPPLET	CFREPORT	CFAUTHENTICATE[*]
CFGRID	CFCHART	CFIMPERSONATE[*]
CFGRIDCOLUMN	CFCHARTDATA	
CFGRIDROW	CFCHARTSERIES	
CFGRIDUPDATE	CFGRAPH[†]	
	CFGRAPHDATA[†]	

[*] obsolete in CFMX
[†] deprecated in CFMX and BlueDragon

4.2 Supported with Differences

The following CFML tags are supported by BlueDragon with differences relative to the CF5 and CFMX implementations as noted; tags are listed alphabetically.

4.2.1 CFCACHE

BlueDragon does not support the TIMEOUT attribute, which was deprecated in CFMX; using this attribute in BlueDragon will generate an error. BlueDragon does support the TIMESPAN attribute which was introduced in CFMX. Also, the CACHEDIRECTORY attribute was deprecated in CFMX and is simply ignored by BlueDragon.

4.2.2 CFCOLLECTION

BlueDragon does not support the LANGUAGE attribute. The default is always English. Additionally, BlueDragon does not support the MAP, OPTIMIZE, or REPAIR values for the ACTION attribute, as these are not required due to differences in the underlying search engine technology.

See additional compatibility information under CFINDEX and CFSEARCH.

4.2.3 CFCOMPONENT

BlueDragon requires that a component file (.cfc) contain opening and closing CFCOMPONENT tags; CFMX does not require use of the CFCOMPONENT tag within component files.

4.2.4 CFCONTENT

BlueDragon does not support using a pair of CFCONTENT tags, as in: `<CFCONTENT…>`*some data*`</CFCONTENT>`. In BlueDragon, only the opening CFCONTENT tag is supported.

4.2.5 CFDIRECTORY

For the .NET edition of BlueDragon, whenever a directory within an ASP.NET web application is renamed the ASP.NET web application is restarted; this in turn causes BlueDragon.NET to be restarted. Therefore, renaming a directory within an ASP.NET web application using CFDIRECTORY will cause BlueDragon to restart.

4.2.6 CFDUMP

BlueDragon currently ignores the EXPAND attribute available in ColdFusion. See the available VERSION attribute which has been added to CFDUMP in BlueDragon, as described in the *BlueDragon 6.2.1 CFML Enhancements Guide.*

4.2.7 CFFLUSH

The INTERVAL attribute behaves differently on BlueDragon than on ColdFusion; see Section 6.6 of this document, below, for a detailed discussion.

4.2.8 CFGRAPH

CFGRAPH, which was supported in BlueDragon 3.0 and 6.1, has been deprecated and is no longer supported as of BlueDragon 6.2. Further, in the .NET edition of BlueDragon the tag is obsoleted (does not function at all).

4.2.9 CFIMPORT

BlueDragon does not support using CFIMPORT to execute JSP custom tag libraries. It does support using CFIMPORT as an alternative approach for executing CFML custom tags.

4.2.10 CFINDEX

In BlueDragon, the search capability (CFSEARCH, CFINDEX, and CFCOLLECTION tags) is based on the Jakarta Lucene project open source search engine, which results in differences between the BlueDragon and ColdFusion implementations. For additional information, see the *BlueDragon 6.2.1 CFML Enhancements Guide*.

BlueDragon does not support the LANGUAGE attribute; the default language is always English. Also, BlueDragon is currently limited to searching HTML- and text-based documents.

See additional compatibility information under CFSEARCH and CFCOLLECTION.

4.2.11 CFINSERT

Like CFMX, the Java/J2EE editions of BlueDragon do not support the following attributes, which were used in CF5 to enable "DSN-less" connections:

```
connectString
dbType="dynamic"
```

The .NET edition of BlueDragon, however, does support these attributes and the DSN-less connections feature. See the discussion of CFQUERY tag enhancements in the *BlueDragon 6.2.1 CFML Enhancements Guide.*

Additionally, again like CFMX, BlueDragon does not support the following attributes, which were used in CF5 for Native Driver and OLEDB connections:

```
dbName                    provider
dbServer                  providerDSN
```

4.2.12 CFLOG

When used without a FILE or LOG attribute, CFLOG defaults to "application".

The location of output for CFLOG depends on the edition used. In BlueDragon Server and Server JX, logs are written to the work\cflog directory of the BlueDragon installation directory.

For BlueDragon/J2EE, logs are written to the WEB-INF\bluedragon\work\cflog directory of the J2EE web application. For information on the work directory in the .NET edition, see *Deploying CFML on ASP.NET and the Microsoft .NET Framework.*

4.2.13 CFLOOP

BlueDragon only allows integer values for the TO and FROM attributes. While CFMX supports non-integer values for these attributes, including floating point numbers and dates, the CFMX documentation explicity warns against the use of non-integer values:

> "Using anything other than integer values in the from and to attributes of an index loop can product unexpected results."

http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/tags-p77.htm#wp1101087

4.2.14 CFMAIL

BlueDragon does not support the SPOOLENABLE attribute, introduced in CFMX.

4.2.15 CFOBJECT

BlueDragon supports "java", "component", and "webservice" values for the type attribute; it does not support the values "com" or "corba".

For the .NET edition, an additional type of ".net" has been added, which should be used when referring to .NET objects and managed assemblies (though "java" is supported for backward compatibility and is synonymous). See *Integrating CFML with ASP.NET and the Microsoft .NET Framework* for more information.

For additional information on COM object integration, see section 6.10 of this document.

4.2.16 CFPROCESSINGDIRECTIVE

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.

In BlueDragon, the setting of the `CFPROCESSINGDIRECTIVE SUPPRESSWHITESPACE` attribute is not applied to custom tag calls; instead, the default for whitespace compression configured in the BlueDragon admin console is applied during custom tag calls. The setting of the `CFPROCESSINGDIRECTIVE SUPPRESSWHITESPACE` attribute does remain in effect for templates included via `CFINCLUDE`, for user-defined functions (UDFs), and for CFC function calls.

See the discussion in Section 6.1 of this document, below, for additional details on white space compression.

4.2.17 CFPROCPARAM

The `CFPROCPARAM` tag is used to pass parameters to a stored procedure. When nested within `CFSTOREDPROC`, the tags can pass the parameters either positionally, or by name by using the `DBVARNAME` attribute.

If used with an ODBC datasource, the `DBVARNAME` attribute is ignored in CF5; `DBVARNAME` is always ignored in CFMX and the Java/J2EE editions of BlueDragon. In these cases, parameters must be passed positionally in the order defined by the `CFPROCPARAM` tags, which must match the order defined within the stored procedure.

The .NET edition of BlueDragon, however, supports the `DBVARNAME` attribute when using the Microsoft SQL Server or Oracle database drivers; therefore if the `DBVARNAME` attribute is specified its value must match an actual parameter name defined in the stored procedure, otherwise you'll get a CFML runtime error. The error may arise in code that worked in CF5, CFMX, or the Java/J2EE editions of BlueDragon when the `DBVARNAME` was otherwise being ignored.

Finally, note that while both positional and named parameters are support by BlueDragon.NET, using named parameters (by specifying `DBVARNAME`) will result in better performance than positional ordering.

4.2.18 CFQUERY

4.2.18.1 Unsupported Tag Attributes

Like CFMX, the Java/J2EE editions of BlueDragon do not support the following attributes, which were used in CF5 to enable "DSN-less" connections:

```
connectString
dbType="dynamic"
```

The .NET edition of BlueDragon, however, does support these attributes and the DSN-less connections feature. See the discussion of `CFQUERY` tag enhancements in the *BlueDragon 6.2.1 CFML Enhancements Guide*.

Additionally, again like CFMX, BlueDragon does not support the following attributes, which were used in CF5 for Native Driver and OLEDB connections:

```
dbName                     provider
dbServer                   providerDSN
```

Finally, the following attributes were deprecated in CFMX and BlueDragon does not them:

```
blockfactor
timeout
```

4.2.18.2 Query of Query processing

While BlueDragon supports query of query processing (specified by dbType="query"). CFMX added support for new SQL language statements such as distinct and union. BlueDragon does not yet support all those statements. BlueDragon does support many of the added query of query SQL statements, including sum, avg, lower, upper, and others.

4.2.19 CFREGISTRY

BlueDragon simulates the Windows registry on all operating systems, including Windows, UNIX/Linux, and Mac OS X. Therefore, it is not possible to read, write, or delete "registry" entries other than those created using the CFREGISTRY tag. (Note that CFMX deprecated use of CFREGISTRY on UNIX/Linux).

In other words, on Windows systems, you cannot access the real Registry using the CFREGISTRY tag. For the Java/J2EE editions of BlueDragon, there are available third-party Java utilities that you can use in order to gain access to the real registry via CFOBJECT calls to the utility's methods. Possible candidates you may consider include:

http://www.trustice.com/java/jnireg/

http://sourceforge.net/projects/jregistrykey/

For BlueDragon.NET, the registry can be accessed via CFOBJECT calls to the Microsoft.Win32.Registry class (see the Microsoft .NET Framework documentation for additional information).

4.2.20 CFSEARCH

In BlueDragon, the search capability (CFSEARCH, CFINDEX, and CFCOLLECTION tags) is based on the Jakarta Lucene project open source search engine, which results in differences between the BlueDragon and ColdFusion implementations.

Text searches work similarly; while BlueDragon does not identically support all the ColdFusion search language keywords such as NEAR, STEM, WILDCARD, CONTAINS, and others, which you might specify with a CFSEARCH Type="explicit", BlueDragon does contain its own rich set of search language keywords. Many of them are the same or very similar to ColdFusion's keywords and operators. Note that while ColdFusion allows use

of a wildcard at the start of the `CRITERIA` search string, the Lucene engine and therefore BlueDragon do not.

Syntax and simple examples are offered in the brief but quite complete "Query Syntax" document available on the Lucene web site:

> http://jakarta.apache.org/lucene/docs/queryparsersyntax.html

As for the `CFSEARCH` tag itself, BlueDragon does not support the `LANGUAGE` attribute. The default is always `English`.

In BlueDragon, the `CUSTOM1`, `CUSTOM2`, and `URL` attributes are expected to be query columns. If you prefer to use a string instead, you will need to add a column in the SELECT statement of the query using the AS keyword and use that column reference instead.

See additional compatibility information under `CFINDEX` and `CFCOLLECTION`.

4.2.21 CFSETTING

The Java/J2EE editions of BlueDragon do not support the `REQUESTTIMEOUT` attribute which was introduced in CFMX. This attribute is supported in the .NET edition of BlueDragon; however, note that the error reported will be a .NET error page reporting "Request Timed Out" rather than a BlueDragon error page.

(Like CFMX, BlueDragon does not support use of a `REQUESTTIMEOUT` directive in the URL query string. Also, BlueDragon does not support an admin console setting for page timeouts.)

BlueDragon also does not support the `CATCHEXCEPTIONSBYPATTERN` attribute, which was made obsolete in CFMX.

4.2.22 CFSTOREDPROC

Like CFMX, the Java/J2EE editions of BlueDragon do not support the following attributes, which were used in CF5 to enable "DSN-less" connections:

```
connectString
dbType="dynamic"
```

The .NET edition of BlueDragon, however, does support these attributes and the DSN-less connections feature. See the discussion of `CFQUERY` tag enhancements in the *BlueDragon 6.2.1 CFML Enhancements Guide.*

Additionally, again like CFMX, BlueDragon does not support the following attributes, which were used in CF5 for Native Driver and OLEDB connections:

```
dbName                  provider
dbServer                providerDSN
```

4.2.22.1 Oracle Stored Procedures and Reference Cursors

There are some aspects of processing stored procedures on Oracle, especially with respect to reference cursors, that differ in BlueDragon compared to CF5 and CFMX (note that the method for calling Oracle stored procedure differs between CF5 and CFMX).

Oracle returns result sets from stored procedures as OUT parameters of type REF CURSOR. In BlueDragon, you simply use the CFPROCPARAM tag and specify the variable name to hold the result set. The following code running on BlueDragon returns the result set in a query variable named "myResults":

```
<cfstoredproc proc="myProc" datasource="dsn">
  <cfprocparam type="IN" value="#inValue#">
  <cfprocparam type="OUT"
              cfsqltype="CF_SQL_REFCURSOR"
              variable="myResults">
</cfstoredproc>
```

In CF5, you're required to specify a "dummy" OUT parameter and then use CFPROCRESULT to create a variable to hold the actual result set. The following code running on CF5 returns the result set in a query variable named "myResults":

```
<cfstoredproc proc="myProc" datasource="dsn">
    <cfprocparam type="IN" value="#inValue#">
    <cfprocparam type="OUT"
                cfsqltype="CF_SQL_REFCURSOR"
                variable="dummy">
    <cfprocresult name="myResults" resultset="1">
</cfstoredproc>
```

CFMX differs from CF5 in that you don't use the CFPROCPARAM tag to specify a dummy OUT variable, but only specify the CFPROCRESULT tag to access the results. The following code running on CFMX returns the result set in a query variable named "myResults":

```
<cfstoredproc proc="myProc" datasource="dsn">
    <cfprocparam type="IN" value="#inValue#">
    <cfprocresult name="myResults" resultset="1">
</cfstoredproc>
```

4.2.23 CFTEXTINPUT

ColdFusion allows the BGCOLOR and TEXTCOLOR attributes to be specified as hex values in the form "#nnnnnn", although the documentation indicates the pound sign should be escaped, as in "##nnnnnn". BlueDragon follows the documented approach and requires that the pound sign be escaped. As a work-around, if you used the undocumented single pound sign, changing it to use two will still be compatible with ColdFusion.

4.2.24 CFTREE

BlueDragon does not support the following optional CFTREE attributes:

```
completePath          onValidate
delimiter
```

4.2.25 CFTREEITEM

BlueDragon does not support the following optional CFTREEITEM attributes:

```
img
imgOpen
```

4.2.26 CFUPDATE

Like CFMX, the Java/J2EE editions of BlueDragon do not support the following attributes, which were used in CF5 to enable "DSN-less" connections:

```
connectString
dbType="dynamic"
```

The.NET edition of BlueDragon, however, does support these attributes and the DSN-less connections feature. See the discussion of CFQUERY tag enhancements in the *BlueDragon 6.2.1 CFML Enhancements Guide.*

Additionally, again like CFMX, BlueDragon does not support the following attributes (which were used in CF5 for Native Driver and OLEDB connections):

```
dbName              provider
dbServer            providerDSN
```

4.2.27 CFWDDX

The following limitations exist in the BlueDragon implementation of the CFWDDX tag relative to the ColdFusion implementation:

1. BlueDragon cannot deserialize binary data from WDDX to CFML. BlueDragon can serialize all other kinds of CFML data, including query resultsets, arrays, and structures to name a few.

2. The USETIMEZONEINFO attribute is not supported by BlueDragon (defaults to "Yes").

3. The VALIDATE attribute is not supported by BlueDragon.

5 CFML Functions

5.1 Unsupported Functions

The following CFML functions are not supported by BlueDragon:

Unsupported Functions		
AuthenticatedContext†	GetException	getK2ServerDocCount*
AuthenticatedUser†	GetMetricData	getK2ServerDocCountLimit*
IsAuthenticated†	LSParseDateTime	isK2ServerABroker*
IsAuthorized†	LSParseEuroCurrency	isK2ServerDocCountExceeded*
IsProtected†		isK2ServerOnLine*
		ReleaseCOMObject

*deprecated in CFMX
†obsolete in CFMX

5.2 Supported with Limitations

5.2.1 CreateObject

The first argument of `CreateObject()` describes the `type` of object being called. See the discussion of CFOBJECT for value `type` values supported by BlueDragon.

5.2.2 Date functions

In ColdFusion, several date handling function (such as `dateformat`, `dayofyear`, `month`, and more) will process even if their `year` argument is left off, defaulting to the current year. BlueDragon requires the `year` argument to be specified.

5.2.3 DateAdd

While ColdFusion allows `DateAdd` arguments as either `datepart/number/date` or `datepart/date/number`, supports only the first form. The following expression, `#DateAdd('d',now(),-27)#`, will return "`data not supported`" in BlueDragon.

5.2.4 DateFormat/TimeFormat

BlueDragon does not support `timeformat` masks in `DateFormat()` as completely as ColdFusion does. For instance, in BlueDragon, the `HH` mask returns 12-hour rather than 24-hour time. The value is returned correctly when used in `TimeFormat()`.

5.2.5 Decrypt/Encrypt

BlueDragon does not support decryption of text that was encrypted by ColdFusion pages using their implementation of this function (such as data stored in a file or database after encryption). Conversely, text encrypted on BlueDragon cannot be decrypted in ColdFusion; as a work-around, simply re-run the process to encrypt the text using BlueDragon.

5.2.6 GetMetaData

BlueDragon only support invoking GetMetaData() on CFCs. It does not support invoking GetMetaData() on Java objects, `CFFUNCTION`-based UDFs, or any variable other than a CFC instance.

6 Miscellaneous

There are various other aspects of working with ColdFusion and CFML that may be slightly different in BlueDragon, but don't fit neatly into a discussion of tags or functions.

6.1 WhiteSpace Compression

BlueDragon supports control of whitespace compression through both an Administration Console setting (`Application->settings->Whitespace Compression`) and the `SuppressWhiteSpace` attribute of the `CFPROCESSINGDIRECTIVE` tag. BlueDragon's compression of whitespace is more aggressive than ColdFusion, which only compress whitespace that exists within CFML tags and expressions.

First, BlueDragon compresses the entire content stream (as controlled by the Administration console setting or `CFPROCESSINGDIRECTIVE`). An implication of this is that if compression is enabled, you may want to limit compression within some block of HTML code that may be dependent on whitespace remaining as coded, such as with HTML tags like `SCRIPT`, `PRE`, or `TEXTAREA`. In such cases, you can use the `CFPROCESSINGDIRECTIVE` `SuppressWhiteSpace="no"` to prevent compression.

Also, when `CFPROCESSINGDIRECTIVE SuppressWhiteSpace="yes"` is used to enable compression, BlueDragon propagates that tag's settings into templates included using `CFINCLUDE`, for user-defined functions, and for CFC function calls as discussed in section 4.2.16. You may need to use the tag with its "`no`" option to disable compression in selected pages.

Fusebox applications in particular are susceptible to this issue, since the core files intentionally turn on whitespace suppression. In BlueDragon that behavior trickles down to all pages included from the core files, which in Fusebox is all files in the applications.

To alleviate these issues, as of BlueDragon 6.2 the compression tries to achieve a balance, which may reduce the need to worry about turning off the compression for some runs of HTML when it's enabled. If a run of whitespace contains a newline ("\n") character anywhere in the run, the run of whitespace is collapsed to a single newline character. If it does not contain a newline character, it's collapsed to be the first character in the run.

6.2 Response Buffering and CFFLUSH

The "`Response Buffer Size`" configuration setting on the "`Application->settings`" Administration console page sets the intial size of BlueDragon's response buffer. This is the number of bytes that will be buffered before the response is flushed to the client

browser (this setting has the default of "`Buffer Entire Page`"). After the response is flushed to the client the first time, all subsequent output is no longer buffered by BlueDragon, but is instead written directly to the response buffers of the underlying ASP.NET or J2EE server.

Note that after the response is flushed to the client, it is no longer possible to set response headers, set cookies, use the `CFHTMLHEAD` tag, or set Client scope variables if these are being stored in cookies.

Setting the "`Response Buffer Size`" to the smallest possible value can significantly increase the performance of your application, especially for larger pages, because it is much more efficient for BlueDragon to write output directly to the underlying ASP.NET or J2EE server response buffers, rather than to first buffer the response itself and then copy the output to the underlying server response buffers.

Similarly, the `INTERVAL` attribute of `CFFLUSH` only sets the initial size of BlueDragon's response buffer. After the response is flushed to the client the first time, all subsequent output is written directly to the underlying ASP.NET or J2EE server response buffers. In this sense, the `INTERVAL` attribute of `CFLLUSH` is essentially used to override the "`Response Buffer Size`" Administration console setting; note that BlueDragon allows you to specify `INTERVAL=`"`Page`" to buffer the entire page.

Finally, note that the `CFFLUSH` tag without the `INTERVAL` attribute can be used to flush output to the client at any time (that is, it works the same as in ColdFusion).

6.3 Search Process for Application.cfm

In both ColdFusion and BlueDragon, if an `Application.cfm` file is not located in the same directory as a page being requested, each ancestor directory (parent, grandparent, etc.) will be searched until an `Application.cfm` is found. In BlueDragon, the search will stop at the web server document root directory or J2EE web application root directory, whereas ColdFusion will search beyond that to the drive root.

6.4 ColdFusion Components as Web Services

There are a few differences in CFC (ColdFusion Component) processing in BlueDragon, with respect to their use as web services.

For web services interoperability with CFMX, web services that use CFCs as parameters/return types must define identical CFCs in both BlueDragon and CFMX. CFC names and properties are case sensitive.

Query result datatypes used in web services are only preserved as such when using BlueDragon as both web services client and server.

Complex object web services parameters will only be received as CFC instances if a CFC is defined that matches the complex object structure and is found in or with a directory or name that corresponds to the complex object namespace or name. Received complex

objects that do not have a corresponding CFC (e.g., that are unknown to BlueDragon) will be deserialized into an XML object.

Finally, web services method names in BlueDragon are case-sensitive.

6.5 CFLOG File Placement

When using `CFLOG` in BlueDragon Server and Server JX, logs are written to the `work\cflog` directory of the BlueDragon installation directory. For BlueDragon/J2EE, logs are written to the `WEB-INF\bluedragon\work\cflog\` directory of the J2EE web application. For information on the work directory in the .NET edition, see *Deploying CFML on ASP.NET and the Microsoft .NET Framework.*

6.6 Case Sensitivity in Java Method Calls

When accessing methods of Java objects (whether accessed by way of `CFOBJECT/CreateObject()` or when referring to variable scopes that may have Java objects within them), CFML (in both ColdFusion and BlueDragon) is case insensitive. BlueDragon does, however, try to do a case-sensitive match first (to see if the case you specified matches the case of an existing method in the Java object).

On the rare occasion where you need to call a method where there are two methods of the same name differing only by case then case becomes important. BlueDragon will attempt to resolve the ambiguity by performing a case sensitive match first. If that fails then an exception is thrown reporting that the ambiguity could not be resolved.

For example, consider attempting to use the `isRequestedSessionIdFromURL` in the Java servlet page context. There are actually two methods of that name in that object. One is spelled with `URL` in caps while the other uses "camel case", capitalizing only the first letter of each word after the first word, in this case with the last word spelled `Url`.

If you made a request for the method using either exact case of one of the methods, BlueDragon would execute it. But consider when the case of another part of the method name does not exactly match. For instance, notice that the I in `Id` is capitalized in both method names. Now, consider making a CFML request for that with the "`i`" in lower-case, as in:

```
<cfset rqObj = getPageContext().getRequest()>
<cfoutput>#rqObj.isRequestedSessionidFromURL()#</cfoutput>
```

Which one should BlueDragon select? It doesn't really matter which case the `URL` portion of the methodname is spelled. You could even request it with:

```
<cfoutput>#rqObj. isRequestedSessionidFromUrl()#</cfoutput>
```

The issue is that no method exists with the lowercase i. It wouldn't be an issue if there was only one method of the given name, but since there are two, varying only by case, BlueDragon can't know which method you mean. So BlueDragon will throw an error:

```
    Method isRequestedSessionidFromURL is ambiguous as there is
more than one method that could correspond to the provided argu-
ment types. If possible, use 'javacast()' to resolve this ambigu-
ity.
```

Despite the message's content, even use of `JavaCast()` won't help here. The message also refers to other situations of ambiguity (as when multiple methods exist accepting different arguments), and in those cases `JavaCast()` can help. In the case of two methods of the same name (and same arguments) with only case differentiating them, there's simply no way for BlueDragon to determine which to select when you don't use the exact case. (ColdFusion instead chooses one of the two, with no apparent logic as to which it chooses).

6.7 UDF Forward References

BlueDragon does not support "forward references" to user-defined functions. That is, a UDF must be defined before you can call it. For example, consider the following code, which works on ColdFusion:

```
<cfset sayHelloScript()>
<cfset sayHelloTag()>

<cfscript>
function sayHelloScript() {
    writeOutput( "<p>Hello from a CFSCRIPT-based UDF" );
}
</cfscript>

<cffunction name="sayHelloTag" output="true">
    <p>Hello from a CFFUNCTION-based UDF
</cffunction>
```

On BlueDragon the above code will generate "undefined function" errors. To resolve these errors, move the code that references the UDFs (the CFSET tags) to appear in the CFML page after the UDF declarations.

The exception to this is that BlueDragon does support forward references to component functions declared using CFFUNCTION (but not CFSCRIPT) within the CFC pseudo-constructor. So the following code works the same on both ColdFusion and BlueDragon:

```
<cfcomponent>
    <cfset init()>
    <cffunction name="init">
        ...component initialization code goes here...
    </cffunction>
</cfcomponent>
```

6.8 Dynamic UDF Declarations

ColdFusion does not dynamically evaluate code that declares user-defined functions (UDFs) via CFSCRIPT or CFFUNCTION, but BlueDragon does. Therefore, the following

will work in ColdFusion but fail in BlueDragon, because in BlueDragon the CFSCRIPT function declaration is not rendered:

```
<CFIF 1 EQ 0>
  <CFSCRIPT>
  function dohello() {
         return "hello there";
  }
  </CFSCRIPT>
</CFIF>

<CFOUTPUT>#dohello()#</CFOUTPUT>
```

6.9 Comments within CFML Tags

BlueDragon does not allow comments within CFML tags. For example, the following will result in a syntax error on BlueDragon, but will parse successfully on ColdFusion:

```
<cfset firstName="Tom" <!--- customer's first name --->>
```

Instead, put the comment outside of the CFML tag for compatibility with both BlueDragon and ColdFusion:

```
<cfset firstName="Tom"> <!--- customer's first name --->
```

6.10 COM Object Integration

As discussed previously, BlueDragon does not support COM object integration via CFOBJECT TYPE="com" or CreateObject("com", comobject). It is possible, however, to perform COM object integration on both the Java/J2EE and .NET versions of BlueDragon.

In the case of the .NET edition, it's possible to call COM objects the .NET Framework supports COM objects by way of .NET's support for "runtime callable wrappers". See the manual, *Integrating CFML with ASP.NET and the Microsoft .NET Framework,* for more information.

In the case of the Java/J2EE editions, there are available Java-COM bridge products available (some open source, some commercial). Some of the alternatives include:

http://sourceforge.net/projects/jacob-project

http://www.gis.net/~amesar/links/java.html#JavaCOM

http://j-integra.intrinsyc.com/products/com/

New Atlanta has not yet tested these with BlueDragon, but since BlueDragon does support CFML-Java integration (see the *BlueDragon 6.2.1 User Guide*), it should be possible to call the Java objects created by these tools which would point to the intended COM objects.

6.11 Request Timeout Processing in BlueDragon

BlueDragon does not support page execution timeouts in all the same ways developers may expect from their experience with ColdFusion.

First, BlueDragon does not currently support an admin console setting to "Timeout Requests" for all pages.

Second, like CFMX, BlueDragon does not support the use of REQUESTTIMEOUT in the URL query string.

The .NET edition of BlueDragon does support the new CFSETTING REQUESTTIMEOUT attribute, which was introduced in CFMX. If a CFML page on BlueDragon.Net gets the .NET error, "Request timed out", this can be resolved by adding the CFSETTING tag (or increasing its timeout value) either on the page that's timing out or in the application.cfm (to affect all pages in that application).

Finally, the Java/J2EE editions of BlueDragon do not support the CFSETTING REQUESTTIMEOUT attribute--where it is simply ignored--or any other way to limit execution time for a CFML page.