Contents

CONTENTS .......................................................................................................... II

1 INTRODUCTION .................................................................................................. 1
  1.1 ColdFusion 5 Compatibility ........................................................................... 1
  1.2 ColdFusion MX Compatibility ....................................................................... 1
  1.3 Technical Support ......................................................................................... 1

2 VARIABLES ....................................................................................................... 2
  2.1 Variable Names ............................................................................................ 2
  2.2 URL and FORM Variables ............................................................................ 2
  2.3 SERVER Variables ....................................................................................... 2
  2.4 CALLER Prefix ............................................................................................ 2

3 CFML TAGS ....................................................................................................... 3
  3.1 Tag Formatting ............................................................................................ 3
  3.2 Unsupported Tags ......................................................................................... 3
  3.3 Supported with Limitations ......................................................................... 4
    3.3.1 CFAPPLICATION .................................................................................. 4
    3.3.2 CFGRAPH ............................................................................................ 4
    3.3.3 CFI F ..................................................................................................... 4
    3.3.4 CFINSERT ........................................................................................... 4
    3.3.5 CFOBJECT .......................................................................................... 4
    3.3.6 CFPARAM .......................................................................................... 5
    3.3.7 CFQUERY .......................................................................................... 5
    3.3.8 CFREGISTRY ....................................................................................... 6
    3.3.9 CFSET ................................................................................................. 6
    3.3.10 CFSTOREDPROC ............................................................................. 7
    3.3.11 CFTREE ............................................................................................ 7
    3.3.12 CFTREEITEM .................................................................................. 7
    3.3.13 CFUPDATE ....................................................................................... 7
    3.3.14 CFWDDX ......................................................................................... 7
  3.4 Enhancements ............................................................................................... 8
    3.4.1 CFBASE .............................................................................................. 8
    3.4.2 CFDUMP ............................................................................................. 8
    3.4.3 CFFORWARD ...................................................................................... 8
    3.4.4 CFIMAGE .......................................................................................... 8
    3.4.5 CFIMAP .............................................................................................. 10
    3.4.6 CFINCLUDE ....................................................................................... 15
    3.4.7 CFLOCK ............................................................................................ 15
    3.4.8 CFMAIL ............................................................................................. 15
    3.4.9 CFPAUSE ........................................................................................... 16
    3.4.10 CFQUERY (Caching) ........................................................................ 16
    3.4.11 CFSET (Multi-dimensional arrays) ...................................................... 16
    3.4.12 URIDIRECTORY Attribute (FILE Attribute Modifier) ....................... 17

4 CFML FUNCTIONS ........................................................................................... 17
4.1 Unsupported Functions ............................................................................................................. 17
  4.1.1 ParameterExists ............................................................................................................... 17

4.2 Supported with Limitations ............................................................................................... 18
  4.2.1 CreateObject................................................................................................................... 18
  4.2.2 IsDefined ....................................................................................................................... 18
  4.2.3 ParagraphFormat ........................................................................................................... 18

4.3 Enhancements ................................................................................................................... 18
  4.3.1 URLSessionFormat .......................................................................................................... 18

5 CFX TAGS ......................................................................................................................... 19
BlueDragon 3.0 CFML Compatibility and Reference Guide

1 Introduction

The ColdFusion Markup Language (CFML) tag and function descriptions in this document apply to all BlueDragon Server, BlueDragon Server JX, BlueDragon for J2EE, and BlueDragon for .NET product editions unless explicitly noted otherwise. This document describes BlueDragon 3.0 Service Pack 1.

1.1 ColdFusion 5 Compatibility

The BlueDragon implementation of the CFML scripting language is highly compliant with the Macromedia ColdFusion Server 5 (CF5) implementation. This document describes differences in syntax and semantics between the two implementations, including BlueDragon enhancements that are not supported by CF5.

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

Core ColdFusion 5
by Eben Hewitt
http://www.corecoldfusion.com/index.cfm?go=books&bk=core

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

Macromedia’s CFML documentation for CF5 can be found on their web site:

http://www.macromedia.com/support/coldfusion/documentation.html

1.2 ColdFusion MX Compatibility

In general, BlueDragon does not yet implement new features introduced by Macromedia ColdFusion MX (CFMX), such as ColdFusion Components (CFC) or XML processing via the CFXML tag. Development is under way to bring BlueDragon up to full compliance with CFMX in a future release.

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:

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

http://www.newatlanta.com/support/bluedragon/index.jsp

2 Variables

2.1 Variable Names

In CF5, a variable name must start with a letter and can only contain letters, numbers and the underscore (_) character. In BlueDragon, a variable name may additionally contain the dollar sign ($) character and a variable name may start with an underscore, dollar sign, or letter. Also, in BlueDragon, a variable name may not contain the period or “dot” (.) character, other than to reference variable scopes or structures; see further discussion of the CFSET tag, below.

2.2 URL and FORM Variables

In CF5, the URL scope is used to store variables that appear in the query string portion of the URL; the FORM scope is used to store variables that are submitted as part of an HTML form in an HTTP POST request.

In BlueDragon, all request parameters, whether appearing in the URL query string or submitted via an HTML form POST are stored in the FORM scope. The URL scope is simply an alias to the FORM scope; the URL and FORM prefixes may be used interchangeably when referencing variables.

2.3 SERVER Variables

For BlueDragon, the variable Server.ColdFusion.ProductName returns the value “BlueDragon”.

2.4 CALLER Prefix

When running within CFML custom tags, both BlueDragon and CF5 support the CALLER prefix to provide access to variables in the VARIABLES scope of the page that called the custom tag. For example, both support the following (where the variable parentLocal is defined in the VARIABLES scope of the calling page):

<CFOUTPUT>#CALLER.parentLocal#</CFOUTPUT>

Also, while running within CFML custom tags, both BlueDragon and CF5 support access directly to scopes other than the CALLER scope, such as:

<CFOUTPUT>#FORM.requestParm#</CFOUTPUT>

However, CF5 allows access to scopes other than the CALLER scope using the CALLER prefix, but BlueDragon does not allow such redundant use of the CALLER prefix. For ex-
ample, the following is supported in CF5 and is equivalent to the preceding example, but generates an “undefined variable” runtime error on BlueDragon:

<!-- CFOUTPUT:#CALLER.FORM.requestParm# -->

Furthermore, CF5 will search other scopes when the CALLER prefix is specified but the variable does not exist within the caller’s VARIABLES scope, whereas BlueDragon does not search other scopes. For example, the following is supported in CF5 and is equivalent to the preceding two examples (where requestParm exists within the FORM scope), but generates an “undefined variable” runtime error on BlueDragon:

<!-- CFOUTPUT:#CALLER.requestParm# -->

In BlueDragon, the CALLER prefix can only be used to access variables in the caller’s VARIABLES scope; it cannot be used to refer to variables in any other scope.

3 CFML Tags

3.1 Tag Formatting

BlueDragon expects to find the beginning of a tag name immediately following the opening left angle bracket (<) that signifies the start of an HTML or CFML tag. BlueDragon does not allow any whitespace between the opening left angle bracket and the tag name, where CF5 does. For example, the following will be processed properly by CF5 but will not be recognized as a CFML tag by BlueDragon, which will ignore it:

<CFOUTPUT>Today is #DayOfWeek(Now())#</CFOUTPUT>

3.2 Unsupported Tags

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

<table>
<thead>
<tr>
<th>Client-side Java</th>
<th>Extensibility</th>
<th>Web Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfapplet</td>
<td>cfcollection</td>
<td>cfassociate</td>
</tr>
<tr>
<td>cffgrid</td>
<td>cfindex</td>
<td>cfauthenticate*</td>
</tr>
<tr>
<td>cffgridcolumn</td>
<td>cfreport</td>
<td>cfimpersonate*</td>
</tr>
<tr>
<td>cffgridrow</td>
<td>cfsearch</td>
<td></td>
</tr>
<tr>
<td>cffgridupdate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*obsolete in Macromedia CFMX
3.3 **Supported with Limitations**

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

### 3.3.1 CFAPPLICATION

In BlueDragon, the `name` attribute for `CFAPPLICATION` is always required; this attribute is optional in CF5.

### 3.3.2 CFGRAPH

BlueDragon only supports creating graphs in JPEG format; it does not support GIF or Flash formats. Therefore, the only valid value for the `fileFormat` attribute is “JPG”; if any other value is assigned to this attribute an error will be generated by BlueDragon.

### 3.3.3 CFIF

BlueDragon requires that at least one whitespace character appear between the end of the `CFIF` tag name and the opening parenthesis of the tag expression, where CF5 does not. For example, the following is valid on CF5 but will generate an error on BlueDragon:

```xml
<CFIF(DayOfWeek(Now()) IS 1)>
Today is Sunday!
</CFIF>
```

For the above code to work properly on BlueDragon, a blank space must appear between the `CFIF` tag name and the opening parenthesis of the tag expression:

```xml
<CFIF (DayOfWeek(Now()) IS 1)>
Today is Sunday!
</CFIF>
```

### 3.3.4 CFINSERT

BlueDragon does not support the following optional `CFINSERT` attributes:

- `connectString`
- `dbName`
- `dbServer`
- `dbType`
- `provider`
- `providerDSN`

BlueDragon always requires the `dataSource` attribute for `CFINSERT` tags.

### 3.3.5 CFOBJECT

BlueDragon only supports “java” as the value for the `type` attribute; it does not support the values “com” or “corba”.
3.3.6 CFPARAM

3.3.6.1 Creating Cookies
In BlueDragon, it is not possible to create browser cookies using the CFPARAM tag; use the CFCOOKIE tag instead. For example, the following tag will generate an error in BlueDragon:

```cfparam
<cfparam NAME="COOKIE.VisitStart" TYPE="date" DEFAULT="#Now()#">
```

Instead of the above, the following equivalent code should be used to create a cookie if the cookie does not already exist:

```cfif
<cfif IsDefined("COOKIE.VisitStart") EQ "No">
  <cfcookie NAME="VisitStart" VALUE="#Now()#">
</cfif>
```

Using CFCOOKIE to create or modify cookies is generally preferred to using CFPARAM because the CFCOOKIE tag allows setting the cookie attributes EXPIRES, DOMAIN, PATH, and SECURE, which cannot be set using the CFPARAM tag.

3.3.6.2 Creating CGI Variables
In BlueDragon, it is not possible to create CGI variables using the CFPARAM tag (there is no way to create or modify CGI variables in BlueDragon).

3.3.7 CFQUERY

3.3.7.1 Supported Tag Attributes
BlueDragon does not support the following optional CFQUERY attributes:

- blockfactor
- connectString
- dbName
- dbServer
- debug
- provider
- providerDSN
- timeout

The only value supported by BlueDragon for the dbType attribute is “query”; all other values are ignored if the dataSource attribute is also specified. BlueDragon requires that CFQUERY tags have either the dbType="query" attribute or the dataSource attribute specified.

3.3.7.2 Escaping Single Quote Characters
BlueDragon does not “escape” single-quote characters within CFML variables when those variables are used to create SQL statements within CFQUERY tags. For example, the following will cause a database error on BlueDragon but not CF5:

```cfparam
<cfparam NAME="JOB." TYPE="string" DEFAULT="#" EQUAleston>
```
<CFSET EmployeeName="O'Neil">
<CFQUERY NAME="employees" DATASOURCE="MyCompany">
SELECT * FROM Employees
WHERE LastName = '#EmployeeName#'
</CFQUERY>

For this to work properly on BlueDragon, you must use the CFQUERYPARAM tag (this use of the CFQUERYPARAM tag is also valid on CF5):

<CFSET EmployeeName="O'Neil">
<CFQUERY NAME="employees" DATASOURCE="MyCompany">
SELECT * FROM Employees
WHERE LastName = <CFQUERYPARAM VALUE="#EmployeeName#">
</CFQUERY>

For security reasons, it is recommended to always use the CFQUERYPARAM tag within CFQUERY tags to protect your databases from access by unauthorized users.

### 3.3.8 CFREGISTRY

BlueDragon simulates the Windows registry on all operating systems, including Windows. Therefore, it is not possible to read, write, or delete “registry” entries other than those created using the CFREGISTRY tag.

### 3.3.9 CFSET

#### 3.3.9.1 Creating and Modifying Cookies

In BlueDragon, it is not possible to create or modify the values of browser cookies using the CFSET tag; use the CFCOOKIE tag instead. For example, the following tag will generate an error in BlueDragon:

<CFSET COOKIE.VisitStart=Now()>  

Instead, the following equivalent tag should be used to create or modify cookies:

<CFCOOKIE NAME="VisitStart" VALUE="#Now()#">

Using CFCOOKIE to create or modify cookies is generally preferred to using CFSET because the CFCOOKIE tag allows setting the cookie attributes EXPIRES, DOMAIN, PATH, and SECURE, which cannot be set using the CFSET tag.

#### 3.3.9.2 Creating and Modifying CGI Variables

In BlueDragon, it is not possible to create or modify the values of CGI variables using the CFSET tag (there is no way to create or modify CGI variables in BlueDragon).

#### 3.3.9.3 Periods in Variable Names

In BlueDragon, a variable name may not contain the period (.) character, other than to reference variable scopes or structures. Consider the following:

<CFSET a.b="some value">

If a variable named “a” does not exist, CF5 creates a variable named “a.b” whereas BlueDragon creates a structure “a” that contains variable “b”.  

---

BlueDragon 3.0 CFML Compatibility and Reference Guide
3.3.10 CFSTOREDPROC
BlueDragon does not support the following optional CFSTOREDPROC attributes:

- connectString
- dbName
- dbServer
- dbType
- provider
- providerDSN

BlueDragon always requires the dataSource attribute for CFSTOREDPROC tags.

3.3.11 CFTREE
BlueDragon does not support the following optional CFTREE attributes:

- completePath
- delimiter
- onValidate

3.3.12 CFTREEITEM
BlueDragon does not support the following optional CFTREEITEM attributes:

- img
- imgOpen

3.3.13 CFUPDATE
BlueDragon does not support the following optional CFUPDATE attributes:

- connectString
- dbName
- dbServer
- dbType
- provider
- providerDSN

BlueDragon always requires the dataSource attribute for CFUPDATE tags.

3.3.14 CFWDDX
The following limitations exist in the BlueDragon implementation of the CFWDDX tag relative to the CF5 implementation:

1. BlueDragon cannot deserialize binary data from WDDX to CFML.
2. The USETIMEZONEINFO attribute is not supported by BlueDragon (defaults to “Yes”).
3. The VALIDATE attribute is not supported by BlueDragon.
4. BlueDragon uses a different notation than CF5 when JavaScript objects are created:

   - CF5 notation: MyStock[“price”] = “66.25”;
   - BlueDragon notation: MyStock.price = “66.25”;

---
BlueDragon 3.0 CFML Compatibility and Reference Guide 7
3.4 **Enhancements**

This section lists enhanced and new CFML tags in BlueDragon.

### 3.4.1 CFBASE

*CFBASE* is a new CFML tag introduced by BlueDragon that is primarily intended for use in *BlueDragon for J2EE Servers*. The *CFBASE* tag can be used to create an absolute URL that serves as the base for resolving relative URLs within a CFML page (such as in *IMG* tags). The absolute URL created by the *CFBASE* tag includes the J2EE web application context path. See the document *Deploying CFML on J2EE Servers Using BlueDragon* for a detailed discussion of *CFBASE*.

### 3.4.2 CFDUMP

The *CFDUMP* tag *var* attribute is required in CF5, but is optional in BlueDragon; if omitted, variables in all scopes are displayed:

```html
<CFDUMP VAR="#SESSION#"> <!--- display SESSION variables --->
<CFDUMP> <!--- display variables in all scopes --->
```

### 3.4.3 CFFORWARD

*CFFORWARD* is a new tag introduced by BlueDragon that allows you to do a “server-side redirect” to another CFML page, a Java server, or a JavaServer Page (JSP). In a “client-side redirect,” which is done using the *CFLOCATION* tag, a response is sent to the browser telling it to send in a new request for a specified URL. In contrast, *CFFORWARD* processing is handled completely on the server.

The advantages of *CFFORWARD* over *CFLOCATION* are:

- There is no need for extra messaging between the server and browser.
- Variables in the *URL*, *FORM*, and *REQUEST* scopes are available to the target of the *CFFORWARD* tag.

*CFFORWARD* has a single attribute, *PAGE*, which specifies the target page as a relative path from the location of the current CFML page:

```html
<CFFORWARD PAGE="nextpage.cfm">
<CFFORWARD PAGE="nextpage.jsp">

Like *CFLOCATION*, processing of the current page is terminated as soon as the *CFFORWARD* tag is processed.

### 3.4.4 CFIMAGE

*CFIMAGE* is a new tag introduced by BlueDragon that allows you to modify an existing GIF or JPEG image file to produce a new image file that is resized and/or has a text label added to the image. Variables returned by this tag provide information about the new image file.
The following table lists the **CFIMAGE** tag attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRCFILE</td>
<td>Required. The file name of the source image file that is to be modified. Can be either a full physical path or a relative path (see the URIDIRECTORY attribute).</td>
</tr>
<tr>
<td>DESTFILE</td>
<td>Required if ACTION=EDIT, Optional if ACTION=INFO. The file name of the new image file to be created by the CFIMAGE tag. Can be either a full physical path or a relative path (see the URIDIRECTORY attribute).</td>
</tr>
<tr>
<td>ACTION</td>
<td>Optional. The action to be taken by the CFIMAGE tag. The value INFO populates the CFIMAGE variables with information about the image file specified by the SRCFILE attribute without modifying the image. The value of EDIT creates a new image file by resizing and/or adding a text label to the source image file. Defaults to EDIT.</td>
</tr>
<tr>
<td>TYPE</td>
<td>Optional. The image file type, either GIF or JPEG. If this attribute is not specified, the CFIMAGE tag attempts to determine the image type based on the file name extension.</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Optional. The width of the new image, can be specified either in pixels or as a percentage of the source image width. Defaults to “100%”.</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Optional. The height of the new image, can be specified either in pixels or as a percentage of the source image height. Defaults to “100%”.</td>
</tr>
<tr>
<td>FONTSIZE</td>
<td>Optional. An integer value that specified the font size of the text label to be added to the image. Defaults to 12.</td>
</tr>
<tr>
<td>FONTCOLOR</td>
<td>Optional. Specifies the font color of the text label to be added to the image. Accepts any value that is valid for use in the FONT tag. Defaults to “black”.</td>
</tr>
<tr>
<td>TEXT</td>
<td>Optional. The text label to add to the image.</td>
</tr>
<tr>
<td>POSITION</td>
<td>Optional. The position of the text label to add to the image; valid valued are “north” and “south”. Defaults to “south”.</td>
</tr>
<tr>
<td>NAMECONFLICT</td>
<td>Optional. Indicates the behavior of the CFIMAGE tag when the file specified by DESTFILE already exists. Valid values are ERROR, which generates a runtime error; SKIP, which causes the CFIMAGE tag to do nothing without generating an error; OVERWRITE, to overwrite the existing image; and, MAKEUNIQUE, which causes CFIMAGE to create a new unique file name for the new image file. Defaults to ERROR.</td>
</tr>
<tr>
<td>URIDIRECTORY</td>
<td>Optional. If YES, relative paths specified in SRCFILE and DESTFILE are calculated from the web server document root directory. If NO, relative paths are calculated as relative to the current file. Defaults to NO.</td>
</tr>
</tbody>
</table>

The following table lists the variables returned by the **CFIMAGE** tag.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFIMAGE.SUCCESS</td>
<td>Contains the value TRUE or FALSE to indicate whether image processing was successful.</td>
</tr>
<tr>
<td>CFIMAGE.ERRORTEXT</td>
<td>If processing was unsuccessful, contains a text message describing the error.</td>
</tr>
<tr>
<td>CFIMAGE.WIDTH</td>
<td>For ACTION=EDIT, the width in pixels of the new image. For ACTION=INFO, the width in pixels of the image.</td>
</tr>
<tr>
<td>CFIMAGE.HEIGHT</td>
<td>For ACTION=EDIT, the height in pixels of the new image. For ACTION=INFO, the height in pixels of the image.</td>
</tr>
<tr>
<td>CFIMAGE.PATH</td>
<td>The full physical path to the image.</td>
</tr>
<tr>
<td>CFIMAGE.NAME</td>
<td>The name of the new image file.</td>
</tr>
<tr>
<td>CFIMAGE.FILESIZE</td>
<td>The size in bytes of the new image file.</td>
</tr>
</tbody>
</table>
The following example displays two images – the original image “picture.gif”, and the processed image “newPicture.gif”.

```html
<cfimage action="edit"
  srcfile="picture.gif"
  destfile="newPicture.gif"
  uridirectory="yes"
  text="Copyright 2002"
  width="50%"
  height="50%"
  fontsize=20
  fontcolour="violet"
  position="SOUTH"
  nameconflict="overwrite">
  <img src="picture.gif">
  <img src="newPic.gif">
</cfimage>
```

The following example displays information about an existing image file named “picture.jpg”.

```html
<cfimage action="info" srcfile="picture.jpg">
<cfoutput>
Success : #cfimage.success# <BR>
Dimensions : #cfimage.width# x #cfimage.height# <BR>
Path : #cfimage.filepath# <BR>
Name : #cfimage.filename# <BR>
Size (bytes) : #cfimage.filesize# <BR>
Error message : #cfimage.errortext# <BR>
</cfoutput>
```

### 3.4.5 CFIMAP

The `CFIMAP` tag allows you to interact with both IMAP and POP mail servers (CFIMAP may be used instead of `CFPOP` to interact with POP mail servers). Generally, the sequence of steps to interact with a mail server is:

1. Open a connection to the mail server (OPEN action).
2. Get a list of folders from the mail server (LISTFOLDERS action).
3. Get a list of messages within a specific folder (LISTMAIL action).
4. Perform actions with specific messages (READMAIL, MARKREAD, DELTEMAIL, and MOVEMAIL actions).
5. Perform actions with folders (DELETEFOLDER and RENAMEFOLDER actions).
6. Close the connection (CLOSE action).

Sending email messages is done using the `CFMAIL` tag, not CFIMAP. However, BlueDragon has added two new attributes to the `CFMAIL` tag to allow you to store sent mail in an IMAP server folder. See the section on the `CFMAIL` tag for details.
3.4.5.1 Opening a Connection

Before performing actions such as reading mail, you must first open a connection with the IMAP or POP server. Specify a value of OPEN for the ACTION attribute. The name specified for the CONNECTION attribute will be used to refer to this connection when performing subsequent actions with the IMAP or POP server, such as reading mail.

```xml
<CFIMAP ACTION="OPEN"
SERVICE="POP3 or IMAP"
CONNECTION="name"
SERVER="mail.yourdomain.com"
USERNAME="username"
PASSWORD="password">
```

Two variables are always returned by the CFIMAP tag:

- IMAP.SUCCEEDED – “true” or “false” depending on whether the previous action succeeded
- IMAP.ERRORTEXT – an error message, if the previous action failed

3.4.5.2 Closing a Connection

An IMAP or POP server connection can be closed by specifying ACTION="CLOSE" and the name of the connection:

```xml
<CFIMAP ACTION="CLOSE"
CONNECTION="name">
```

After closing a connection, any attempts to use the connection will generate an error.

3.4.5.3 Listing Mailbox Folders

Use ACTION="LISTALLFOLDERS" to get a list of folders on the IMAP or POP server:

```xml
<CFIMAP ACTION="LISTALLFOLDERS"
CONNECTION="name"
NAME="queryname">
```

The folder list is returned in a Query structure with the name you specified in the NAME attribute. The fields of the Query structure are:

- FULLNAME – the full path to the folder (used to retrieve folder message info)
- NAME – shortcut name to the folder
- TOTALMESSAGES – total unread messages this folder is holding
- UNREAD – total unread messages in this folder
- NEW – total new messages in this folder

The FULLNAME field is used for making subsequent calls to folders with other CFIMAP action parameters.
3.4.5.4 *Listing Mail Messages*

You can retrieve high-level information about the messages within a folder by specifying `ACTION="LISTMAIL"`; this action does not retrieve the message bodies. To read a message body you must first get an email message ID using the `LISTMAIL` action and then specify the message ID in the `READMAIL` action as described in the next section.

The `FOLDER` attribute must contain the name of a folder as contained in the `FULLNAME` field of the Query structure returned by `ACTION="LISTALLFOLDERS"`.

```
<CFIMAP  ACTION="LISTMAIL"
  CONNECTION="name"
  FOLDER="fullname"
  NAME="queryname">
```

The message information is returned in a Query structure with the name you specified in the `NAME` attribute. The fields of this Query structure are:

- **SUBJECT** – subject line of the mail message
- **ID** – unique ID of this mail message (used to retrieve the message body)
- **RXDDATE** – the date this mail message was received
- **SENTDATE** – the date this mail message was sent
- **FROM** – address structure (see below)
- **TO** – array of address structures (see below)
- **CC** – array of address structures (see below)
- **BCC** – array of address structures (see below)
- **SIZE** – size in bytes of this mail message
- **LINES** – number of lines of this mail message
- **ANSWERED** – boolean flag if this mail message has been answered
- **DELETED** – boolean flag if this mail message has been deleted
- **DRAFT** – boolean flag if this mail message is an unsent draft
- **FLAGGED** – boolean flag if this email has been flagged
- **RECENT** – boolean flag if this email is recent
- **SEEN** – boolean flag if this email has been seen (read)

Internet email addresses are stored as structures with two fields:

- **NAME** – name of the person
- **EMAIL** – email address of the person

The **TO**, **CC**, and **BCC** fields contain arrays of these structures.
3.4.5.5 Reading a Mail Message

You can read a specific email message by specifying ACTION="READMAIL", the folder name, and the email message ID as returned by the LISTMAIL action:

```cftml
<CFIMAP ACTION="READMAIL"
    CONNECTION="name"
    FOLDER="foldername"
    MESSAGEID="messageid"
    ATTACHMENTSURI="uritofolder"
    NAME="messagename">
```

This action will retrieve the given message and fill in a structure variable containing information regarding the retrieved email message. In addition to this, should the message have any attachments, you specify the URI of the folder you wish the email attachment to be stored in. Note this is a URI and not a real directory. The fields of the returned structure are:

- **SUBJECT** – subject of the email
- **ID** – unique ID to this mail
- **RXDDATE** – the date this mail was received
- **SENTDATE** – the date this email was sent
- **FROM** – address structure (see below)
- **TO** – array of Address Structures (see below)
- **CC** – array of Address Structures (see below)
- **BCC** – array of Address Structures (see below)
- **SIZE** – size in bytes of this email
- **LINES** – number of lines of this email
- **ANSWERED** – boolean flag if this email has been answered
- **DELETED** – boolean flag if this email has been deleted
- **DRAFT** – boolean flag if this email is a draft
- **FLAGGED** – boolean flag if this email has been flagged
- **RECENT** – boolean flag if this email is recent
- **SEEN** – boolean flag if this email has been seen
- **BODY** – array of Body structures [see below]

The body of the email is treated with some consideration. Due to the various properties a MIME type email message can have, each element in the array is effectively the MIME part that was transmitted with the email.

- **MIMETYPE** – the MIME type of this part
This action will not overwrite any existing files; instead, it will create a unique name for it.

### 3.4.5.6 Marking Mail Messages as “Read”

You can mark messages as having been read by specifying `ACTION="MARKREAD"`, a folder name, and a list of message IDs:

```xml
<CFIMAP ACTION="MARKREAD"
   CONNECTION="name"
   FOLDER="toplevelfoldername"
   MESSAGELIST="list of IDs">
```

The message list is either a single message ID or a comma-separated list of IDs.

### 3.4.5.7 Deleting Mail Messages

You can delete messages by specifying `ACTION="DELETEMAIL"`, a folder name, and a list of message IDs:

```xml
<CFIMAP ACTION="DELETEMAIL"
   CONNECTION="name"
   FOLDER="toplevelfoldername"
   MESSAGELIST="list of IDs">
```

The message list is either a single message ID or a comma-separated list of IDs.

### 3.4.5.8 Moving Mail Messages between Folders

You can move a list of messages from one mail server folder to another by specifying `ACTION="MOVEMAIL"`:

```xml
<CFIMAP ACTION="MOVEMAIL"
   CONNECTION="name"
   FOLDER="toplevelfoldername"
   DESTFOLDER="toplevelfoldername"
   MESSAGELIST="list of IDs">
```

The message list is either a single message ID or a comma-separated list of IDs.
3.4.5.9 Deleting a Folder

Specifying `ACTION="DELETEFOLDER"` will delete a folder from the mail server, including all of its contents (mail messages):

```
<CFIMAP ACTION="DELETEFOLDER"
    CONNECTION="name"
    FOLDER="fullfoldername">
```

The folder name is the complete path to the folder. This is a very powerful action and should be used with extreme care, as it can remove all messages and folders from the mail server.

3.4.5.10 Renaming a Folder

Specifying `ACTION="RENAMEFOLDER"` will rename a folder on the mail server:

```
<CFIMAP ACTION="RENAMEFOLDER"
    CONNECTION="name"
    OLDFOLDER="fullfoldername"
    NEWFOLDER="fullfoldername">
```

The folder name is the complete path to the folder.

3.4.5.11 Sending Mail Messages

Sending email messages is done using the `CFMAIL` tag, not `CFIMAP`. However, BlueDragon has added two new attributes to the `CFMAIL` tag to allow you to store sent mail in an IMAP server folder. See the section on the `CFMAIL` tag for details.

3.4.6 CFINCLUDE

BlueDragon allows you to include the output of Java servlets or JavaServer Pages (JSP) in your CFML pages via the new `PAGE` attribute to the `CFINCLUDE` tag. The `PAGE` attribute specifies the path to the JSP as a relative path from the current CFML page:

```
<CFINCLUDE PAGE="menu.jsp">
```

3.4.7 CFLOCK

BlueDragon supports the full syntax and semantics of `CFLOCK`, but does not require the use of `CFLOCK` when accessing variables in the Session, Application, and Server scopes. BlueDragon manages concurrent access to these variable scopes internally.

3.4.8 CFMAIL

BlueDragon has added two new attributes to the `CFMAIL` tag to allow you to store sent mail in an IMAP server folder. In order to use these attributes you must first open a connection to the IMAP server using the `CFIMAP` tag (see above). These two new attributes are used in conjunction with the existing `CFMAIL` attributes to send an email message and have it saved on an IMAP server:

```
<CFMAIL IMAPCONNECTION="name"
    IMAPFOLDER="fullfoldername"
    ...
```
3.4.9 CFPAUSE

CFPAUSE is a new tag introduced by BlueDragon to assist in debugging CFML pages. The CFPAUSE tag allows you to pause the execution of a page for a specified number of seconds. The interval attribute is required and must specify an integer value:

```cfml
<CFPAUSE INTERVAL="seconds to pause">
```

3.4.10 CFQUERY (Caching)

BlueDragon implements improved caching for CFQUERY tags via the new CACHENAME and ACTION attributes. The optional CACHENAME attribute can be used to assign a unique name for cached CFQUERY results:

```cfml
<CFQUERY NAME="users" DATASOURCE="mycompany" CACHENAME="usercache">
SELECT * FROM USERS
</CFQUERY>
```

In the above example, the CFQUERY results will be cached under the name “usercache” and when this query is run again the results from the cache will be used. You must specify a unique value for CACHENAME; if the same value for CACHENAME is specified for multiple CFQUERY tags, whether on the same or different CFML pages, the results in the cache will be overwritten.

The CACHWITHIN and CACHAFTER attributes as implemented by CF5 can be used in conjunction with CACHENAME.

A CFQUERY cache can be flushed using the new optional ACTION attribute:

```cfml
<CFQUERY ACTION="flushcache" CACHENAME="usercache">
```

All CFQUERY cached results can be cleared using a single tag:

```cfml
<CFQUERY ACTION="flushall">
```

A CFQUERY tag that uses the ACTION attribute to flush a cache can appear on the same or a different CFML page from the CFQUERY tag that defines the cache.

3.4.11 CFSET (Multi-dimensional arrays)

CF5 limits multi-dimensional arrays to three dimensions; BlueDragon does not impose any limit. For example the following tags are supported by BlueDragon, but will generate errors in CF5:

```cfml
<CFSET myArray=ArrayNew(8)>
<CFSET myArray[2][3][4][4][2][3][4][4]="BlueDragon">
```
3.4.12 URIDIRECTORY Attribute (FILE Attribute Modifier)

There are a number of CFML tags that manipulate the file system via the FILE attribute. In CF5, you must specify a full file system path for the FILE attribute for these tags:

- CFCACHE
- CFCONTENT
- CFDIRECTORY
- CFEXECUTE
- CFFILE
- CFHTTP
- CFFTP
- CFIMAGE
- CFLOG
- CFPOP
- CFSCHEDULE

BlueDragon adds an optional URIDIRECTORY attribute to these tags to indicate whether the FILE attribute specifies a full file system path or a URI path that is relative to the web server’s document root directory. For example, the following tags would produce the same result on Microsoft IIS:

```xml
<CFFILE ACTION="delete" FILE="C:\Inetpub\wwwroot\images\a.jpg">
<CFFILE ACTION="delete" FILE="/images/a.jpg" URIDIRECTORY="Yes">
```

Specifying FILE attributes as relative URI paths improves the portability of CFML pages by eliminating web server and operating system specific physical path specifications. Note in the above example that the first tag is not portable to a Linux, Solaris, or other UNIX operating system running Apache, but the second one is.

The optional URIDIRECTORY attribute accepts the values “Yes” and “No”; the default value is “No”.

4 CFML Functions

4.1 Unsupported Functions

4.1.1 ParameterExists

This function is not supported by BlueDragon. Use the IsDefined function instead.

Note that there is an important difference between the way parameters are passed to the ParameterExists and IsDefined functions. ParameterExists takes an unquoted variables name as its parameter, while IsDefined takes a quoted variable name. For example, the following two function calls are equivalent:

```csharp
ParameterExists( Form.Username )
IsDefined( “Form.Username” )
```
If you pass an unquoted variable name as the parameter to \texttt{IsDefined}, it will use the value of the variable as the variable name. For example, consider the following:

\begin{verbatim}
<cfparam name="Username" value="Bob">
<cfif IsDefined( Username )>
...
</cfif>
\end{verbatim}

In the above example, \texttt{IsDefined} will return “false” because it’s equivalent to:

\begin{verbatim}
<cfif IsDefined( "Bob")>
\end{verbatim}

Therefore, be careful when converting from \texttt{ParameterExists} to \texttt{IsDefined}—it’s not a simple search-and-replace!

\section*{4.2 Supported with Limitations}

\subsection*{4.2.1 CreateObject}
BlueDragon only supports \texttt{JAVA} as the first argument to \texttt{CreateObject}; it does not support \texttt{COM} or \texttt{CORBA}.

\subsection*{4.2.2 IsDefined}
In CF5, if a query fails to execute for any reason (for example, if there is a SQL syntax error that causes a \texttt{<CFQUERY> to fail}), then a call to \texttt{IsDefined} for the query variable returns \texttt{false}. In BlueDragon, the call to \texttt{IsDefined} will return \texttt{true} even if the query fails to execute; check for \texttt{RecordCount} greater than 0 instead.

\subsection*{4.2.3 ParagraphFormat}
From the CFML Reference for CF5:

\begin{quote}
“Returns \texttt{string} with converted single newline characters (\texttt{CR/LF} sequences) into spaces and double newline characters into HTML paragraph markers (\texttt{<p>}).”
\end{quote}

BlueDragon varies from this behavior in that it converts single newline characters into HTML break tags (\texttt{<br>}) instead of spaces. Double newline characters are converted into HTML paragraph markers (\texttt{<p>}) by both BlueDragon and CF5.

\section*{4.3 Enhancements}

\subsection*{4.3.1 URLSessionFormat}
The \texttt{URLSessionFormat} function can be used to automatically append session tracking information (\texttt{CFID/CFTOKEN} variables for CFML session tracking, or the \texttt{jsessionid} variable for J2EE session tracking) to a URL to support session tracking when cookies are disabled in a client browser. If browser cookies are enabled, this function does not modify the URL. Example usage:

\begin{verbatim}
<CFOUTPUT>
<FORM ACTION="#URLSessionFormat( ‘SubmitPage.cfm’ )#">
</CFOUTPUT>
\end{verbatim}
The `URLSessionFormat` function should be used instead of manually appending `CFID/CFTOKEN` values to URLs because it supports J2EE session tracking and only appends the values when required.

## 5 CFX Tags

BlueDragon does not support CFX tags implemented in C++ (BlueDragon *does* support CFX tags implemented in Java).