BlueDragon

BlueDragon™ 7.1

What’s New in BlueDragon 7.1

NEW ATLANTA COMMUNICATIONS, LLC

BlueDragon™ 7.1
What's New in BlueDragon 7.1

May 5, 2009
Version 7.1

Contents

What’s New in BlueDragon 7.1

1 Introduction

BlueDragon is a 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 scripting 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 Platform, Enterprise Edition (Java EE) technologies.

1.1 About This Manual

This *What’s New in BlueDragon 7.1* document is intended for customers who are upgrading from ColdFusion or previous versions of BlueDragon. This document provides an introduction and overview of new features available in BlueDragon 7.1 and 7.0.1.

Customers who are upgrading from ColdFusion 5 or ColdFusion MX should also refer to the *BlueDragon 7.1 CFML Compatibility Guide* for a description of differences between the BlueDragon and ColdFusion implementations of CFML; and, to the *BlueDragon 7.1 CFML Enhancements Guide* for a description of unique BlueDragon features that are not available in any version of ColdFusion.

1.2 BlueDragon Product Configurations

BlueDragon is currently available in three product configurations. Details about these configurations—BlueDragon Server JX, BlueDragon for Java EE 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.

1.3 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 discussion forum 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/biz/support/index.jsp`

1.4 Additional Documentation

The other manuals available in the BlueDragon documentation library are:

- *BlueDragon 7.1 CFML Compatibility Guide*
- *BlueDragon 7.1 CFML Enhancements Guide*
- *BlueDragon 7.1 Server JX Installation Guide*
- *BlueDragon 7.1 User Guide*
- *Deploying CFML on Java EE 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, and they are available in PDF format from New Atlanta’s web site:

`http://www.newatlanta.com/products/bluedragon/self_help/docs/index.cfm`

2 What's New in BlueDragon 7.1

BlueDragon 7.1 is classified as a "minor" feature release and is a free upgrade for customers who have purchased BlueDragon 7.0 or 7.0.1. Below is a list of the feature enhancements in BlueDragon 7.1; details are provided in the following sections.

<u>All BlueDragon editions</u>

- `CFIMAGE`: added the ability to crop, rotate, add border, adjust brightness and contrast, and apply gray scale filter; added support for `PNG` images
- ColdFusion 8 (CF8) compatible enhancements
    - `CFTHREAD` compatibility
    - `onMissingMethod` event handler for CFCs
    - `CFQUERY` added result variables to specify the ID of an inserted row
    - `MonthAsString()` function added the "locale" parameter
    - added "component" as a supported value for the `CFARGUMENT TYPE` attribute and `CFFUNCTION RETURNTYPE` attribute
- Added the "Generate UUID as CFTOKEN value" configuration option
- performance enhancements
- bug fixes

<u>BlueDragon for the Microsoft .NET Framework</u>

- IIS 7.0 integrated request pipeline
- IIS 7.0 integrated configuration
- IIS 7.0 integrated administration
- Added the ability to create CFCs from ASP.NET pages without inheriting from the `NewAtlanta.BlueDragon.CfmPage` class
- Added the `NewAtlanta.BlueDragon.CfmUserControl` class
- Added support for ASP.NET connection strings
- Added support for the Oracle Data Provider for .NET (ODP.NET)
- Added support for the IBM DB2 provider for ADO.NET

<u>BlueDragon Server JX</u>

- Upgraded embedded servlet engine to ServletExec 6.0
- IIS 7.0 HttpHandler replaces C/C++ web server adapter
- Added the `CFCProxy` class for invoking CFCs from Java/JSP

2.1 Enhancements in all BlueDragon editions

The following BlueDragon 7.1 feature enhancements are available in all BlueDragon editions.

2.1.1 CFIMAGE Enhancements

BlueDragon 7.1 adds the following enhancements to the `CFIMAGE` tag: the ability to crop, rotate, add border, adjust brightness and contrast, and apply gray scale filter. BlueDragon 7.1 also adds support for `PNG` image files.

Cropping an image is done using an action of “crop”, specifying the starting “x” and “y” coordinates (in pixels), and the new “width” and “height” attributes (in pixels) as shown here:

```
<cfimage action="crop" x=”20” y="20" width="118" height="111"
  srcfile="#srcFilePath#" destfile="#destFilePath1#">
```

Rotating an image is done using an action of “rotate” and the new “angle” attribute (in positive or negative degrees) as shown here:

```
<cfimage action="rotate" angle="-5" srcfile="#srcFilePath#"
  destfile="#destFilePath1#">
```

Creating a border around an image is done using an action of “border” and the new attributes “thickness” (in pixels) and “color” (an HTML color value) as shown here:

```
<cfimage action="border" thickness="5" color="red"
  srcfile="#srcFilePath#" destfile="#destFilePath1#">
```

Adjusting contrast is done using an action of “edit” and the new “contrast” attribute as shown here (the “contrast” value is a floating point number; a value of “0” will result in an entirely black image; a value of “1” will leave the image unchanged):

```
<cfimage action="edit" contrast="1.25" srcfile="#srcFilePath#"
  destfile="#destFilePath1#">
```

Adjusting brightness is done using an action of “edit” and a new “brightness” attribute as shown here (the “brightness” value is an integer; a value of “-255” will result in an entirely black image; a value of “255” will result in an entirely white image; a value of “0” will leave the image unchanged):

```
<cfimage action="edit" brightness="128" srcfile="#srcFilePath#"
  destfile="#destFilePath1#">
```

Applying a grayscale filter is done using an action of “grayscale” as shown here:

```
<cfimage action="grayscale" srcfile="#srcFilePath#"
  destfile="#destFilePath1#">
```

The `TYPE` attribute now accepts the value `PNG`.

2.1.2 ColdFusion 8 (CF8) Compatible Enhancements

BlueDragon 7.1 introduces several feature enhancements for compatibility with ColdFusion 8 (CF8) as described in the following sections.

2.1.2.1 CFTHREAD

Multi-threaded programming via the `CFTHREAD` tag was first made available to CFML programmers with the release of BlueDragon 7.0 in March 2007 (see the description of the `CFTHREAD` tag later in this document). When CF8 was released several months later, it contained its own `CFTHREAD` implementation the differed in minor ways from the BlueDragon 7.0 implementation. For reference, here's a link to the CF8 documentation for `CFTHREAD`:

`http://livedocs.adobe.com/coldfusion/8/Tags_t_04.html`

BlueDragon 7.1 retains full backwards-compatibility with the BlueDragon 7.0 implementation of `CFTHREAD` and related tags and functions as described later in this document, and adds the following enhancements for compatibility with CF8:

- BlueDragon 7.1 adds support for the `CFTHREAD` metadata variables.
- The `CFTHREAD` tag body in BlueDragon 7.1 has access to all variable scopes that are available to the CFML page in which the `CFTHREAD` tag is embedded (in BlueDragon 7.0 the `CFTHREAD` tag body has limited access to variable scopes as described later in this document).
- BlueDragon 7.1 adds support for the `CFTHREAD` variable scope.
- The `CFTHREAD` tag in BlueDragon 7.1 now supports the `ACTION` attribute.
- BlueDragon 7.1 adds support for the `Sleep()` function.
- The `OUTPUT` attribute of the `CFTHREAD` tag in BlueDragon 7.1 now defaults to "true"; in BlueDragon 7.0 the `OUTPUT` attribute defaults to "false" (CF8 does not support the `OUTPUT` attribute for `CFTHREAD`).

2.1.2.2 onMissingMethod

BlueDragon 7.1 adds support for the `onMissingMethod` error handler as described in the CF8 documentation:

`http://livedocs.adobe.com/coldfusion/8/Tags_c_10.html`

2.1.2.3 CFQUERY

BlueDragon 7.1 adds support for the following result variables to contain the ID of an inserted row:

IDENTITYCOL – Microsoft SQL Server

ROWID – Oracle

SYB_IDENTITY – Sybase

SERIAL_COL – Informix

GENERATED_KEY – MySQL

CF8 documentation for the `CFQUERY` tag is available here:

http://livedocs.adobe.com/coldfusion/8/Tags_p-q_17.html

2.1.2.4 MonthAsString()

BlueDragon 7.1 adds support for the "locale" parameter to the `MonthAsString()` function as described in the CF8 documentation:

http://livedocs.adobe.com/coldfusion/8/functions_m-r_06.html

2.1.2.5 "component" Value

BlueDragon 7.1 adds support for the "component" value for the `CFARGUMENT TYPE` attribute and `CFFUNCTION RETURNTYPE` attribute as described in the CF8 documentation:

http://livedocs.adobe.com/coldfusion/8/Tags_a-b_6.html

http://livedocs.adobe.com/coldfusion/8/Tags_f_21.html

2.1.3 "Generate UUID as CFTOKEN value" Option

By default, BlueDragon 7.0.1 and earlier versions always generate UUIDs for CFTOKEN values. However, BlueDragon 7.0.1 also supports an option for "ColdFusion-compatible client data"; when this option is selected, BlueDragon 7.0.1 always generates ColdFusion-compatible integers—not UUIDs—for CFTOKEN values.

With BlueDragon 7.1, it is now possible to specify "ColdFusion-compatible client data" and "Generate UUID as CFTOKEN value" as independent options. These options are use for sharing client data when deploying BlueDragon in clusters with ColdFusion servers. In this configuration, you must set the "ColdFusion-compatible client data" option. For ColdFusion 5 (CF5) and earlier versions, the "Generate UUID as CFTOKEN value" must not be selected. For ColdFusion MX (CFMX) and later versions, including CF8, the "Generate UUID as CFTOKEN value" should be set to match the equivalent ColdFusion setting.

2.1.4 Performance Enhancements

A number of significant performance enhancements were implemented in BlueDragon 7.1. The impact of these performance enhancements will vary based on the specifics of your web applications.

2.1.5 Bug Fixes

A list of bug fixes in BlueDragon 7.1 can be found in the online bug tracking database:

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

2.2 BlueDragon for the Microsoft .NET Framework

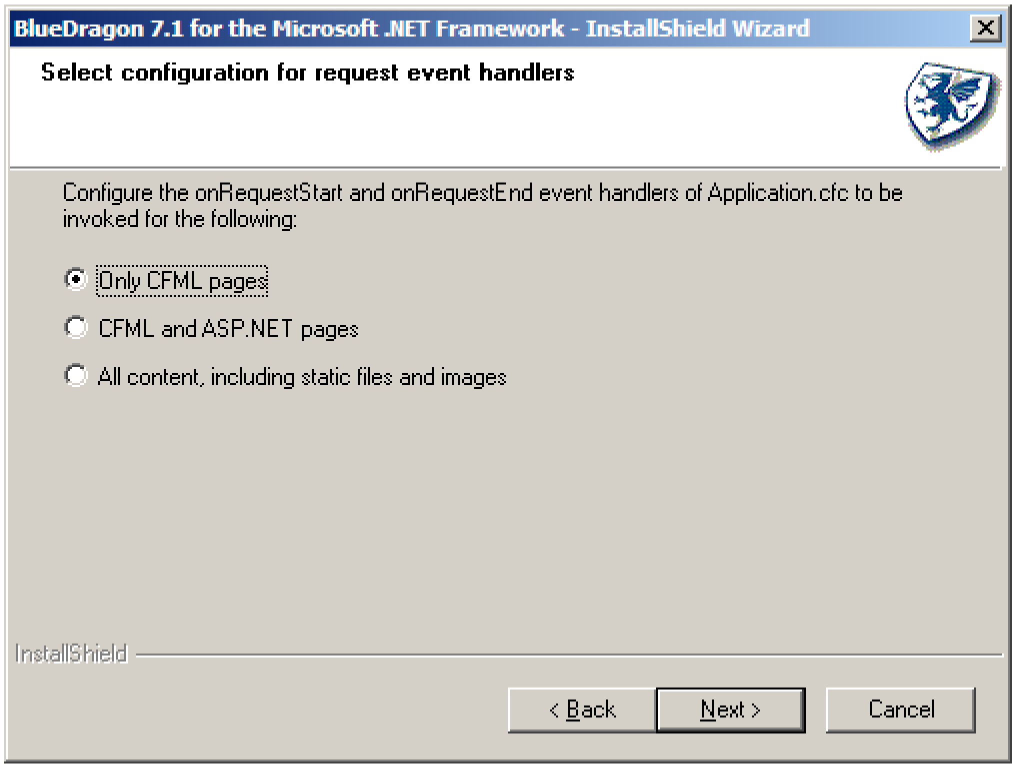
The following enhancements are only available in BlueDragon 7.1 for the Microsoft .NET Framework; these enhancements are not available in the Java editions of BlueDragon.

2.2.1 IIS 7.0 Integrated Request Pipeline

When deployed on IIS 7.0 on Windows Vista or Windows Server 2008, BlueDragon 7.1 takes advantage of the new ASP.NET and IIS 7.0 integrated request pipeline to optionally allow the `onRequestStart` and `onRequestEnd` events handlers of Application.cfc to be run for all content served by IIS 7.0—such as static HTML files, GIF and JPG images, ASP.NET `.aspx` pages, etc.—and not just CFML pages.

As illustrated in the figure below, the BlueDragon 7.1 installer gives you the option to configure the `onRequestStart` and `onRequestEnd` event handler to be invoked for:

- Only CFML pages (this is the default for backwards compatibility with BlueDragon 7.0.1 and ColdFusion);
- CFML and ASP.NET pages; or,
- All content, including static files and images.


BlueDragon 7.1 for the Microsoft .NET Framework - InstallShield Wizard
InstallShield
< Back
Next >
Cancel

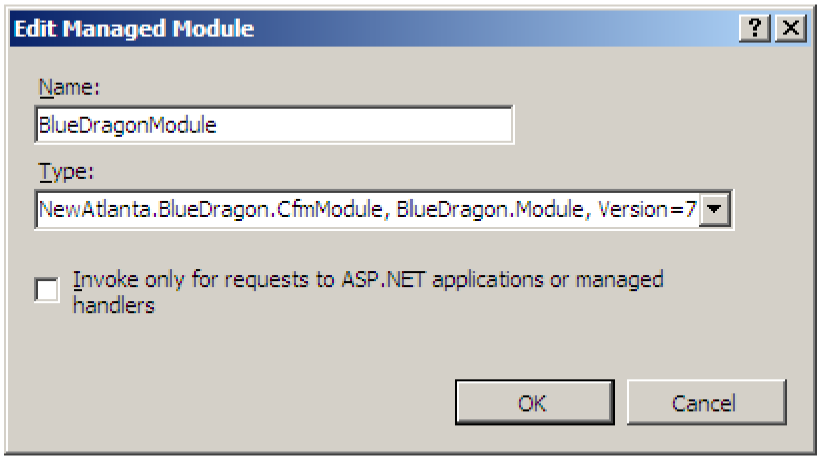
Select configuration for request event handlers

Configure the onRequestStart and onRequestEnd event handlers of Application.cfc to be invoked for the following:

[✓ Only CFML pages]
CFML and ASP.NET pages
All content, including static files and images

After installing BlueDragon 7.1, the initial configuration option can be modified or overridden for individual web sites or web applications by editing the Modules configuration data from the IIS Manager. If the BlueDragonModule is not configured for a web site or web application, this is equivalent to the “Only CFML pages” option of the installer.

If the BlueDragonModule is configured—as illustrated in the figure below—selecting the “Invoke only for requests to ASP.NET applications or managed handlers” option is equivalent to the “CFML and ASP.NET pages” option of the installer. Otherwise, it is equivalent to the “All content, including static files and images” installer option.



Name: BlueDragonModule

Type: NewAtlanta.BlueDragon.CfmModule, BlueDragon.Module, Version=7

☐ Invoke only for requests to ASP.NET applications or managed handlers

2.2.2 IIS 7.0 Integrated Configuration

When deployed on IIS 7.0 on Windows Vista or Windows Server 2008, BlueDragon 7.1 takes advantage of the new integrated configuration model of ASP.NET and IIS 7.0 to store all BlueDragon configuration data within the IIS 7.0 `applicationHost.config` and `web.config` files. Among the advantages this provides are:

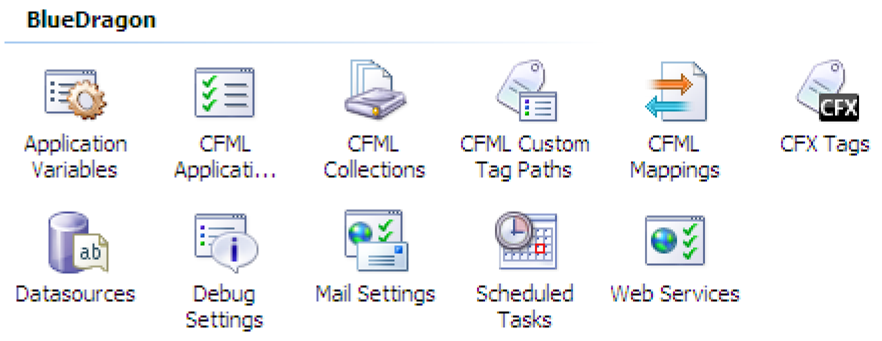
- All IIS 7.0, ASP.NET, and BlueDragon configuration data is now stored in the same plain-text XML-based configuration files. This makes it very easy to replicate configuration data across multiple servers in a cluster, or to copy configuration data from a staging server to a production server.
- BlueDragon can be configured independently for every web site and web application (this feature was available in earlier versions of BlueDragon, but is made even easier with this new feature).
- IIS 7.0 configuration files exist in an inheritance hierarchy, allowing you to configure global setting within `applicationHost.config` and then override or modify those settings for individual web sites or web applications via `web.config` files. BlueDragon 7.1 now takes advantage of this inheritance hierarchy.
- All of the IIS 7.0 configuration and administration tools can be used to configure and administer BlueDragon:

  `http://www.iis.net/getstarted/PowerfulAdministrationTools`

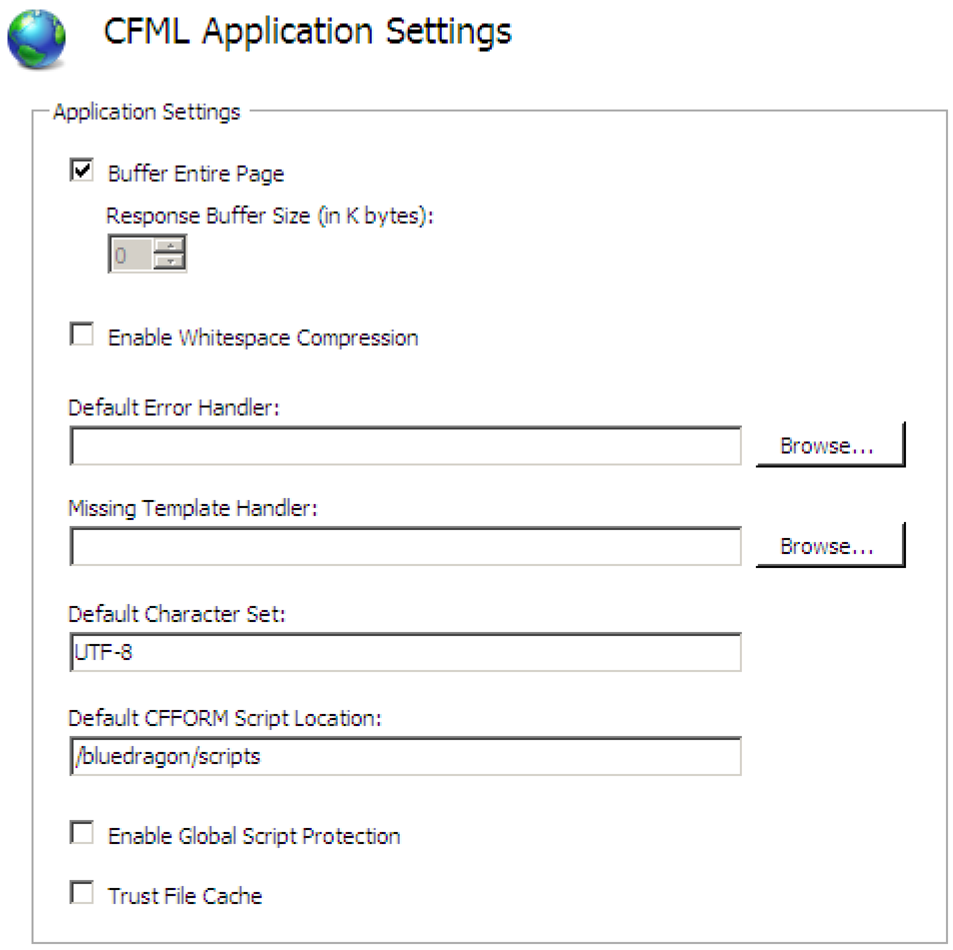
The IIS 7.0 Integrated Configuration feature is very tightly coupled with the IIS 7.0 Integrated Administration feature discussed in the next section of this document.

2.2.3 IIS 7.0 Integrated Administration

When deployed on IIS 7.0 on Windows Vista or Windows Server 2008, the new IIS Manager for IIS 7.0 is used to configure and administer BlueDragon; this replaces the browser-based administration console of BlueDragon 7.0.



As illustrated in the figures above and below, BlueDragon 7.1 installs a set of extensions to the IIS Manager for configuring BlueDragon. There are help pages available for each extension, though their use will be readily apparent to anyone familiar with ColdFusion or earlier versions of BlueDragon. See the *Deploying CFML on ASP.NET and the Microsoft .NET Framework* document for additional information.



CFML Application Settings

Application Settings

[✓] Buffer Entire Page

Response Buffer Size (in K bytes): 0

[ ] Enable Whitespace Compression

Default Error Handler:

Missing Template Handler:

Default Character Set: UTF-8

Default CFFORM Script Location: /bluedragon/scripts

[ ] Enable Global Script Protection

[ ] Trust File Cache

2.2.4 Creating CFCs without inheriting from CfmPage

As described in Section 5 of the *Integrating CFML with ASP.NET and the Microsoft .NET Framework* document, it's possible for ASP.NET pages that inherit from the `NewAtlanta.BlueDragon.CfmPage` class to invoke CFCs by using the `CreateComponent()` method of the base class. For example, to create an instance of the "employees" CFC and invoke its "getemployees" method from C#:

```
using NewAtlanta.BlueDragon;

CfComponent EmployeesCFC = base.CreateComponent( "employees" );
System.Data.DataTable employeesTable =
                        EmployeesCFC.Invoke( "getemployees" );
```

In BlueDragon 7.1, it's not necessary for an ASP.NET page to inherit from the `CfmPage` class in order to create CFCs. A new constructor has been added to the `CfComponent` class so that the above example can be rewritten as follows:

```
using NewAtlanta.BlueDragon;

CfComponent EmployeesCFC = new CfComponent( "employees" );
System.Data.DataTable employeesTable =
                        EmployeesCFC.Invoke( "getemployees" );
```

Note that when passing a CFC name to the `CfComponent` constructor (or to the `Create-Component` method of `CfmPage`), the same path lookup and mapping rules are used to locate the CFC source code file as if the CFC was being created from within a CFML page.

2.2.5 CfmUserControl

BlueDragon 7.1 adds the `NewAtlanta.BlueDragon.CfmUserControl` class to the `BlueDragon.Controls` library (see Section 5.1.1 of the *Integrating CFML with ASP.NET and the Microsoft .NET Framework* document for a discussion of the `BlueDragon.Controls` library). `CfmUserControl` extends the ASP.NET `System.Web.UI.UserControl` class, adding the `CfmPage` property. You can create user controls that extend `CfmUserControl` in order to invoke the `CfmPage` methods that provide for interaction between ASP.NET and CFML components. User controls that inherit from `CfmPage` can only be referenced from ASP.NET pages that inherit from `NewAtlanta.BlueDragon.CfmPage`.

2.2.6 ASP.NET Connection Strings

BlueDragon 7.1 supports the use of ASP.NET connection strings in place of BlueDragon datasources. When rendering a `CFQUERY` tag—or any other tag that specifies a `DATASOURCE` attribute—BlueDragon 7.1 will first look for a configured datasource of the specified name. If a BlueDragon datasource of the specified name is not found, BlueDragon will then look for an ASP.NET connection string of that name.

This feature allows ASP.NET connections strings and associated ADO.NET connection pools to be shared across ASP.NET and CFML pages.

2.2.7 Additional ADO.NET Providers

BlueDragon 7.1 adds support for the Oracle Data Provider for .NET (ODP.NET) and the IBM DB2 provider for ADO.NET. These providers are not included with BlueDragon, but must be downloaded and installed separately.

2.3 BlueDragon Server JX

The following enhancements are only available in BlueDragon Server JX 7.1.

2.3.1 Upgraded ServletExec Engine

The embedded servlet engine in BlueDragon Server JX 7.0.1 and earlier versions is based on New Atlanta's ServletExec 4.0. In BlueDragon 7.1, the embedded servlet engine has been upgraded to ServletExec 6.0, which supports all of the latest Java Servlet and JSP specifications, including:

- Java Servlet API 2.5
- JavaServer Pages (JSP) 2.1
- JSP Standard Tag Library (JSTL) 1.2

Visit New Atlanta's web site for additional information regarding ServletExec 6.0:

`http://www.newatlanta.com/products/servletexec/index.jsp`

One of the key features provided by ServletExec 6.0 is support for servlet filters, which allows third-party products based on servlet filters—such as the FusionReactor server monitor—to be deployed on BlueDragon Server JX 7.1. Visit Intergral's web site for additional information regarding FusionReactor:

`http://www.fusion-reactor.com/`

New Atlanta's web site contains a FAQ describing how to install and configure FusionReactor on BlueDragon Server JX 7.1:

`http://www.newatlanta.com/c/support/bluedragon/self_help/faq/detail?faqId=352`

2.3.2 IIS 7.0 HttpHandler

When deployed on IIS 7.0 on Windows Vista or Windows Server 2008, BlueDragon Server JX 7.1 integrates with IIS 7.0 via a new .NET-based HttpHandler that replaces the native C/C++ ISAPI web server adapter used with earlier versions of IIS.

The new IIS 7.0 HttpHandler provides improved performance and reliability over the C/C++ ISAPI web server adapter, and provides opportunities for enhanced integration with IIS 7.0 and ASP.NET that will be implemented in future versions of BlueDragon Server JX.

2.3.3 CFCProxy

BlueDragon 7.1 introduces the `com.newatlanta.cfc.CFCProxy` class that allows you to create and invoke methods on CFCs from Java servlets or JSP pages. The `CFCProxy` class declares two constructors with the following signatures:

```
public CFCProxy( String componentName,
            javax.servlet.http.HttpServletRequest request,
            javax.servlet.http.HttpServletResponse response )
      throws com.naryx.tagfusion.cfm.engine.cfmRunTimeException

public CFCProxy( String componentName,
            javax.servlet.http.HttpServletRequest request,
            javax.servlet.http.HttpServletResponse response,
            java.io.Writer writer )
      throws com.naryx.tagfusion.cfm.engine.cfmRunTimeException
```

The `componentName` argument takes the same value that you would pass to the `CFOBJECT` tag or `CreateObject()` function when creating a CFC within CFML; the same path lookup and mapping rules are used to locate the CFC source code file as if the CFC was being created from within a CFML page.

The first constructor causes the output of any subsequent CFC method invocations to be discarded; the second constructor causes the output of any subsequent CFC method invocations to be written to the provided `writer`.

After creating a `CFCProxy` instance, methods on the CFC can be invoked using the `invoke` method, which has the following signature:

```
public Object invoke( String methodName, Object... args )
            throws cfmRunTimeException, java.io.IOException
```

The `CFCProxy` class also provides a method for retrieving the “this” scope of the CFC as a `java.util.Map` instance:

```
public java.util.Map getThisScope()
```

3 What's New in BlueDragon 7.0.1

Below is a list of the new BlueDragon 7.0.1 features; details are provided in the following sections. All features are supported on all BlueDragon editions unless otherwise noted.

- Multi-threaded programming (`CFTHREAD`, and related tags and functions)
- Interfaces and Abstract CFCs
- "null" keyword and `IsNull()` function
- `CFQUERY` Enhancements
    - `CACHEDUNTILCHANGE` attribute
    - `BACKGROUND` attribute
- Application.cfc
    - `onClientStart()` handler
    - `onMissingTemplate()` handler
- `CFDOCUMENT`
- `CFCHART`
- `CFSEARCH` Enhancements
    - support for Word and PDF documents
    - support for multiple languages
- UDF Forward References in CFML pages
- `CFREGISTRY` support for Windows registry
- ASP.NET Partial Trust Mode (BD.NET)
- support for MySQL 5.0, including stored procedures
- support for 64-bit Windows and Linux
- support for Mac OS X on the Intel architecture
- miscellaneous CFMX 7 compatible enhancements
- performance enhancements
- bug fixes

3.1 Multi-threaded programming (CFTHREAD, etc.)

BlueDragon 7.0 introduces several news tags and functions to support multi-threaded programming. The basic idea is very simple: any CFML code within the body of the new CFTHREAD tag is rendered on a separate (new) thread while the main request thread continues processing after the closing </cfthread> tag without waiting for the CFTHREAD body to finish rendering. This is sometimes known as asynchronous processing; it gives you a way to do two things at the same time.

When BlueDragon encounters a CFTHREAD tag, it creates a new thread and starts rendering the CFTHREAD body on the new thread. It then immediately continues rendering the rest of the page after the closing </cfthread> tag at the same time that the CFTHREAD body is rendering on its separate thread.

3.1.1 CFTHREAD Variables

The CFTHREAD body does not have access to the Variables scope of the page or CFC that contains the CFTHREAD tag; instead—like a custom tag—the CFTHREAD body has its own separate Variables scope. Unlike a custom tag, there is no Caller scope within the CFTHREAD body.

The CFTHREAD body can invoke functions that are declared in the page that contains the CFTHREAD tag. If the CFTHREAD tag is rendered within a CFC function or pseudo-constructor, the CFTHREAD body has access to the CFC "this" scope and may invoke any function defined for the CFC.

Like a custom tag, CFTHREAD tag attributes are available in the Attributes scope of the CFTHREAD tag body. You may specify a struct in the ATTRIBUTECOLLECTION attribute of the CFTHREAD tag; the fields of this struct are copied to the Attributes scope of the CFTHREAD body.

The CFTHREAD body has access to the Application scope (if it exists), and the Server scope. The CTHREAD body does not have access to any of the following scopes: CGI, Request, Form, URL, Cookie, Session, or Client. You should not attempt to create Session or Client scopes by using the CFAPPLICATION tag within a CFTHREAD body (doing so will throw an exception in future BlueDragon 7.0 releases; the beta1 release will not throw an exception but will produce unpredictable results).

3.1.2 CFTHREAD EXAMPLE 1

Assume that every time someone logs in to your web site, you want to check and see if they've made any purchases in the past 30 days. If they haven't, then you want to send them an email with special offers to entice them to make another purchase.

The point of this example is that we have two potentially long-running actions that have nothing to do with displaying the page: querying the database to see if the customer has made any purchases in the past 30 days and then sending them an email if they haven't. Without the CFTHREAD tag you'd have to run these actions synchronously within the page, which means your customer would have to wait for these to finish before the page is rendered and returned to the browser.

With the CFTHREAD tag, the long-running actions can be done at the same time that the rest of the page is rendered. In fact, the page can be completely rendered and sent back to the customer before the CFTHREAD body is even finished.

```
<cfthread customer="#customer.ID#" email="#customer.email#">
    <!--- see if any purchases in the past 30 days --->
    <cfquery datasource="myStore" name="purchases">
    SELECT * FROM purchases
    WHERE ( customerID =
             <cfqueryparam value="#attributes.customerID#"> )
    AND ( purchaseDate >
             <cfqueryparam value="#DateAdd( 'd', -30, Now() )#"> )
    </cfquery>

    <cfif purchases.recordCount eq 0>
        <cfmail to="#attributes.email#" from="sales@mystore.com"
                subject="Special Offers">
        Please come back and buy more stuff!
        </cfmail>
    </cfif>
</cfthread>

<!--- display the page the customer requested --->
```

CFTHREAD Example 1

3.1.3 Joining (waiting for) a CFTHREAD

In computer science lingo, if you want one thread to wait for another thread to finish, this is called “joining” the thread. BlueDragon allows any thread to join (wait for) any thread created by CFTHREAD. To join a thread you must give it a name using the NAME attribute of the CFTHREAD tag. You then use the CFJOIN tag to wait for the CFTHREAD body to complete, passing the name as an attribute to CFJOIN; for example:

```
<cfthread name="myThread">
    <!--- do some work here --->
</cfthread>

<!--- do some work here --->

<!--- wait up to 100 milliseconds for myThread to finish
        before continuing --->
<cfjoin thread="myThread" timeout="100">
```

Joining threads in this way allows you to do two separate subtasks in parallel, but not to finish the overall page rendering until all subtasks are completed (a detailed example follows). The TIMEOUT value is specified in milliseconds; setting the TIMEOUT value to “0” will wait indefinitely.

3.1.4 Returning Data from a CFTHREAD

Sometimes you'll want to create a new thread to perform some work in parallel to the request thread, but then get results back from the CFTHREAD execution. When you specify the NAME attribute with CFTHREAD, BlueDragon creates a thread variable with that name (this is similar to the way CFQUERY works: after the query executes, a query variable is created using the NAME attribute of CFQUERY).

A thread variable is a structure that has three fields: Name, GeneratedContent and ReturnVariable. A thread variable may also contain an Exception field if an exception occurs during execution of the CFTHREAD body; more on this later. Note that these fields are not populated until after the CFTRHEAD body is finished rendering.

If the OUTPUT attribute is set to YES/TRUE, the GeneratedContent field of the thread variable contains the content generated by rendering the CFTHREAD tag body. Otherwise the GeneratedContent field is an empty string. The default value for the OUTPUT attribute is NO/FALSE. Here's an example:

```
<cfthread name="countingThread" output="true" number="10">
    <cfloop from="1" to="#attributes.number#" index="i">
        <cfoutput>#i#</cfoutput>
    </cfloop>
</cfthread>

<!--- wait for countingThread to finish --->
<cfjoin thread="countingThread">

<!--- output should be: 1 2 3 4 5 6 7 8 9 10 --->
<cfoutput>#countingThread.generatedContent#</cfoutput>
```

The ReturnVariable field of the thread variable contains a variable returned by a CFRETURN tag within the CFTHREAD body, or an empty string if the CFTHREAD body does not contain a CFRETURN tag. Here's an example:

```
<cfthread name="helloThread">
    <cfreturn "Hello, World!">
</cfthread>

<!--- wait for helloThread to finish --->
<cfjoin thread="helloThread">

<!--- output should be: Hello, World! --->
<cfoutput>#helloThread.returnVariable#</cfoutput>
```

The ReturnVariable can be any CFML variable type, including complex types such as structs, arrays, queries, or CFCs.

3.1.5 CFTHREAD EXAMPLE 2

Example 2, below, demonstrates multiple web service calls being made on separate threads. The first three calls are rendered on threads created using CFTHREAD; the fourth call is made on the main request thread. After the request thread completes its web service call, it waits for (joins) the other three threads to display the final results. This is an example of three long-running tasks being run in parallel on separate threads, and shows how to use the ReturnVariable field of the thread variable to receive results of a CFTHREAD execution. It also illustrates the ability to invoke user-defined functions from within the CFTHREAD body. (See Example 2 on the next page).

3.1.6 CFTHREAD Error Handling

You can use CFTRY/CFCATCH within your CFTHREAD body to handle errors as you would normally; for example:

```
<cfthread>
    <cftry>
        <!--- do some work here --->

        <cfcatch>
            <!--- handle all exceptions here --->
         </cfcatch>
    </cftry>
</cfthread>
```

However, putting a CFTHREAD tag within a CFTRY body does not work the way you might expect, because the CFTHREAD body and the CFTRY/CFCATCH are not running on the same thread. In the following example, the CFCATCH clause will not catch exceptions thrown within the CFTHREAD body:

```
<cftry>
    <cfthread>
        <!--- do some work here --->
    </cfthread>

    <cfcatch>
        <!--- cfthread exceptions are not caught here --->
    </cfcatch>
</cftry>
```

If you don’t use CFTRY/CFCATCH within your CFTHREAD body and an exception occurs, BlueDragon attempts to invoke error handlers in the following order: (1) the Application.cfc onError() handler; (2) the error handler defined by CFERROR; and, (3) the global error handler configured via the admin console. If no error handler is found, a runtime log file is written to the BlueDragon “work/temp/rtelogs” directory.

Regardless of whether an error handler is successfully invoked, exceptions within CFTHREAD that are not caught by CFTRY/CFCATCH are return in the Exception field of the thread variable.

```
<!--- run first three stock quotes on separate threads --->
<cfthread name="adbe">
    <cfreturn GetStockQuote( "ADBE" )>
</cfthread>

<cfthread name="beas">
    <cfreturn GetStockQuote( "BEAS" )>
</cfthread>

<cfthread name="msft">
    <cfreturn GetStockQuote( "MSFT" )>
</cfthread>

<!--- run RHAT on the request thread --->
<cfset rhat = GetStockQuote( "RHAT" )>

<!--- wait for the first three threads to finish --->
<cfjoin thread="adbe">
<cfjoin thread="beas">
<cfjoin thread="msft">

<cfoutput>
<table border>
<th>Company</th>
<th>Stock Price</th>
<th>Quote Time</th>
<tr>
    <td>Adobe</td>
    <td>#adbe.returnVariable.Price#</td>
    <td>#adbe.returnVariable.Time#</td>
</tr>
<tr>
    <td>BEA</td>
    <td>#beas.returnVariable.Price#</td>
    <td>#beas.returnVariable.Time#</td>
</tr>
<tr>
    <td>Microsoft</td>
    <td>#msft.returnVariable.Price#</td>
    <td>#msft.returnVariable.Time#</td>
</tr>
<tr>
    <td>Red Hat</td>
    <td>#rhat.Price#</td>
    <td>#rhat.Time#</td>
</tr>
</table>
</cfoutput>

<cffunction name="GetStockQuote" returntype="struct" output="false">
    <cfargument name="tickerSymbol" type="string" required="true">
    <cfinvoke webservice="http://www.webservicex.net/stockquote.asmx?WSDL"
              method="GetQuote" symbol="#tickerSymbol#"
              returnvariable="strQuoteXML">
    <cfset objQuote = xmlParse( strQuoteXML )>
    <cfset structQuote = StructNew()>
    <cfset structQuote.price = objQuote.StockQuotes.Stock.Last.XMLText>
    <cfset structQuote.time = objQuote.StockQuotes.Stock.Time.XMLText>
    <cfreturn structQuote>
</cffunction>
```

CFTHREAD Example 2

3.1.7 Additional Tags and Functions

In addition to CFTHREAD and CFJOIN, there are several other tags and functions related to multi-threaded programming:

<u>CFPAUSE</u> – causes a thread to “sleep” or pause processing for the number of seconds specified in the INTERVAL attribute

<u>CFINTERRUPT</u> – “wakes up” a thread—specified by the THREAD attribute—which is “sleeping” as the result of rendering the CFPAUSE tag

<u>GetAllThreads()</u> – returns an array of thread variables representing all actively running threads created using CFTHREAD. This function does not take any parameters.

<u>ThreadInterrupt(thread)</u> – this function allows you to “wake up” a thread that is “sleeping” as the result of rendering the CFPAUSE tag (same behavior as the CFINTERRUPT tag). The parameter is a thread variable.

<u>ThreadIsAlive(thread)</u> – returns a boolean indicating whether the specified thread is still running. The parameter is a thread variable.

<u>ThreadJoin(thread, [timeout])</u> – causes the current thread to wait for the specified thread (same behavior as the CFJOIN tag). The first parameter is a thread varible; the second parameter is optional, and is the timeout in milliseconds.

<u>ThreadRunningTime(thread)</u> – returns the amount of time in milliseconds that the thread has been running. The parameter is a thread variable.

<u>ThreadStop(thread)</u> – halts execution of the specified thread. The parameter is a thread variable. ThreadStop() runs asynchronously. If it is important that the thread be stopped before any further processing is one in the page that call ThreadStop(), either ThreadJoin() or CFJOIN should be used immediately after ThreadStop().

3.2 Interfaces and Abstract CFCs

Interfaces and abstract classes are common and useful features of most object-oriented programming languages such as C++, Java, and C#. BlueDragon 7.0 introduces these features for CFCs. If you're familiar with these concepts, then their implementation in BlueDragon 7.0 should be straightforward and easy to understand.

If the concepts of interfaces and abstract classes are new to you, it helps to begin by understanding abstract functions. An abstract function is a function that does not (and cannot) implement a function body. In BlueDragon 7.0, a function can be declared to be abstract using the new TYPE attribute of the CFFUNCTION tag:

```
<cffunction name="myAbstractFunction" type="abstract"/>
```

Abstract functions can only be declared within CFCs; that is, you cannot declare abstract user-defined functions within CFML pages. The body of an abstract function may only contain CFARGUMENT tags; if it contains any other tags an exception is thrown.

A CFC that contains one or more abstract functions must be declared to be abstract using the new TYPE attribute of the CFCOMPONENT tag (an "abstract CFC"):

```
<cfcomponent type="abstract">
    <cffunction name="myConcreteFunction">
        <cfreturn "I have an implementation, so I'm concrete"/>
    </cffunction>

    <cffunction name="myAbstractFunction" type="abstract"/>
    ...
</cfcomponent>
```

You may not create an instance of an abstract CFC using CFOBJECT, CreateObject, or CFINVOKE. Abstract CFCs may only be used as base classes for other CFCs (that is, other CFCs may extend abstract CFCs). If a CFC extends an abstract CFC, the subclass CFC must implement all of the abstract functions declared by the abstract superclass CFC, or the subclass CFC must itself be decared abstract.

If a CFC contains only abstract functions, it may be declared to be an interface using the new TYPE attribute of the CFCOMPONENT tag (an "interface CFC"):

```
<cfcomponent type="interface">
    <cffunction name="abstractFunctionOne" type="abstract"/>
    <cffunction name="abstractFunctionTwo" type="abstract"/>
    <cffunction name="abstractFunctionThree"/>
    ...
</cfcomponent>
```

Because an interface CFC may only contain abstract functions, you can omit the type="abstract" declaration for its CFFUNCTION tags. Like abstract CFCs, you cannot create an instance of an interface CFC.

An interface CFC may extend another interface CFC, but may not extend an abstract CFC or concrete (“regular”) CFC.

CFCs may declare that they implement one or more interface CFCs using the new IMPLEMENTS attribute of the CFCOMPONENT tag. A CFC may implement multiple interfaces by declaring them as a comma-separated list:

```
<cfcomponent implements="interfaceOne, interfaceTwo"
             extends="mySuperClass">
    ...
</cfcomponent>
```

A CFC must implement all of the abstract functions declared by all of the interfaces declared in its IMPLEMENTS attribute.

An abstract CFC or interface CFC may be specified as the TYPE attribute for CFARGUMENT, or RETURNTYPE attribute for CFFUNCTION.

3.3 "null" keyword and IsNull() function

BlueDragon 7.0 introduces the concept of null variables in CFML via the "null" keyword and IsNull() function. There are several uses for these: (1) checking to see if a value returned by a database is null versus an empty string; (2) passing null values to Java object methods without using JavaCast; (3) checking for null values returned by Java object method calls; and, (4) returning null values from CFC function calls where the return type is specified to be a CFC.

3.3.1 Database nulls

Prior to BlueDragon 7.0, null values returned from a database were treated as empty strings; and, in BlueDragon 7.0 you can continue to treat them as empty strings for backwards compatibility with existing code. However, you now have the option of explicitly checking for null values returned from a database. A simple example demonstrates this:

```
<!--- get all employees whose email address is null --->
<cfquery datasource="cfsnippets" name="employees">
SELECT * FROM Employees
WHERE Email IS NULL
</cfquery>

<cfif employees.Email eq "">
    <h3>Yes, employees.Email equals empty string</h3>
</cfif>

<cfif employees.Email eq null>
    <h3>Yes, employees.Email is null</h3>
</cfif>

<cfif isNull( employees.Email )>
    <h3>Yes, employees.Email is still null</h3>
</cfif>
```

3.3.2 Java Methods

Many Java methods allow you to pass null values as parameters. For example, the java.io.File.createTempFile() method takes three parameters: prefix, suffix, and directory. The last two parameters may be null, in which case the default suffix (".tmp") and default directory are used. Prior to BlueDragon 7.0 you were required to use the JavaCast method to pass null values to Java methods; with BlueDragon 7.0 you can now simply use the "null" keyword.

```
<cfset f = createObject( "java", "java.io.File" )>

<!--- old way, using JavaCast (still works in BD 7.0) --->
<cfset tempFile = f.createTempFile( "test", JavaCast( "null", "" ),
                                        JavaCast( "null", "" ) )>

<!--- new way, using "null" keyword, BD 7.0 only --->
<cfset tempFile = f.createTempFile( "test", null, null )>
```

Similarly, many Java methods will return null under certain circumstances. For example, the java.io.File.getParentFile() method returns null if the specified path does not have a parent. In CFMX, if the return value of a Java method is assigned to a CFML variable, the result is that the variable becomes undefined, even if it was defined previously. For example, consider the following code:

```
<cfset parentFile="initial value">
<cfset testFile=createObject("java","java.io.File").init("C:\")>

<!--- returns null because "C:\" doesn't have a parent --->
<cfset parentFile=testFile.getParentFile()>
```

At this point, if you tried to output the value of "parentFile" in CFMX you'd get an "undefined variable" exception, which might seem a little strange since "parentFile" was created and assigned a value in the first CFSET tag.

In BlueDragon 7.0, "parentFile" gets assigned the value "null", which you can test for using the IsNull() function:

```
<cfif isNull( parentFile )>
    parent file is null
</cfif>
```

These may be trivial examples, but if you write a lot of code that call Java methods you'll find numerous examples where these issues become more critical. Support for the "null" keyword and IsNull() function allow you to interact with Java (and .NET) methods in a more natural manner.

3.3.3 CFC Functions

Just as there are Java methods that accept null values as parameters and return null values, there are many cases where it makes sense to write CFC functions that do the same. Prior to BlueDragon 7.0, however, there was no way to do this. If you specify a CFC as the TYPE attribute for a CFARGUMENT tag, or as the RETURNTYPE attribute for a CFFUNCTION tag, there's no simple, convenient way to specify a null value for an argument, or return a null value from a function.

Consider the following CFC named "Employee". This CFC implements a single function, "init" that takes an employee ID as a parameter (you can easily imagine additional functions to retrieve the employee data and otherwise manipulate an Employee instance). The init function uses the employee ID to look up the employee data in the company database and return an Employee instance populated with the data. The RETURNTYPE of the init function is specified as "Employee"; that is, it returns a CFC of type "Employee."

But what happens if the employee specified by the employee ID doesn't exist? Prior to BlueDragon 7.0 there really were no good solutions. But, now the answer is simple: return null from the init function, and have the caller check for a null return value using the IsNull() function. Here's the CFC:

```
<cfcomponent name="Employee" output="false">

    <cffunction name="init" returntype="Employee" output="false">
        <cfargument name="empID" type="numeric" required="true">

        <cfquery datasource="cfsnippets" name="employee">
        SELECT * FROM Employees
        WHERE EmpID = <cfqueryparam cfsqltype="cf_sql_integer"
                                      value="#arguments.empID#">
        </cfquery>

        <cfif employee.recordCount eq 0>
            <!--- the employee doesn't exist, so return null --->
            <cfreturn null>
        <cfelse>
            <!--- populate and return this instance --->
            <cfset this.empID = employee.empID>
            <cfset this.FirstName = employee.FirstName>
            <cfset this.LastName = employee.LastName>
            <cfset this.Email = employee.Email>
            <cfset this.Phone = employee.Phone>
            <cfset this.Department = employee.Department>

            <cfreturn this>
        </cfif>
    </cffunction>
</cfcomponent>
```

And here's how a caller would invoke the init method and use the IsNull() function to check for a valid Employee instance:

```
<cfset empID=1>

<cfset employee=createObject("component","Employee").init(empID)>
<cfif isNull( employee )>
    Employee <cfoutput>#empID#</cfoutput> does not exist
</cfif>
```

Support for null in BlueDragon 7.0 provides a simple, elegant way to solve to these types of programming problems.

3.4 CFQUERY Enhancements

BlueDragon 7.0 introduces two new attributes for the CFQUERY tag.

3.4.1 CACHEDUNTILCHANGE Attribute

Using the CACHEDWITHIN attribute for CFQUERY caching suffers from one major drawback: what's the right interval? If you choose an interval that's too short, then you're not getting the full benefits of caching because you're retrieving data more often than you should. But if you choose an interval that's too long, then you're sometimes using "stale" data. Unfortunately, there's often no good answer.

BlueDragon for the Microsoft .NET Framework (BlueDragon.NET), when used in conjunction with Microsoft SQL Server 2005 offers a better alternative to CACHEDWITHIN for query caching. The new CACHEDUNTILCHANGE attribute allows you to do "perfect" query caching. That is, BlueDragon caches the query results until notified by the database server that the data has changed. Thus you're always using the latest data, and data is retrieved from the database only when needed.

Simply set the CACHEDUNTILCHANGE attribute to "true" or "yes" on the CFQUERY tag:

```
<cfquery datasource="myCompany" name="engineers"
                                    cacheduntilchange="true">
SELECT EmpID, FirstName, LastName, Department
FROM Employees
WHERE Department = "Engineering">
</cfquery>
```

3.4.2 BACKGROUND Attribute

The new BACKGROUND attribute for CFQUERY allows you to perform database inserts or updates on a background thread. Processing of main request continues while the insert or update is processed on a separate thread. This is similar in concept to CFTHREAD, except that using the BACKGROUND attribute does not create a new thread for each tag execution. Instead, a single background thread is dedicated to all background operations, which are queued for execution by the CFQUERY tag when the BACKGROUND attribute is specified.

For example, consider a simple custom logger that you might implement in the onRequestEnd() event handler of your Appliction.cfc (see the example, below). The onRequetEnd() method gets invoked at the end of every request, but there's no need to delay sending the response to the user while the database insert is executed. Instead, set the BACKGROUND attribute to "true" or "yes" to have the insert done on a background thread.

Any database insert or update that does not affect page rendering is a candidate for using the BACKGROUND attribute.

```
<cffunction name="onRequestEnd" returnType="void" output="false">
    <cfargument name="targetPage" type="string" required="true">

    <!--- queue log insertion for background processing --->
    <cfquery datasource="myLogs" background="true">
    INSERT INTO RequestLog (
                RequestTime,
                RequestPage,
                REMOTE_ADDR,
                HTTP_USER_AGENT
    ) VALUES ( <cfqueryparam value="#Now()#">,
               <cfqueryparam value="#arguments.targetPage#">,
               <cfqueryparam value="#cgi.REMOTE_ADDR#">,
               <cfqueryparam value="#cgi.HTTP_USER_AGENT#">
    )
    </cfquery>
</cffunction>
```

3.5 Application.cfc

The Application.cfc component was introduced in CFMX 7 as an enhancement and replacement for Application.cfm and OnRequestEnd.cfm. The CFMX 7 Application.cfc implements the following event handlers: onApplicationStart(), onApplicationEnd(), onSessionStart(), onSessionEnd(), onRequestStart(), onRequestEnd(), onRequest(), and onError(). The BlueDragon 7.0 implementation of Application.cfc is compatible with CFMX 7, with the addition of two new event handlers, described below.

3.5.1 onClientStart() Event Handler

The new onClientStart() event handler introduced by BlueDragon 7.0 is invoked whenever a new client session is created. It has the following signature:

```
<cffunction name="onClientStart" returntype="void">
</cffunction>
```

Use the onClientStart() event handler to initialize Client scope variables.

3.5.2 onMissingTemplate() Event Handler

The new onMissingTemplate() event handler introduced by BlueDragon 7.0 is invoked whenever BlueDragon encounters a file-not-found condition. The search for the Application.cfc file starts in the physical directory represented by the request URI, and proceeds up the parent directories in the normal fashion. The signature of the onMissingTemplate() event handler is:

```
<cffunction name="onMissingTemplate" returnType="boolean">
    <cfargument type="String" name="targetPage" required="true">
    <cfreturn true>
</cffunction>
```

Return "true" to indicate that the event has been processed; return "false" to indicate that the event has not been processed. In the latter case, BlueDragon continues with its normal file-not-found processing, including rendering the site-wide Missing Template Handler (if configured) or returning a 404 file-not-found error page to the browser. You do not have to explicitly return true if you omit the returntype attribute.

Prior to invoking the onMissingTemplate method, the Application.cfc pseudo-constructor is rendered, but the application and session are not started, so those scopes and the client scope are not available. The remaining scopes--Request, URL, Form, CGI, Server, etc.--are all available within the onMissingTemplate handler.

The onApplicationStart, onSessionStart, onRequestStart, onRequest, and onRequestEnd handlers are not invoked, and processing of the request terminates when the onMissingTemplate handler returns. If an error occurs within the onMissingTemplate handler, the onError handler is not invoked; however, the site-wide Default Error Handler is invoked (if configured).

3.6 CFDOCUMENT

The CFDOCUMENT tag and its CFDOCUMENTITEM and CFDOCUMENTSECTION sub-tags were introduced in CFMX 7 for rendering PDF and FlashPaper documents.

The BlueDragon 7.0 implementation of the CFDOCUMENT tag and its CFDOCUMENTITEM and CFDOCUMENTSECTION sub-tags is compatible with CFMX 7, except that BlueDragon supports PNG and JPEG output formats instead of FlashPaper. BlueDragon also introduces several enhancements that are described below.

3.6.1 FORMAT attribute values “png” and “jpg”

BlueDragon 7.0 does not support “flash” as a value for the FORMAT attribute. BlueDragon 7.0 supports “pdf”, and introduces the new “png”, “jpg”, and “jpeg” values. Using “png”, “jpg”, or “jpeg” produces an image of the first page of the URL or CFDOCUMENT body based on the page size, margins, and specified scale. Such images are useful for creating “thumbnails” of web pages.

3.6.2 UNIT attribute value “px”

In addition to “in” and “cm”, BlueDragon 7.0 introducs the new “px” value for the UNIT attributes to define page and margin sizes in pixels. This particularly useful when creating images using the “png”, “jpg”, or “jpeg” values for the FORMAT attribute.

3.6.3 TIMEOUT attribute

BlueDragon 7.0 introduces the new timeout attribute to control the CFDOCUMENT rendering time. The attribute value unit is seconds; the default value is 60; a value of 0 indicates an infinite timeout. An exception is thrown if the CFDOCUMENT tag has not finished rendering before the timeout value expires.

3.6.4 WATERMARK and WATERMARKIMAGE attributes

BlueDragon 7.0 introduces the new WATERMARK and WATERMARKIMAGE attributes. The WATERMARK attribute accepts a text value that is used to create a slighly transparent watermark in the background of in the resulting PDF.

The WATERMARKIMAGE attribute accepts either the name of an image file, which must be in the same directory as the CFML page, or the full absolute file system path to an image file. The image will be stretched across each page and will be rendered with slight transparency. Using a .gif or .jpg with transparency works best; images with white backgrounds will fade the document contents noticeably.

3.6.5 AUTHOR, SUBJECT, TITLE, and KEYWORDS attributes

BlueDragon 7.0 introduces several new attributes that correspond to PDF document meta-data attributes: AUTHOR, SUBJECT, TITLE, and KEYWORDS.

3.7 CFCHART

CFCHART and its CFCHARTDATA and CFCHARTSERIES sub-tags were introduced in CFMX 6.0 and enhanced in later CFMX releases.

The BlueDragon 7.0 implementation of CFCHART and its CFCHARTDATA and CHCHARTSERIES sub-tags is compatible with CFMX 7, with differences and enhancements described below.

3.7.1 Differences

The BlueDragon 7.0 implementation of CFCHART differs from CFMX 7 in the following way:

1) CFCHART
   a. The STYLE attribute is not supported. Instead BlueDragon implements the DEFAULT attribute. See the enhancements section, below, for more details.
   b. The FONT attribute value "arialUnicodeMS" is not supported.
   c. The FORMAT attribute value "flash" is not supported.
   d. The GRIDLINES attribute is not supported. Instead BlueDragon implements the "yAxisUnits" attribute. See the enhancements section, below, for more details.
   e. The SERIESPLACEMENT attribute value "stacked" is only supported with category bar and horizontalbar charts.
   f. The SERIESPLACEMENT attribute value percent is only supported with 2D category bar and horinzontalbar charts.
   g. The SHOW3D attribute value "yes" is only supported with pie charts and category bar, line and horizontalbar charts.
   h. The TIPBDCOLOR attribute is not supported since it is specific to the flash format.
   i. The XOFFSET attribute takes a value between 0 and the CHARTWIDTH instead of between -1 and 1.
   j. The YOFFSET attribute takes a value between 0 and the CHARTHEIGHT instead of between -1 and 1.

2) CFCHARTSERIES
   a. The PAINTSTYLE attribute value "raise" is not supported.
   b. The PAINTSTYLE attribute values "shade" and "light" are only supported with bar, horizontal bar and area charts.
   c. The TYPE attribute values "pyramid", "cone", "curve" and "cylinder" are not supported.

3) CFCHARTDATA
   a. NONE

3.7.2 Enhancements

The BlueDragon 7.0 implementation of CHCHART introduces the following enhancements:

1) CFCHART
    a. CFCHART can be configured to store the generated charts to disk, to a session or to a DB. Session or DB storage should be used in clustered environments.
    b. CFCHART supports category charts with y values as symbols. These charts are generated by setting the yAxisType to symbols and setting the new attribute yAxisSymbols to a comma-separated list of symbols. If the symbols contain commas then a different separator can be specified by using the new attribute symbolsSeparator.
    c. New attributes:
        i. default – This attribute can be used to easily give many charts the same look-and-feel. This is done by assigning this attribute the location of a file that contains a CFCHART tag. The attribute values for the CFCHART tag in this file will be used as the default values for the CFCHART tag.
        ii. maxCategoryLabelLines – This attribute specifies the maximum number of lines that can be used for a category label. Category labels that need to be displayed on more than the maximum number of lines are truncated. The default value is 5.
        iii. yAxisUnits – This attribute specifes the units that are used for the tick marks on the y-axis. For example, if this attribute is set to 10 then the y-axis tick marks would be labeled 0, 10, 20, 30, etc.
        iv. xAxisUpperMargin – This attribute indicates how much of a margin is used on the x-axis. This value is specified as a percentage of the last x value. For example, if the last x value is 100 and this attribute is set to .1 then the x-axis will have a margin of 10. The default value is .05.
        v. yAxisUpperMargin – This attribute indicates how much of a margin is used on the y-axis. This value is specified as a percentage of the last y value. For example, if the last y value is 100 and this attribute is set to .1 then the y-axis will have a margin of 10. The default value is .05.
        vi. yAxisSymbols – This attribute is used to specify the symbols for a category chart that uses symbols for the y-axis. The default separator for the symbols is a comma.
        vii. symbolsSeparator – This attribute is used to specify the separator used by the yAxisSymbols attribute.

2) CFCHARTSERIES
    a. The markerStyle attribute also supports the values triangledown, triangleright, triangleleft, horizontalrectangle and verticalrectangle.
    b. The type attribute also supports the value ring.
    c. The dataLabelStyle attribute also accepts a string that contains the following markers: {0}, {1}, {2}, {3} and {4}. These markers are replaced by the following values:
        i. {0} – replaced by the series label

        ii. {1} – replaced by the category name or x-value
        iii. {2} – replaced by the y-value
        iv. {3} – replaced by the percent of the y-value of the total of all the y-values in the series
        v. {4} – replaced by the total of all the y-values in the series
    d. New attributes:
        i. negativeDataLabelPosition - This attribute specifies the position of the data labels relative to the data points for negative values. The valid values are TOP, TOP_INSIDE, LEFT, LEFT_INSIDE, CENTER, RIGHT_INSIDE, RIGHT, BOTTOM_INSIDE and BOTTOM. The XXX_INSIDE values are only relevant for bar and horizontalbar charts.
        ii. positiveDataLabelPosition - This attribute specifies the position of the data labels relative to the data points for positive values. The valid values are TOP, TOP_INSIDE, LEFT, LEFT_INSIDE, CENTER, RIGHT_INSIDE, RIGHT, BOTTOM_INSIDE and BOTTOM. The XXX_INSIDE values are only relevant for bar and horizontalbar charts.
        iii. dataLabelAngle – This attribute specifies the angle in degrees that the data labels are rotated. For positive values the data labels are rotated in a clock-wise direction.
        iv. dataLabelColor – This attribute specifies the color of the data labels.
        v. dataLabelFont – This attribute specifies the font used by the data labels.
        vi. dataLabelFontBold – This attribute specifies if the data labels are bold.
        vii. dataLabelFontItalic - This attribute specifies if the data labels are italic.
        viii. dataLabelFontSize – This attribute specifies the font size of the data labels.

3) <u>CFCHARTDATA</u>
    a. NONE

4) <u>CFCHARTRANGEMARKER</u> – this tag is used to add a range marker to the y-axis. One or more of these tags can be placed within a CFCHART tag. The attributes it supports are:
    a. start – the start value on the y-axis for the marker
    b. end – the end value on the y-axis for the marker
    c. color – the color of the marker
    d. label – the label for the marker
    e. labelColor – the color of the marker label
    f. labelPosition – the position of the label in the marker. Valid values are: TOP_LEFT, TOP, TOP_RIGHT, LEFT, CENTER, RIGHT, BOTTOM_LEFT, BOTTOM and BOTTOM_RIGHT.
    g. font – the font used by the marker label
    h. fontBold – specifies if the marker label is bold
    i. fontItalic – specifies if the marker label is italic
    j. fontSize – specifies the font size of the marker label

5) CFCHARTDOMAINMARKER – this tag is used to add a domain marker to the x-axis. One or more of these tags can be placed within a CFCHART tag. The attributes it supports are:
   a. value – the x-value to be highlighted by the marker
   b. color – the color of the marker
   c. shape – the shape of the marker. The valid values are line or region. The default value is line.
   d. label – the label for the marker
   e. labelColor – the color of the marker label
   f. labelPosition – the position of the marker label. Valid values are: TOP_LEFT, TOP, TOP_RIGHT, LEFT, CENTER, RIGHT, BOTTOM_LEFT, BOTTOM and BOTTOM_RIGHT.
   g. font – the font used by the marker label
   h. fontBold – specifies if the marker label is bold
   i. fontItalic – specifies if the marker label is italic
   j. fontSize - specifies the font size of the marker label

6) CFCHARTLEGEND – this tag is used to configure the legend of the generated chart. Only one of these tags can be placed within a CFCHART tag. The attributes it supports are:
   a. backgroundColor – the color of the legend background
   b. labelColor – the color of the labels in the legend
   c. font – the font used for the labels in the legend
   d. fontBold – specifies if the legend labels are bold
   e. fontItalic – specifies if the legend labels are italic
   f. fontSize – specifies the font size of the legend labels
   g. position – specifies the position of the legend relative to the generated chart. Valid values are: TOP, BOTTOM, LEFT and RIGHT.
   h. showBorder – specifies if a border is displayed around the legend

7) CFCHARTTITLE – this tag is used to configure the title(s) of the generated chart. One or more of these tags can be placed within a CFCHART tag. The attributes it supports are:
   a. Text – the text for the title
   b. backgroundColor – the color of the title background
   c. labelColor – the color of the title text
   d. font – the font used for the title text
   e. fontBold – specifies if the title text is bold
   f. fontItalic – specifies if the title text is italic
   g. fontSize – specifies the font size of the title text
   h. position - specifies the position of the title relative to the generated chart. Valid values are: TOP, BOTTOM, LEFT and RIGHT.
   i. showBorder – specifies if a border is displayed around the title
   j. padding – specifies the number of pixels used to add padding space around the title

k. margin – specifies the number of pixels used as a margin between the title and other objects in the chart

8) CFCHARTIMAGE – this tag is used to configure an image for the chart background. Only one of these tags can be placed within a CFCHART tag. The attributes it supports are:
   a. file – Required. The path to a GIF or JPG image. Can be either a full physical path or a relative path (see the uriDirectory attribute).
   b. uriDirectory – Optional. If YES, relative paths specified in File are calculated from the web server document root directory. If NO, relative paths are calculated as relative to the current file. Defaults to NO.
   c. alignment – Optional. Determines how the image is aligned in the background of the chart. Valid values are: TOP_LEFT, TOP, TOP_RIGHT, LEFT, CENTER, RIGHT, BOTTOM_LEFT, BOTTOM, BOTTOM_RIGHT, FIT, FIT_HORIZONTAL and FIT_VERTICAL. Defaults to FIT.

3.8 CFSEARCH Enhancements

The CFSEARCH implementation in BlueDragon 7.0 has been enhanced to include support for Word and PDF documents, and multiple languages.

3.8.1 Word and PDF Documents

The CFSEARCH implementation in BlueDragon 6.2 only supports text documents. BlueDragon 7.0 adds support for Word and PDF documents.

3.8.2 Multiple Languages

The CFSEARCH implementation in BlueDragon 6.2 only supports the English language. BlueDragon 7.0 adds support for the following languages: Brazilian, Chinese, Czech, Danish, Dutch, English, Finnish, French, German, Greek, Italian, Norwegian, Portugese, Russian, Spanish, Swedish, Korean, and Japanese.

3.9 UDF Forward References in CFML Pages

In BlueDragon 6.2, a user-defined function (UDF) must appear in the CFML page before the code that calls it; BlueDragon 6.2 does not support "forward references" to UDFs that appear later in the page.

BlueDragon 7.0 supports UDF forward references for both CFSCRIPT-based and CFFUNCTION-based UDFs.

3.10 CFREGISTRY Support for Windows Registry

BlueDragon 6.2 does not support accessing the Window registry using the CFREGISTRY tag; instead BlueDragon 6.2 implements an emulated registry on both Windows and UNIX/Linux.

BlueDragon 7.0 supports access to the Windows registry using the CFREGISTRY tag, and removes support for CFREGISTRY completely on UNIX/Linux (it no longer implements an emulated registry on UNIX/Linux).

3.11 ASP.NET Partial Trust Mode (BD.NET)

All of this information about partial trust mode is specific to BD.NET installed in a single virtual directory. When BD.NET is installed globally in the GAC then it is automatically trusted and you cannot make it partially trusted. Here's a blog that discusses this:

http://blogs.msdn.com/shawnfa/archive/2005/02/15/373604.aspx

3.11.1 General Info

1) The trust level for a .NET web application is set by adding the following element to web.config as a subelement of <system.web>: <trust level="<level>" originUrl=""/>.
2) .NET comes with 5 predefined trust levels: Full, High, Medium, Low and Minimal. These are defined in the global web.config file. Each trust level has its own policy file that is located in the global .NET CONFIG directory. Additional trust levels can be defined by creating a new policy file and configuring it in the global web.config file.

3.11.2 High Trust Mode

Using the predefined High trust level, the following BD.NET functionality will not work:

1) CFEXECUTE. This tag will only work in full trust mode.
2) RANDOMIZE. This function will only work in full trust mode.
3) Web services. This functionality only works in full trust mode.
4) CFINDEX with the language attribute set to finnish. This functionality can be enabled by changing the ReflectionPermission Flags attribute from "ReflectionEmit" to "AllFlags" in the web_hightrust.config policy file.
5) Auto-configuration of ODBC datasources. This functionality can be enabled by adding "UnmanagedCode" to the list of SecurityPermission Flags in the web_hightrust.config policy file.
6) CFFILE tag with an ATTRIBUTES attribute. This functionality can be enabled by adding "UnmanagedCode" to the list of SecurityPermission Flags in the web_hightrust.config policy file.
7) CFMULTICAST. This tag can be enabled by adding "UnmanagedCode" to the list of SecurityPermission Flags in the web_hightrust.config policy file.
8) Client data. This functionality can be enabled by adding "SerializationFormatter" to the list of SecurityPermission Flags in the web_hightrust.config policy file.
9) Precompiled templates. This functionality can be enabled by adding "SerializationFormatter" to the list of SecurityPermission Flags in the web_hightrust.config policy file.
10) CFX tags. This functionality can be enabled by adding "ControlAppDomain" to the list of SecurityPermission Flags in the web_hightrust.config policy file.
11) CFLDAP. This tag can be enabled by making the following changes to the web_hightrust.config policy file:
    a. Add the following security class to the list of security classes: <SecurityClass Name="DirectoryServicesPermission" Description="System.DirectoryServices.DirectoryServicesPermission,

System.DirectoryServices, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
    b. Add the following permission to the ASP.NET permission set: <IPermission class="DirectoryServicesPermission" version="1" Unrestricted="true" />

12) ODBC datasources. This functionality can be enabled by making the following changes to the web_hightrust.config policy file:
    a. Add the following security class to the list of security classes: <SecurityClass Name="OdbcPermission" Description="System.Data.Odbc.OdbcPermission, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
    b. Add the following permission to the ASP.NET permission set: <IPermission class="OdbcPermission" version="1" Unrestricted="true"/>

3.11.3 Medium Trust Mode

Using the predefined Medium trust level, the following configuration changes must be made for BD.NET to function:

1) In web.config, the attribute requirePermission="false" must be added to the Blue-Dragon section element under the <configSections> element.
2) In the web_mediumtrust.config policy file, change the EnvironmentPermission attribute Read="TEMP;TMP;USERNAME;OS;COMPUTERNAME" to Unrestricted="true".

The following BD.NET functionality will not work:

1) All of the functionality listed for the High trust level.
2) CFHTTP. This tag can be enabled by changing the WebPermission ConnectAccess URI value from “$OriginHost$” to “.*” in the web_mediumtrust.config policy file.
3) CFFTP. This tag can be enabled by adding a SocketPermission with a ConnectAccess for the FTP server in the web_mediumtrust.config policy file. Here’s an example: <IPermission class="SocketPermission" version="1"><ConnectAccess><ENDPOINT host="<ftp server hostname or ip address>" transport="Tcp" port="-1"/></ConnectAccess></IPermission>.
4) CFPOP. This tag can be enabled by adding a SocketPermission with a ConnectAccess for the mail server in the web_mediumtrust.config policy file. Here’s an example: <IPermission class="SocketPermission" version="1"><ConnectAccess><ENDPOINT host="<mail server hostname or ip address>" transport="Tcp" port="-1"/></ConnectAccess></IPermission>.
5) CFIMAP. This tag can be enabled by adding a SocketPermission with a ConnectAccess for the mail server in the web_mediumtrust.config policy file. Here’s an example: <IPermission class="SocketPermission" version="1"><ConnectAccess><ENDPOINT host="<mail server hostname or ip address>" transport="Tcp" port="-1"/></ConnectAccess></IPermission>.
6) CFCOLLECTION. This tag can be enabled by adding the files and directories it needs to access to the FileIOPermission attribute in the web_mediumtrust.config policy file.

7) CFDIRECTORY. This tag can be enabled by adding the files and directories it needs to access to the FileIOPermission attribute in the web_mediumtrust.config policy file.
8) CFIMAGE. This tag can be enabled by adding the files and directories it needs to access to the FileIOPermission attribute in the web_mediumtrust.config policy file.
9) CFZIP. This tag can be enabled by adding the files and directories it needs to access to the FileIOPermission attribute in the web_mediumtrust.config policy file.

3.11.4 Low Trust Mode

Using the predefined Low trust level, the following configuration changes must be made for BD.NET to function:

1) All of the configuration changes listed for the Medium trust level.
2) In the web_lowtrust.config policy file, add Write and Append permissions to the FileIOPermission element.

The following BD.NET functionality will not work:

1) All of the functionality listed for the Medium trust level.
2) In development mode BD.NET will only accept requests that use 127.0.0.1 for the IP address. The local IP address can only be used if DnsPermission is added to the web_lowtrust.config policy file.
3) SQL Server. BD.NET can only connect to SQL Server if SqlClientPermission is added to the web_lowtrust.config policy file.
4) Setting of ASP.NET script timeout. This functionality can be enabled by changing the AspNetHostingPermission level to Medium in the web_lowtrust.config policy file.
5) Enabling ASP.NET debugging in web.config or in an ASP.NET page. This functionality can be enabled by changing the AspNetHostingPermission level to Medium in the web_lowtrust.config policy file.

3.11.5 Minimal Trust Mode

Using the predefined Minimal trust level, the following configuration changes must be made for BD.NET to function:

1) All of the configuration changes listed for the Low trust level.
2) In the web_minimaltrust.config policy file, change the AspNetHostingPermission level to Low.

The following BD.NET functionality will not work:

1) All of the functionality listed for the Low trust level.

3.12 MySQL 5.0, including stored procedures

All BlueDragon 7.0 editions, including BlueDragon.NET, fully support MySQL 5.0, including stored procedures, on Windows and UNIX/Linux.

3.13 Miscellaneous CFMX 7 Compatible Enhancements

CFMX 7 introduced a number of new tag and functions, and modifications to existing tags and functions that are not described in the preceding sections. Nearly all of the new and modified tags and functions are supported by BlueDragon 7.0 as listed in the following sections.

3.13.1 New Tags

The following new CFMX 7 tags, others than those discussed in the preceding sections, are implemented in BlueDragon 7.0:

CFTIMER

3.13.2 Modified Tags

The following CFMX 7 tag modifications are implemented in BlueDragon 7.0:

Tag	Changes
CFARGUMENT	Added value “xml” to TYPE attribute
CFCOLLECTION	Added categories attribute and categorylist action. Added CATEGORIES, SIZE, DOCCOUNT, and LASTMODIFIED to list of variables returned by the list action.
CFCOMPONENT	Added support for publishing document-literal style web services. Added the style, namespace, serviceportname, porttypename, wsdlfile, and bindingname attributes. Extended functionality for the hint and displayname attributes when publishing document-literal style web services.
CFCONTENT	Added the VARIABLE attribute (equivalent to the OUTPUT attribute supported by BD 6.2).
CFDIRECTORY	Added the recurse attribute and directory result-set column.
CFDUMP	Added the TOP attribute.

CFFILE	Added the RESULT attribute.
CFFTP	Added the RESULT attribute.
CFFUNCTION	Added the value “xml” to the RETURNTYPE attribute.
CFHTTP	Added the RESULT attribute.
CFINDEX	Added the status, custom3, custom4, category, and categoryTree attributes.
CFINVOKE	Added the SERVICEPORT attribute.
CFINVOKEARGUMENT	Added the OMIT attribute.
CFLDAP	Added the RETURNASBINARY attribute.
CFLOOP	Added the ability to loop over data/time ranges.
CFPARAM	Added min, max, and pattern attributes. Added creditcard, email, eurodate, float, integer, range, regex, regular_expression, ssn, social_security_number, time, URL, USdate, and zipcode attributes of the type attribute.
CFQUERY	Added the result attribute for specifying an alternate name for the structure that holds the result variables. Added result variables for the SQL statement executed (sql), the number of records returned (recordcount), whether the query was cached (cached), an array of cfqueryparam values (sqlparameters), and the list of columns in the returned query (columnlist).
CFSEARCH	Added category, categoryTree, status, suggestions, contextPassages, contextBytes, contextHighlightBegin, contextHighlightEnd, and previousCriteria attributes. Added author, category, categoryTree, context, rank, size, recordsSearched, and type result columns. Added information on the status structure and its associated keys. Set exception types to "searchengine".
CFSTOREDPROC	Added the RESULT attribute.

CFXML	Added support for using an XML declaration at the start of the text.

3.13.3 New Functions

The following new CFMX 7 functions are implemented in BlueDragon 7.0:

AddSOAPRequestHeader
AddSOAPResponseHeader
BinaryDecode
BinaryEncode
CharetDecode
CharsetEncode
GenerateSecretKey
GetContextRoot
GetLocaleDisplayName
GetSOAPRequest
GetSOAPRequestHeader
GetSOAPResponse
GetSOAPResponseHeader
IsSOAPRequest
IsValid
IsXml
IsXmlAttribute
IsXmlNode
ToScript
XmlGetNodeType
XmlValidate

3.13.4 Modified Functions

The following CFMX 7 function modifications are implemented in BlueDragon 7.0:

Function	Changes
CreateObject	Added the PORTNAME parameter (web services objects)
DayOfWeekAsString	The returned string is now in the language of the current locale.
Decrypt	Added the algorithm and encoding parameters.
Encrypt	Added the algorithm and encoding parameters.
GetLocale	Added support for all Java locales and locale names.
SetLocale	Added support for all Java locales and locale names.
GetMetaData	Added support for getting query object metadata.
Hash	Added the algorithm and encoding parameters
QueryAddColumn	Added the datatype parameter
QueryNew	Added columntypelist parameter
Rand	Added the algorithm parameter
Randomize	Added the algorithm parameter
XmlElemNew	Added the namespace parameter.
XmlParse	Added the validator parameter, support for file-names and URLs in the xmlText parameter, and support for relative URLs and path names.
XmlSearch	Added support for attribute searches
XmlTransform	Added the parameters parameter and the ability to use a file for the XSL

3.14 Performance Enhancements

A number of significant performance enhancements were implemented in BlueDragon 7.0. The impact of these performance enhancements will vary based on the specifics of your web applications.

3.15 Bug Fixes

A list of bug fixes in BlueDragon 7.0 can be found in the online bug tracking database:

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