APPENDIX A

ColdFusion Tag Reference

ColdFusion tags are the CFML extensions to HTML. These tags are the instructions to ColdFusion to perform database queries, process results, generate output, control program flow, handle errors, send and receive e-mail, and much more.

The tags are presented here in alphabetical order and are cross-referenced to any related tags wherever appropriate.

<CFABORT>

Description: The <CFABORT> tag is used to immediately halt processing of a ColdFusion template. It optionally presents a user-defined error message. <CFABORT> attributes are listed in Table A.1.

Syntax:

```
<CFABORT SHOWERROR="Error text">
```

TABLE A.1 <CFABORT> ATTRIBUTES

Attribute	Description	Notes
SHOWERROR	Error message text	Optional

Example: The following example aborts template processing if the user has not properly logged in, as evidenced by the existence of a session variable:

```
<CFIF NOT IsDefined("Session.LoggedIn")>
  <CFABORT SHOWERROR="You are not authorized to use this function!">
</CFIF>
```

Tip

<CFABORT> can be used to safely terminate the processing of a template if an error condition occurs. For example, if your template were expecting a URL parameter to be passed, you could use the ParameterExists or IsDefined function to verify its existence and terminate the template with an appropriate error message if it did not exist.

See also: <CFEXIT>, <CFBREAK>

<CFAPPLET>

Description: <CFAPPLET> is used to embed user-supplied Java applets in CFFORM forms. Table A.2 is the complete list of attributes supported by <CFAPPLET>. In addition, you can pass your own attributes as long as they have been registered along with the applet itself.

Before you can use an applet with <CFAPPLET>, it must be registered with the ColdFusion Administrator.

<CFAPPLET> must be used within <CFFORM> and </CFFORM> tags.

Syntax:

```
<CFAPPLET ALIGN="Alignment" APPLETSOURCE="Registered Name" HEIGHT="Height"
HSPACE="Horizontal Spacing" NAME="Field Name"
 NOTSUPPORTED="Text for non Java browsers" VSPACE="Vertical Spacing"
 WIDTH="Width">
```

TABLE A.2 <CFAPPLET> ATTRIBUTES

Attribute	Description	Notes
ALIGN	Applet alignment	Optional; valid values are LEFT, RIGHT, BOTTOM, TOP, TEXTTOP, MIDDLE, ABSMIDDLE, BASELINE, and ABSBOTTOM
APPLETSOURCE	Name of the registered applet	Required
HEIGHT	Number of pixels	Optional; height of applet
HSPACE	Number of pixels	Optional; horizontal space around applet
NAME	Form field name	Required
NOTSUPPORTED	Text to display on browsers that do not support Java	Optional
VSPACE	Number of pixels	Optional; vertical space around applet
WIDTH	Number of pixels	Optional; width of applet
PARAM*n*	Parameter names	Optional; parameter names registered with the applet

Example: The following example embeds a Java spin control applet into a <CFFORM> form, passing the required attributes and two applet-specific attributes:

```
<CFFORM ACTION="process.cfm">
<CFAPPLET APPLETSOURCE="spin" NAME="quantity" MIN="1" MAX="10">
</CFFORM>
```

For more information about the ColdFusion Administrator and registering applets, see Chapter 4, "Accessing the ColdFusion Administrator." For more information about using <CFFORM> and <CFAPPLET>, see Chapter 25, "Enhancing Forms with Client-Side Java."

Note

Controls embedded with <CFAPPLET> are accessible only by users with Java-enabled browsers.

See also: <CFFORM>, <CFGRID>, <CFSLIDER>, <CFTEXTINPUT>, <CFTREE>

<CFAPPLICATION>

Description: <CFAPPLICATION> is used to define the scope of an application and to specify several aspects of the application's configuration. These include the use of session and client variables. This tag should be included in all templates that are part of the application and therefore intended for use in your Application.cfm template. <CFAPPLICATION> attributes are shown in Table A.3.

Syntax:

```
<CFAPPLICATION APPLICATIONTIMEOUT ="Timeout" CLIENTMANAGEMENT ="Yes or No"
 CLIENTSTORAGE="Storage Type" NAME="Application Name"
 SESSIONMANAGEMENT ="Yes or No" SESSIONTIMEOUT ="Timeout"
 SETCLIENTCOOKIES="Yes or No">
```

TABLE A.3 <CFAPPLICATION> ATTRIBUTES

Attribute	Description	Notes
APPLICATIONTIMEOUT	Time interval	Optional; application variable timeout; defaults to value in ColdFusion Administrator.
CLIENTMANAGEMENT	YES or NO	Optional; defaults to NO. Enable or disable client variables.
CLIENTSTORAGE	Mechanism for storage of client information	Optional; defaults to REGISTRY; other values are COOKIE and any ODBC data source name.
NAME	Name of application	Required if you are using application variables; can be up to 64 characters long.
SESSIONMANAGEMENT	YES or NO	Optional; defaults to NO. Enable or disable session variables.
SESSIONTIMEOUT	Time interval	Optional; session variable timeout; defaults to value in ColdFusion Administrator.

TABLE A.3 CONTINUED

Attribute	Description	Notes
SETDOMAINCOOKIES	YES or NO	Enables use of CFID and CFTOKEN cookies across an entire domain, rather than just a single host. Intended for use in a CF Server cluster.
SETCLIENTCOOKIES	YES or NO	Optional; enable or disable client cookies; defaults to YES.

Example: The following example is designed for use in an Application.cfm file. It names part of an application Administration and enables session variables. It then looks for the presence of a particular session variable and redirects the user to the specified directory if the session variable doesn't exist:

```
<CFAPPLICATION NAME="Administration" SESSIONMANAGEMENT="Yes"
➥SESSIONTIMEOUT="CreateTimeSpan(0,0,45,0)">
<CFIF NOT IsDefined("Session.LoggedIn")>
  <CFLOCATION URL="/Login">
</CFIF>
```

Tip

Use the CreateTimeSpan() function to create the time interval parameters required in the SESSIONTIMEOUT and APPLICATIONTIMEOUT attributes. See Appendix B, "ColdFusion Function Reference," for further details.

For more information about enabling or disabling application and session variables, as well as setting default timeout values, see Chapter 19, "Introducing the Web Application Framework." For more information about client, session, and application variables, see Chapter 20, "Working with Sessions."

See also: <CFCOOKIE>, <CFSET>

<CFASSOCIATE>

Description: The <CFASSOCIATE> tag is used to associate subtags, or child tags, with base tags. This tag can be used only inside custom tag templates. It is used in the subtag to make the subtag data available in the base tag. The subtag data is stored in a structure in the base tag. <CFASSOCIATE> attributes are shown in Table A.4.

Syntax:

```
<CFASSOCIATE BASETAG="CF_TAG" DATACOLLECTION="structure">
```

TABLE A.4 <CFASSOCIATE> ATTRIBUTES

Attribute	Description	Notes
BASETAG	Base tag name	Required; name of base tag associated with this subtag.
DATACOLLECTION	Name of the structure in the base tag to store attributes	Optional; defaults to structure of AssocAttribs is used.

Example: The following example associates a subtag with a base tag:

```
<!--- In the subtag CF_USERINFO --->
<CFPARAM NAME="Attributes.LoginName" DEFAULT="">
<CFPARAM NAME="Attributes.Password" DEFAULT="">
<CFPARAM NAME="Attributes.Privilege" DEFAULT="">
<CFASSOCIATE BASETAG="CF_CHECKUSER">
...

<!--- In the base tag, CF_CHECKUSER --->
<CFIF Len(AssocAttribs.Attributes.LoginName) EQ 0>
   <CFEXIT>
</CFIF>
```

See also: <CFMODULE>

<CFAUTHENTICATE>

Description: The <CFAUTHENTICATE> tag authenticates a user against a security context. This defines a security context for the application. The complete list of supported attributes is explained in Table A.5.

Syntax:

```
<CFAUTHENTICATE PASSWORD="password" SECURITYCONTEXT="context" USERNAME="user">
```


TABLE A.5 <CFAUTHENTICATE> ATTRIBUTES

Attribute	Description	Notes
PASSWORD	User password	Required; user password.
SECURITYCONTEXT	Context to authenticate against	Required; context must have been defined in the ColdFusion Administrator.
SETCOOKIE	YES or NO	Optional; defaults to NO. Sets cookie with authentication information, including username, remote address, security context, and HTTP user agent. Setting the cookie means you won't have to invoke this tag on every page in your application.

TABLE A.5 CONTINUED

Attribute	Description	Notes
THROWONFAILURE	YES or NO	Optional; defaults to YES. Determines whether ColdFusion throws an exception upon authentication failure.
USERNAME	Username	Required.

Example: The following example authenticates a user who supplied his name and password in a form against a predefined security context named `ReadOnly`:

```
<CFIF NOT IsAuthenticated()>
  <CFTRY>
  <CFAUTHENTICATE SECURITYCONTEXT="ReadOnly" USERNAME="#FORM.UName#"
   PASSWORD="#FORM.PW#">
  <CFCATCH TYPE="Security">
    <CFABORT SHOWERROR="You cannot be authenticated. Please use the Back button
     and try again.">
  </CFCATCH>
  </CFTRY>
</CFIF>
```

See also: `IsAuthenticated()`

<CFBREAK>

Description: The `<CFBREAK>` tag is used to break out of a looping process. Unlike `<CFABORT>`, it does not stop ColdFusion processing. `<CFBREAK>` has no attributes.

Syntax:

```
<CFBREAK>
```

Example: The following example queries all films from the `Films` table. It then loops through the list, comparing them to a `FORM` parameter, `FORM.LastFilmID`. If it finds a match, it saves the title of the selected film and then breaks out of the loop and continues processing below:

```
<CFQUERY NAME="GetMovies" DATASOURCE="#DSN#">
  SELECT FilmID, MovieTitle
  FROM Films
</CFQUERY>

<CFLOOP QUERY="GetMovies">
  <CFIF GetMovies.FilmID EQ FORM.LastFilmID>
    <CFSET SelectedFilm=GetMovies.MovieTitle>
    <CFBREAK>
  </CFIF>
</CFLOOP>
...
```

See also: `<CFABORT>`, `<CFEXIT>`

<CFCACHE>

Description: The <CFCACHE> tag is used to improve the performance on pages in which content doesn't need to be dynamically created each time the page is requested. ColdFusion instead returns static HTML output created during the last run of the page. The complete list of <CFCACHE> attributes is explained in Table A.6. Available ACTION attribute values are explained in Table A.7.

Note that <CFCACHE> works in conjunction with a mapping file, named Cfcache.map, that ColdFusion creates. The mapping file uses a format similar to a Windows .INI file. It is generated in the directory containing cached templates. The entries in the file identify cached templates, and multiple entries can exist for the same template if it takes URL parameters. The entries identify the static file that is cached in place of the template and the timestamp when a template was placed in the cache.

Syntax:

```
<CFCACHE ACTION="action" PROTOCOL="HTTP protocol" TIMEOUT="datetime"
 DIRECTORY="path" CACHEDIRECTORY="path" EXPIREURL="URL wildcard"
 PORT="port number">
```

TABLE A.6 <CFCACHE> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action	Optional. See Table A.7.
CACHEDIRECTORY	Full path to where pages are to be cached	Optional; defaults to current directory of the page.
DIRECTORY	Full path of directory of the cache.map file to be used	Optional; defaults to current directory. Used when ACTION=FLUSH.
EXPIREURL	URL wildcard reference that ColdFusion matches to all mappings in the cfcache.map file	Optional; used with ACTION=FLUSH. All matching files are flushed. Normally, all mappings are flushed.
PORT	Web server port	Optional; defaults to 80. Port from which page is being requested.
PROTOCOL	Identifies protocol used to create pages from cache.	Optional; defaults to HTTP.
TIMEOUT	DateTime value	Optional; specifies the oldest page that can be cached. ColdFusion uses all cached pages by default. When a cached page is older than the specified DateTime, ColdFusion refreshes the page.

TABLE A.7 <CFCACHE> ACTIONS

Action	Description
CACHE	Indicates that server-side caching is to be used. This is the default action.
FLUSH	Cached page is to be refreshed.
CLIENTCACHE	Indicates that client-side caching is to be used.
OPTIMAL	Indicates that an optimal combination of client- and server-side caching is to be used.

Example: This example caches a template as long as it hasn't changed in the last two hours:

```
<CFCACHE TIMEOUT="#DateAdd("h", "-2", Now())#" ACTION="Optimal">
<HTML>
<HEAD>
<TITLE>Dynamic Page to be Cached</TITLE>
</HEAD>
<BODY>
<H1>This page is cached</H1>

The last version of this page was on: <CFOUTPUT>#DateFormat(Now())#</CFOUTPUT>
</BODY>
</HTML>
```

The cache can be refreshed several ways. First, you can use the TIMEOUT attribute. ColdFusion refreshes the cache when the timestamp of the cached file is older than the TIMEOUT attribute value. It's preferable to use relative dates to fixed dates (as in the previous example).

Second, you can set the ACTION to FLUSH. This forces the cleanup of cached files. This ACTION enables you to use two other attributes, DIRECTORY and EXPIREURL.

<CFCOLLECTION>

Description: The <CFCOLLECTION> tag can be used to programmatically create and administer Verity collections. The complete list of <CFCOLLECTION> attributes is explained in Table A.8. Table A.9 lists the values for the ACTION attribute.

Syntax:

```
<CFCOLLECTION ACTION="action" COLLECTION="collection" LANGUAGE="language"
 PATH="path">
```

TABLE A.8 <CFCOLLECTION> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action	Required; see Table A.9.

Attribute	Description	Notes
COLLECTION	Collection name	Required; name of the collection to be indexed. If using external collections, this must be a fully qualified path to the collection.
LANGUAGE	Collection language	Optional language; defaults to English.
PATH	Collection path	Required if ACTION is CREATE.

TABLE A.9 <CFCOLLECTION> ACTIONS

Action	Description
CREATE	Creates a new collection
DELETE	Deletes a collection
MAP	Assigns an alias to a collection
OPTIMIZE	Purges and reorganizes a collection
REPAIR	Fixes a corrupt collection

The action CREATE creates a directory for the use of Verity, using the PATH attribute value. The COLLECTION attribute is used to create a subdirectory within the PATH directory. So, if PATH=MyDir and COLLECTION=MyCollection, the CREATE action would create a directory named c:\MyDir\MyCollection\.

The MAP action enables ColdFusion to reference a collection with an alias. This alias can be used in <CFINDEX> and to reuse a collection from an earlier installation of ColdFusion. The PATH attribute specifies the fully qualified path to the collection. Based on the example in the preceding paragraph, this would be c:\MyDir\MyCollection\.

Example: The example in Listing A.1 provides a form you can use to invoke any <CFCOLLECTION> action.

LISTING A.1 FORM TO INVOKE ANY <CFCOLLECTION> ACTION

```
<!--- This example contains two templates: a form and the
action template that processes it. --->
<HTML>
<HEAD>
  <TITLE>CFCOLLECTION Example</TITLE>
</HEAD>
<BODY>
<FORM ACTION="CollectionProcessor.cfm" METHOD="post">
<INPUT TYPE="Hidden" NAME="CollectionName_required">
<P>Collection name: <INPUT TYPE="Text" NAME="CollectionName">
<BR>(note: When Action is Map, enter the collection Alias name to use.)
<P>Action:
<SELECT NAME="Actions">
   <OPTION SELECTED>Create
```

LISTING A.1 CONTINUED

```
   <OPTION>Delete
   <OPTION>Map
   <OPTION>Optimize
   <OPTION>Repair
</SELECT>
<P><INPUT TYPE="Submit">
</FORM>
</BODY>
</HTML>

<!--- This is the form's action template, CollectionProcessor.cfm. --->
<CFCOLLECTION ACTION="#FORM.Actions#" COLLECTION="#FORM.CollectionName#">
```

Note

<CFCOLLECTION> works at the collection level only. To add content to a collection, use <CFINDEX>.

For more information about using <CFCOLLECTION> and Verity collections, see Chapter 36, "Full-Text Searching."

For the complete list of search expressions and instructions, see Appendix D, "Verity Search Language Reference."

See also: <CFINDEX>, <CFSEARCH>

<CFCONTENT>

Description: The <CFCONTENT> tags enables you to send non-HTML documents to a client's browser. <CFCONTENT> lets you specify the MIME type of the file and an optional filename to transmit. The complete list of supported attributes is in Table A.10.

Syntax:

```
<CFCONTENT TYPE="MIME Type" FILE="File Name" DELETEFILE="Yes/No" RESET="Yes/No">
```

TABLE A.10 <CFCONTENT> ATTRIBUTES

Attribute	Description	Notes
TYPE	Content MIME Type	Required
FILE	Filename	Optional attribute that specifies the fully qualified path of a file to be transmitted to the user's browser.
RESET	YES or NO	Optional. It discards output preceding call to <CFCONTENT>. Defaults to YES.
DELETEFILE	YES or NO	Optional; deletes file once sent; useful if serving dynamically created graphics.

Note

Because <CFCONTENT> needs to write to the HTTP header, you cannot use it if you've already flushed the header from the ColdFusion output buffer with <CFFLUSH>.

Example: The following example sends tab-delimited output (which easily can be read by a spreadsheet) to the browser. Note the use of tab-delimited field titles immediately following the <CFCONTENT> tag:

```
<CFQUERY NAME="GetFilmBudgets" DATASOURCE="OWS">
   SELECT DISTINCT FilmID, MovieTitle, AmountBudgeted
   FROM Films
   WHERE AmountBudgeted > 1000000.00
</CFQUERY>
<CFIF GetFilmBudgets.RecordCount EQ 0>
   <P>No films with budgets over one million dollars
   <CFABORT>
</CFIF>
<CFCONTENT TYPE="TEXT/TAB-DELIMITED" RESET>FilmID#chr(9)#Title#chr(9)#Budget
<CFOUTPUT QUERY="GetFilmBudgets">#FilmID##chr(9)##MovieTitle#
➥#chr(9)##AmountBudgeted#
</CFOUTPUT>
```

This next example sends a Microsoft Word document:

```
<CFCONTENT TYPE="application/msword" FILE="C:\MyDocs\Proposal.DOC">
```

This final example sends a dynamically created map to the user and then deletes it upon completion of the transmission:

```
<CFCONTENT TYPE="image/gif" FILE="C:\Images\Maps\Temp123.gif" DELETEFILE>
```

<CFCOOKIE>

Description: <CFCOOKIE> enables you to set *cookies*, persistent client-side variables, on the client browser. Cookies enable you to set variables on a client's browser, which are then returned every time a page is requested by a browser. Cookies can be sent securely if required. The tag attributes are described in Table A.11.

To access a returned cookie, specify its name and precede it with the COOKIE designator, as in #COOKIE.USER_ID#.

Users can configure their browsers to refuse cookies. You must never make assumptions about the existence of the cookie. Always use the IsDefined function to check for the existence of the cookie before referencing it.

Syntax:

```
<CFCOOKIE NAME="Cookie Name" VALUE="Value" EXPIRES="Expiration" SECURE="Yes/No"
 PATH="URL Path" DOMAIN=".domain">
```

TABLE A.11 <CFCOOKIE> ATTRIBUTES

Attribute	Description	Notes
DOMAIN	The domain for which the cookies are valid	Required only if PATH is used. Separate multiple domains with a ; character.
EXPIRES	Cookie expiration date	Optional; the cookie expiration date can be specified as a date (as in '10/1/97'), or as relative days (as in '100'), NOW, or NEVER.
NAME	Name of cookie	Required
PATH	Subset of the specified domain to which the cookie applies	Optional; separate multiple paths with a ; character.
SECURE	YES or NO	Optional; specifies that cookie must be sent securely. If it is specified and the browser does not support SSL, the cookie is not sent.
VALUE	Cookie value	Required.

Example: The following example assumes a login form (not presented here) has been used to capture a user's name and password. These are then compared to what is in the database. If there's a match, a secure UserID cookie is created on the user's browser; the cookie will expire in 60 days:

```
<CFQUERY NAME="CheckUser" DATASOURCE="OWS">
   SELECT UserID, UserName
   FROM Users
   WHERE UserName = '#FORM.UserName#'
   AND   Password = '#FORM.UserPassword#'
</CFQUERY>
<CFIF RecordCount EQ 1>
   <CFCOOKIE NAME="USER_ID" VALUE="CheckUser.UserID"
    EXPIRES="#DateFormat(DateAdd('d', 60, Now()), 'mm/dd/yy')#" SECURE="Yes">
<CFELSE>
   <CFABORT SHOWERROR="Invalid login. Go back and try again.">
</CFIF>
```

This next example deletes the UserID cookie:

```
<CFCOOKIE NAME="OrderID" EXPIRES="Now">
```

Tip

Do not use <CFCOOKIE> if you plan to redirect the user with <CFLOCATION> afterward. The cookie will not be created. Use a different technique to redirect the user, such as JavaScript (that is, location=...) or a <META HTTP-EQUIV="refresh" ...> tag.

Note

If you use the SECURE attribute to specify that the cookie must be sent securely, it is sent only if the browser supports SSL. If the cookie cannot be sent securely, it is not sent at all.

Note

Cookies are domain specific, meaning they can be set so just the server that set them can retrieve them.

Note

Because <CFCOOKIE> needs to write to the HTTP header, you cannot use it if you've already flushed the header from the ColdFusion output buffer with <CFFLUSH>.

For more information about using HTTP cookies, see Chapter 20, "Working with Sessions."

See also: <CFAPPLICATION>

<CFDIRECTORY>

Description: <CFDIRECTORY> is used for all directory manipulation, including obtaining directory lists and creating or deleting directories. The tag attributes are described in Table A.12.

<CFDIRECTORY> is a flexible and powerful tag and has many attributes, some of which are mutually exclusive. The values passed to the ACTION attribute dictate what other attributes can be used. The possible ACTION values for <CFDIRECTORY> are listed in Table A.13. LIST is assumed if no ACTION is specified. Table A.14 contains the list of columns returned if ACTION="LIST".

When ACTION="LIST" is specified, the tag creates a query containing the file list from the specified directory.

Syntax:

```
<CFDIRECTORY ACTION="Action Type" DIRECTORY="Directory Name"
 FILTER="Search Filter" MODE="Unix Permissions Mode" NAME="Query Name"
 NEWDIRECTORY="New Directory Name" SORT="Sort Order">
```

TABLE A.12 <CFDIRECTORY> ATTRIBUTES

Attribute	Description	Notes
ACTION	Tag action	Optional; defaults to LIST if omitted.
DIRECTORY	Directory name	Required.
FILTER	Filter spec	Optional; only valid if ACTION="LIST". It can contain wildcard characters.

TABLE A.12 CONTINUED

Attribute	Description	Notes
MODE	Permissions mode	Optional; only valid if ACTION="CREATE". It is used only by the Solaris version of ColdFusion and is ignored by the Windows versions.
NAME	Query name	Required if ACTION="LIST". Query to hold retrieved directory listing.
NEWDIRECTORY	New directory name	Required if ACTION="RENAME"; ignored by all other actions.
SORT	Sort order for use when ACTION="LIST"	Optional comma-delimited list of columns to sort by; each can use ASC for ascending or DESC for descending. Default is ascending.

TABLE A.13 <CFDIRECTORY> ACTIONS

Action	Description
CREATE	Creates a new directory
DELETE	Deletes a directory
LIST	Obtains a list of directory contents
RENAME	Renames a directory

TABLE A.14 <CFDIRECTORY> LIST COLUMNS

Action	Description
ATTRIBUTES	File attributes
DATELASTMODIFIED	Last modified date
MODE	Permissions mode (Solaris, HP-UX, and Linux only)
NAME	File or directory name
SIZE	Size in bytes
TYPE	Type F for file or D for directory

Example: This first example is a template that processes a form in which the user has specified a directory name. The example traps for errors:

```
<CFTRY>
<CFDIRECTORY ACTION="CREATE" DIRECTORY="#FORM.UserDir#">
<CFCATCH TYPE="ANY">
   <CFIF CFCATCH.Detail CONTAINS "when that file already exists">
      <P>Cannot create directory: this directory already exists.
   <CFELSE>
```

```
      <CFOUTPUT>#CFCATCH.Detail#</CFOUTPUT>
   </CFIF>
   <CFABORT>
</CFCATCH>
</CFTRY>
<P>Directory created.
```

This next example retrieves a directory list, sorted by filename. The resulting query is displayed in a table:

```
<CFDIRECTORY ACTION="LIST" DIRECTORY="C:\INETPUB\WWWROOT\TEST\"
 NAME="Stuff" SORT="Name">
<CFTABLE QUERY="Stuff" COLHEADERS="YES" HTMLTABLE="YES" BORDER="YES">
   <CFCOL HEADER="<B>Name</B>" ALIGN="LEFT" TEXT="Name">
   <CFCOL HEADER="<B>Size</B>" ALIGN="LEFT" TEXT="Size">
</CFTABLE>
```

For more information about ColdFusion's file and directory manipulation capabilities, see Chapter 35, "Interacting with the Operating System."

Note

Note that this tag, due to the inherent danger in using it, can be disabled in the ColdFusion Administrator under Basic Security.

See also: <CFFILE>

<CFDUMP>

Description: <CFDUMP> enables you to debug variable values. It can output the contents of simple variables, queries, structures, arrays, and serialized WDDX packets. The attribute for this tag is presented in Table A.15.

Syntax:

```
<CFERROR Var="Variable Name">
```

TABLE A.15 <CFDUMP> ATTRIBUTES

Attribute	Description	Notes
VAR	Variable name	Required; name of variable to dump

Example: Suppose you wanted to view the contents of a query right after it executes. You could use code such as that in Listing A.2. The values being dumped get progressively more complex. Note how you must enclose variable names in the pound sign. Note also that <CFDUMP> will work on nested types of variables.

LISTING A.2 USE OF <CFDUMP> TO VIEW VARIABLE CONTENTS

```
<CFSET TheTime=Now()>
<CFDUMP VAR="#TheTime#">

<CFQUERY NAME="GetContacts" DATASOURCE="OWS">
  SELECT LastName, FirstName
  FROM Contacts
</CFQUERY>
<CFDUMP VAR="#GetContacts#">

<CFSET Meals=StructNew()>
<CFSET Meals["Breakfast"]="Cereal">
<CFSET Meals["Lunch"]=StructNew()>
<CFSET Meals["Lunch"]["MainCourse"]="Sandwich">
<CFSET Meals["Lunch"]["Beverage"]="Bloody Mary">

<CFDUMP VAR="#Meals#">
```

<CFERROR>

Description: <CFERROR> enables you to override the standard ColdFusion error messages and replace them with special error-handling templates that you specify. <CFERROR> requires that you specify the type of error message to be overridden and the template containing the error message to be displayed. Table A.16 provides attribute values.

Four types of error messages are available in ColdFusion. REQUEST errors occur while processing a template, and VALIDATION errors occur when FORM field validation errors occur. MONITOR errors are invoked before <CFTRY>/<CFCATCH> blocks or other <CFERROR> types, so they are useful when debugging. Trapping for EXCEPTION errors enables you to trap any unhandled errors. You also can specify an EXCEPTION-handling template in the ColdFusion Administrator.

You cannot use CFML in error-handling templates for REQUEST errors; only HTML/JavaScript is allowed. Templates that handle EXCEPTION errors, however, are capable of using CFML. MONITOR, REQUEST, and EXCEPTION error-handling templates also have access to a special error structure named ERROR.

Syntax:

```
<CFERROR Type="Error Type" TEMPLATE="Error Message Template
MAILTO="Administrator's e-mail address" EXCEPTION="Exception type">
```

TABLE A.16 <CFERROR> ATTRIBUTES

Attribute	Description	Notes
EXCEPTION	Identifies the exception type	Required if the TYPE is EXCEPTION or MONITOR.

Attribute	Description	Notes
MAILTO	The administrator's e-mail address	The e-mail address of the administrator to be notified of any error messages. This value is available with the error message template as #ERROR.MailTo#.
TEMPLATE	Error message template	Required; name of the template containing the error-handling code.
TYPE	Type of error message	Optional; values are REQUEST, VALIDATION, MONITOR, and EXCEPTION. If this attribute is omitted, the default value of REQUEST is used. An ERROR variable is created for REQUEST, MONITOR, and EXCEPTION types. This is described in Table A.17. The values for VALIDATION errors are presented in Table A.18.

TABLE A.17 COLDFUSION ERROR MESSAGE VARIABLES FOR REQUEST, MONITOR, AND EXCEPTION ERROR TYPES

Variable	Description
#ERROR.BROWSER#	The browser the client was running, with version and platform information if provided by the browser.
#ERROR.DATETIME#	The date and time that the error occurred; can be passed to any of the date/time manipulation functions as necessary.
#ERROR.DIAGNOSTICS#	Detailed diagnostic error message returned by ColdFusion.
#ERROR.GENERATEDCONTENT#	Content generated by the failed template.
#ERROR.HTTPREFERER#	URL of the page from which the template was accessed.
#ERROR.MAILTO#	Administrator's e-mail address; can be used to send notification of the error.
#ERROR.QUERYSTRING#	The URL query string of the request.
#ERROR.REMOTEADDRESS#	Client's IP address.
#ERROR.TEMPLATE#	Template being processed when the error occurred.

TABLE A.18 COLDFUSION ERROR MESSAGE VARIABLES FOR THE VALIDATION ERROR TYPE

Variable	Description
#ERROR.InvalidFields#	List of the invalid form fields
#ERROR.ValidationFooter#	Text for footer of error message
#ERROR.ValidationHeader#	Text for header of error message

Example: The following example, `Application.cfm` (shown in Listing A.3), establishes an error-message template for `REQUEST` errors:

LISTING A.3 USING <CFERROR> TO TRAP EXCEPTIONS

```
<!--- Here's what the Application.cfm might look like --->
<CFAPPLICATION NAME="MyApp">
<CFERROR TYPE="REQUEST" NAME="ERROR_REQUEST.CFM"
 MAILTO="admin@orangeWhipStudios.com">
<CFERROR TYPE="EXCEPTION" NAME="ERROR_EXCEPTION.CFM">
<!--- And here's an example of what the template that handles
  EXCEPTION
 type errors might look like --->
<CFMAIL
  FROM="system@orangeWhipStudios.com"
  TO="admin@orangeWhipStudios.com"
  SUBJECT="Error on OWS application">
An error has occurred:

#ERROR.Detail#
</CFMAIL>

<HTML>
<HEAD>
    <TITLE>Application Error</TITLE>
</HEAD>
<BODY>
<CFIF Error.Template EQ "GetCustomers.cfm">
   <H2>An error occurred getting customer records.</H2>
<CFELSE>
   <H2>An Error Has Occurred</H2>
</CFIF>
<P>The administrator has been notified.
</BODY>
</HTML>
</HTML>
```

Tip

The `<CFERROR>` tag is best used in the `Application.cfm` template, as explained in Chapter 19.

<CFEXECUTE>

Description: Enables you to execute processes on the ColdFusion server machine. `<CFEXECUTE>` is frequently used to execute a server program at a regular interval using `<CFSCHEDULE>`. Table A.19 describes the tag attributes.

Note

Note that you should not put any CFML tags or functions between the start and end tags.

Syntax:

```
<CFEXECUTE NAME="ExecutableName" ARGUMENTS="CommandLine"
 OUTPUTFILE="Pathname For Output" TIMEOUT="Time interval in seconds"></CFEXECUTE>
```

TABLE A.19 <CFEXECUTE> ATTRIBUTES

Attribute	Description	Notes
NAME	Name of executable	Required.
ARGUMENTS	Any command-line parameters that must be passed	Optional; only used if the program needs to be passed command-line arguments.
OUTPUTFILE	Pathname to file into which output from program should be written	Optional; if not specified, any output appears on the page in which this tag is used.
TIMEOUT	Number of seconds that program should be given before it's timed out	Optional; ColdFusion spawns another thread to execute the program. This enables you to ensure that the program doesn't run forever.

Example: This example invokes PKZip to produce a user-specified Zip file by passing it an array of arguments:

```
<CFSCRIPT>
args = ArrayNew(1);
args[1] = "-a";
args[2] = "c:\temp\test.zip";
args[3] = "c:\inetput\wwwwroot\MyApp\UTIL.*";
</CFSCRIPT>
<CFTRY>
<CFEXECUTE NAME="c:\utils\pkzip.exe " ARGUMENTS="#args#"
 OUTPUTFILE="C:\TEMP\TESTOUT.TXT"></CFEXECUTE>
<CFCATCH TYPE="Any">
   <CFOUTPUT>#CFCATCH.Message#</CFOUTPUT>
</CFCATCH>
</CFTRY>
```

Note

Note the arguments can be passed as an array or as a string.

See also: <CFSCHEDULE>

<CFEXIT>

Description: <CFEXIT> aborts the processing of a custom tag without aborting processing of the calling template. When used in a regular template, <CFEXIT> acts just like <CFABORT>. When called within a custom tag, however, <CFEXIT> does not terminate processing of the calling template (like <CFABORT>), but instead returns processing to it. The values for the METHOD attribute are presented in Table A.20.

Syntax:

```
<CFEXIT METHOD="Method">
```

TABLE A.20 METHODS USED IN <CFEXIT>

`ExitTag`	Default; aborts current tag processing, similar to `<CFABORT>`. If used in a custom tag, this causes processing to continue after end tag.
`ExitTemplate`	Exits the currently processing tag. If used where a custom tag's execution mode is `Start`, this will continue processing from the first child tag in the current custom tag's body. If used where a custom tag's execution mode is `End`, it will continue processing after the end tag.
`Loop`	Reexecutes body in current custom tag. This method can be used only when a custom tag's execution mode is `End`. In this case, it will continue processing from the first child tag in the current custom tag's body. In all other contexts, it will return an error.

Example: This example checks to see whether a particular attribute was passed. It stops the custom tag's processing if the attribute is missing but enables processing to continue in the calling template after the custom tag:

```
<CFIF NOT IsDefined("Attributes.MyAttrib")>
   <CFEXIT>
</CFIF>
...
```

See also: `<CFABORT>`

<CFFILE>

Description: `<CFFILE>` is used to perform file-management operations, including uploading files from a browser; moving, renaming, copying, and deleting files; and reading and writing files. The tag attributes are described in Table A.21.

`<CFFILE>` is a flexible and powerful tag; it has several attributes, many of which are mutually exclusive. The values passed to the `ACTION` attribute dictate which other attributes can be used. These values are described in Table A.22. Table A.23 describes the options you have when dealing with filename conflicts arising from file upload operations.

`<CFFILE>` creates a `FILE` object after every `<CFFILE>` operation. You can use the variables in this object as you would any other ColdFusion variables, enabling you to check the results of an operation. However, only one `FILE` object exists, and as soon as you execute a new `<CFFILE>` tag, the prior `FILE` object is overwritten with the new one. `FILE` object variables are described in Table A.24.

Syntax:

```
<CFFILE ACCEPT="Filter" ACTION="Action Type"
 DESTINATION="Destination Directory or File Name" FILE="File Name"
 FILEFIELD="Field Containing File Name" NAMECONFLICT="Conflict Option"
 OUTPUT="Text To Output" SOURCE="Source File Name" VARIABLE="Variable Name">
```

TABLE A.21 <CFFILE> ATTRIBUTES

Attribute	Description	Notes
ACCEPT	File type filter	This optional attribute restricts the types of files that can be uploaded, and can be used only if ACTION is UPLOAD. The filter is specified as a MIME type ("image/*", which allows all image types, but nothing else); multiple MIME types can be specified separated by commas.
ACTION	Desired action	This attribute is required.
DESTINATION	Destination file location	This attribute can be used only if ACTION is one of the following: COPY, MOVE, RENAME, or UPLOAD. Destination can be a filename or a fully qualified file path.
FILE	Name of local file to access	This attribute can be used only if ACTION is APPEND, DELETE, READ, READBINARY, or WRITE, in which case it is required.
FILEFIELD	Name of the FILE type <INPUT> containing the file	This attribute can be used only if ACTION is UPLOAD, in which case it is required.
MODE	Identifies file access permissions for uploaded file	This is used only in Unix-type installations. See Table A.25.
NAMECONFLICT	What to do in case of name conflicts	This optional attribute can be used if ACTION is UPLOAD. It specifies the course to take if a name conflict arises from the upload. If this attribute is omitted, the default value of "ERROR" is used.
OUTPUT	Text to output to file	This attribute can be used only if ACTION is WRITE or APPEND.
SOURCE	Source filename	Name of the source file to be written to, copied, or moved. Can be used only if ACTION is COPY, MOVE, or RENAME.
VARIABLE	Variable to store contents of read file	This attribute can be used only if ACTION is READ or READBINARY, in which case it is required.

TABLE A.22 <CFFILE> ACTIONS

Action	Description
APPEND	Appends one text file to the end of another
COPY	Copies a file

TABLE A.22 CONTINUED

Action	Description
DELETE	Deletes a specified file
MOVE	Moves a specified file from one directory to another, or from one filename to another
READ	Reads the contents of a text file
READBINARY	Reads the contents of a binary file
RENAME	Does the same thing as MOVE; see MOVE
UPLOAD	Receives an uploaded file
WRITE	Writes specified text to the end of a text file

TABLE A.23 FILE UPLOAD NAME CONFLICT OPTIONS

Option	Description
ERROR	The file will not be saved, and ColdFusion will immediately terminate template processing.
SKIP	Neither saves the file nor generates an error message.
OVERWRITE	Overwrites the existing file.
MAKEUNIQUE	Generates a unique filename and saves the file with that new name. To find out what the new name is, inspect the #FILE.ServerFile# field.

TABLE A.24 <CFFILE> FILE OBJECT VARIABLES

Field	Description
#FILE.ATTEMPTEDSERVERFILE#	The original attempted filename; will be the same as #FILE.ServerFile# unless the name had to be changed to make it unique
#FILE.CLIENTDIRECTORY#	The client directory from where the file was uploaded, as reported by the client browser
#FILE.CLIENTFILE#	The original filename as reported by the client browser
#FILE.CLIENTFILEEXT#	The original file extension, as reported by the client browser
#FILE.CLIENTFILENAME#	The original filename as reported by the client browser, but without the file extension
#FILE.CONTENTSUBTYPE#	The MIME subtype of an uploaded file
#FILE.CONTENTTYPE#	The primary MIME type of an uploaded file
#FILE.DATELASTACCESSED#	The last date and time the file was accessed
#FILE.FILEEXISTED#	Yes if file already existed; No if not
#FILE.FILESIZE#	Size of file that was uploaded

Field	Description
#FILE.FILEWASAPPENDED#	Yes if file were overwritten; No if not
#FILE.FILEWASOVERWRITTEN#	Yes if file were overwritten; No if not
#FILE.FILEWASRENAMED#	Yes if file were renamed; No if not
#FILE.FILEWASSAVED#	Yes is file were saved; No if not
#FILE.OLDFILESIZE#	Size of file that was overwritten
#FILE.SERVERDIRECTORY#	The server directory in which the uploaded file was saved
#FILE.SERVERFILEEXT#	The file extension of the uploaded file on the server (does not include period)
#FILE.SERVERFILE#	The name of the file as saved on the server (takes into account updated filename if it were modified to make it unique)
#FILE.SERVERFILENAME#	The name of the uploaded file on the server, without an extension
#FILE.TIMECREATED#	Time when the file was uploaded
#FILE.TIMELASTMODIFIED#	Date and time uploaded file was last modified

Note that when you're uploading files under a Unix-type operating system, you also can use the optional MODE attribute. The possible values for MODE are described in Table A.25.

TABLE A.25 MODE VALUES USED WHEN UPLOADING TO UNIX SYSTEMS

Mode	Description
644	Assigns owner read/write and assigns other/group read permissions
666	Assigns read/write permissions to all
777	Assigns read, write, and execute permissions to all

Example: The two basic categories of use for <CFFILE> are uploading files and file management (copying, moving, renaming, and so on).

The example in Listing A.4 includes both types of use. It includes a form that is used to specify a file to be uploaded and the <CFFILE> code from the form's action template that does the uploading.

It checks to ensure the file is less than 50K (this arbitrary file size was chosen just to illustrate the use of the CFFILE variable). The example uses <CFTRY><CFCATCH> code to trap this user-defined error. If the file is less than 50K, the progress is reported to the user.

It then checks the file extension to ensure that it is a GIF. This again is arbitrary and is included just to demonstrate this technique for limiting file types that can be uploaded.

The last part of the example moves the file to a different directory and reports to the user.

LISTING A.4 EXAMPLES OF FILE UPLOAD AND MANAGEMENT

```
<!--- Code in the upload form. Must use ENCTYPE=MULTIPART/FORM-DATA --->
<FORM ENCTYPE="MULTIPART/FORM-DATA" ACTION="Upload_Action.cfm" METHOD="POST">
<!--- An INPUT of type FILE is required --->
<P>File to upload: <INPUT TYPE="FILE" NAME="UploadFile" SIZE="40">
<P><INPUT TYPE="SUBMIT" NAME="Upload" VALUE="Upload">
</FORM>
...

<!--- Code in the upload form's action template --->
<CFTRY>
<CFFILE
  ACTION="Upload"
  FILEFIELD="UploadFile"
  DESTINATION="D:\TEMP\"
  NAMECONFLICT="MakeUnique"
>

<!--- Make sure file isn't more than 50k --->
<CFIF CFFILE.FileSize GT 51250>
  <CFTHROW TYPE="SizeError" MESSAGE="File too large. Cannot be more than 5k.">
</CFIF>
<CFIF CFFILE.ClientFileExt NEQ "gif">
  <CFTHROW TYPE="ExtError"
   MESSAGE="File is wrong type. You can only upload GIF files.">
</CFIF>

<CFCATCH TYPE="SizeError">
  <CFABORT SHOWERROR="#CFCATCH.Message#">
</CFCATCH>
<CFCATCH TYPE="ExtError">
  <CFABORT SHOWERROR="#CFCATCH.Message#">
</CFCATCH>
</CFTRY>

<!--- Report on status to user --->
<P>File uploaded successfully:
<CFOUTPUT>
<P>Filename on client: #CFFILE.ClientFile#
<P>Filename on server: #CFFILE.ServerFile#
<P>File size: #CFFILE.FileSize#
</CFOUTPUT>

<!--- Now move file to a different folder --->
<CFFILE
  ACTION="Move"
  SOURCE="D:\TEMP\#CFFILE.ServerFile#"
  DESTINATION="D:\JUNK\#CFFILE.ServerFile#"
>
<P>File successfully moved to D:\JUNK\<CFOUTPUT>#CFFILE.ServerFile#</CFOUTPUT>
```

For more information about ColdFusion's file manipulation capabilities, see Chapter 35.

Caution

Be careful to use <CFLOCK> when using <CFFILE> because it uses a shared resource (the server's filesystem) that multiple ColdFusion threads can try to access simultaneously.

See also: <CFDIRECTORY>, <CFFTP>, <CFLOCK>

<CFFLUSH>

Description: <CFFLUSH> flushes ColdFusion's output buffer, sending the contents back to the Web browser. You can control the point at which the flush takes place with the INTERVAL attribute by entering the number of bytes. The attributes for this tag are presented in Table A.26.

Syntax:

```
<CFFLUSH INTERVAL="number of bytes">
```

TABLE A.26 <CFFLUSH> ATTRIBUTES

Attribute	Description	Notes
INTERVAL	Number of bytes	Optional; specifies number of bytes that must be accumulated before buffer is flushed

Note

Several ColdFusion tags are used to write data into the ColdFusion output buffer, including <CFHEADER>, <CFHTMLHEAD>, <CFCOOKIE>, <CFFORM>, <CFCONTENT>, and <CFLOCATION>. A run error will be generated if you try to use these tags after having flushed part of the HTML document (such as the HTTP header or HTML <HEAD>) to which these tags need to write.

Example: The example in Listing A.5 employs <CFFLUSH> to get some output out of the buffer as a page with many queries is built. This gives the user some intermediate feedback as to what is taking place.

LISTING A.5 FLUSHING OUT PARTIAL PAGE OUTPUT WITH <CFFLUSH>

```
<CFQUERY NAME="GetAllFilms" DATASOURCE="OWS">
   SELECT *
   FROM Films
</CFQUERY>

<H1>Retrieved all Films</H1>
<CFFLUSH>

<CFQUERY NAME="GetAllActors" DATASOURCE="OWS">
   SELECT *
   FROM Actors
</CFQUERY>
```

LISTING A.5 CONTINUED

```
<H1>Retrieved all Actors</H1>
<CFFLUSH>

<CFQUERY NAME="GetAllFilmsActors" DATASOURCE="OWS">
   SELECT *
   FROM FilmsActors
</CFQUERY>

<H1>Retrieved Everything</H1>
```

See Chapter 23, "Improving the User Experience," for additional examples.

<CFFORM>, </CFFORM>

Description: <CFFORM> and </CFFORM> are alternatives for the standard HTML <FORM> and </FORM> tags. The <CFFORM> tag is not useful by itself. However, it enables you to use other tags (<CFGRID>, <CFINPUT>, <CFSELECT>, <CFTEXTINPUT>, <CFSLIDER>, <CFTREE>, or any Java applets of your own using <CFAPPLET>), which do add a great deal of functionality to HTML forms. The code generated by <CFFORM> is standard FORM HTML and JavaScript code. The attributes for this tag are presented in Table A.27.

Note that ColdFusion can create JavaScript functions and event handlers in the code that is returned to the browser. These functions are necessary to provide the functionality you specify in your input objects, such as required fields and other validations.

Syntax:

```
<CFFORM ACTION="Action Page" ACTION="action" ENABLECAB="header"
ENCTYPE="mimetype" ONSUBMIT="javascript function"
PASSTHROUGH="HTML attribs" PRESERVEDATA="YES or NO"
TARGET="window">...</CFFORM>
```

TABLE A.27 <CFFORM> ATTRIBUTES

Attribute	Description	Notes
ACTION	Form action page	Required.
ENABLECAB	Header value	Optional; allows the downloading Java classes in Microsoft cabinet files. If Yes, users are asked upon opening the page whether they want to download the CAB file.
ENCTYPE	MIME type used to encode data sent via an HTTP POST method	Optional; default value is application/x-www-form-urlencoded.
NAME	Form name	Optional; if used, you must ensure that the form name is unique.

Attribute	Description	Notes
ONSUBMIT	JavaScript OnSubmit function	Optional name of JavaScript function to be executed prior to form submission.
PASSTHROUGH	HTML <FORM> attributes and values not supported directly by <CFFORM>	Optional; you can pass *attribute=value* pairs that aren't explicitly supported by <CFFORM> and they're passed on to the browser.
PRESERVEDATA	YES or NO	Optional; enables Java controls in form (for example, <CFSLIDER>) to maintain their most recent values when the form acts as its own action template.
TARGET	Target window	Optional target window.

Example: The following is a simple <CFFORM>:

```
<CFFORM ACTION="Test_Action.cfm" PASSTHROUGH="CLASS=""DataEntry""">
<P>First Name: <CFINPUT NAME="FirstName" REQUIRED="Yes"
   MESSAGE="You must enter a First Name.">
<P>Last Name: <CFINPUT NAME="LastName" REQUIRED="Yes"
   MESSAGE="You must enter a Last Name.">
<INPUT TYPE="Submit" VALUE="Save">
</CFFORM>
```

Note that the CLASS attribute and value will be passed through ColdFusion to the browser. Here is some of the code that is returned to the browser by ColdFusion:

```
<FORM NAME="DataEntryTest" ACTION="Test_Action.cfm" METHOD=POST
 onSubmit="return _CF_checkCFForm_1(this)" CLASS="This">
<P>First Name: <INPUT TYPE="Text" NAME="FirstName">
<P>Last Name: <INPUT TYPE="Text" NAME="LastName">
<INPUT TYPE="Submit" VALUE="Save">
</FORM>
```

You can see that ColdFusion passed the CLASS attribute/value pair through to the browser. Also note how it added the onSubmit() JavaScript event to the <FORM> tag. This is due to the use of <CFINPUT> to produce required fields. ColdFusion also creates the JavaScript function CF_CheckCFForm_1() to process the required field validation.

Note

<CFFORM> automatically embeds METHOD="POST" in your form.

Tip

If you specify a value in quotation marks, you must escape the quotation marks by doubling them, for example

```
PASSTHROUGH="STYLE= ""DataEntry"" "
```

Note

Because <CFFORM> must write to the HTML <HEAD>, you cannot use it if you've already flushed the <HEAD> from the ColdFusion output buffer with <CFFLUSH>.

See also: <CFAPPLET>, <CFGRID>, <CFINPUT>, <CFSELECT>, <CFSLIDER>, <CFTEXTINPUT>, <CFTREE>

<CFFTP>

Description: <CFFTP> is the ColdFusion interface to the Internet standard file transfer protocol. It enables your ColdFusion program to function as an FTP client, interacting with remote filesystems. It is a very powerful and complex tag; Table A.28 lists its attributes. Values for the ACTION attribute are presented in Table A.29.

When calls to <CFFTP> are completed—assuming STOPONERROR is set to No (the default)—a series of variables is set so you can determine the success or failure of the operation. These variables are listed in Table A.30. An error code is set if an error occurs. Table A.32 lists the complete set of error codes and their meanings.

<CFFTP> can be used to retrieve remote directory lists. Lists are returned in ColdFusion query format, and Table A.31 lists the query columns.

<CFFTP> is designed to be used two ways: either for single operations or to batch operations together. To use the batch mode (called *cached mode*), you must specify a unique name in the CONNECTION attribute that you can use in future <CFFTP> calls.

Syntax:

```
<CFFTP ACTION="Action" ASCIIEXTENSIONLIST="List"
 ATTRIBUTES="Attributes" CONNECTION="Connection Name" DIRECTORY="Directory"
 EXISTING="Name" FAILIFEXISTS="Yes or No" ITEM="Name" LOCALFILE="Name"
 NAME="Query Name" NEW="Name" PASSWORD="Password" PORT="Port" REMOTEFILE="Name"
 RETRYCOUNT="Count" SERVER="Server Address" STOPONERROR="Yes or No"
 TIMEOUT="Seconds" TRANSFERMODE="Mode" USERNAME="User Name">
```

TABLE A.28 <CFFTP> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action	Required.
ASCIIEXTENSIONLIST	ASCII extensions	Optional; semicolon-delimited list of extensions to be treated as ASCII extensions if using TRANSFERMODE of "AutoDetect"; default is "txt;htm;html;cfm;cfml;shtm;shtml;css;asp;asa".
ATTRIBUTES	Attributes list	Comma-delimited list of attributes; specifies the file attributes for the local file. Possible values are READONLY, HIDDEN, SYSTEM, ARCHIVE, DIRECTORY, COMPRESSED, TEMPORARY, and NORMAL.

Attribute	Description	Notes
CONNECTION	Connection name	Optional; used to cache connections to perform operations without logging in again.
DIRECTORY	Directory on which operation is to be performed	Required if ACTION is CHANGEDIR, CREATEDIR, LISTDIR, or EXISTSDIR.
EXISTING	Current name of file or directory on remote system	Required if ACTION is RENAME.
FAILIFEXISTS	YES or NO	Optional; indicates whether a GETFILE action will fail if a local file with the same name already exists; defaults to YES.
ITEM	Item (file) name	Required if ACTION is EXISTS or REMOVE.
LOCALFILE	Local filename	Required if ACTION is GETFILE or PUTFILE.
NAME	Query name	Required if ACTION is LISTDIR; see Table A.31 for column list.
NEW	New item (file) name when file is being renamed	Required if ACTION is RENAME.
PASSIVE	Allows you to enable or disable passive mode	Optional; defaults to NO.
PASSWORD	Login password	Required when ACTION is OPEN.
PROXYSERVER	Name of proxy server	Used if you must go through a proxy server.
PORT	Server port	Optional attribute; defaults to 21.
REMOTEFILE	Remote filename	Required if ACTION is EXISTSFILE, GETFILE, or PUTFILE.
RETRYCOUNT	Number of retries	Optional retry count; defaults to 1.
SERVER	Server name; DNS or IP address of FTP server	Required when not using a cached connection.
STOPONERROR	YES or NO	Optional; defaults to YES. When it's NO, three status variables are produced, as found in Table A.30.
TIMEOUT	Timeout seconds	Optional; timeout value in seconds.
TRANSFERMODE	Transfer mode	Optional; values can be ASCII, BINARY, or AUTODETECT (default).
USERNAME	Login username	Required when ACTION is OPEN.

TABLE A.29 <CFFTP> ACTIONS

Action	Description
CHANGEDIR	Changes directory
CLOSE	Closes a cached connection
CREATEDIR	Creates a directory
EXISTS	Checks to see whether an object exists
EXISTSDIR	Checks for a directory's existence
EXISTSFILE	Checks for a file's existence
GETCURRENTDIR	Gets current directory
GETCURRENTURL	Gets current URL
GETFILE	Retrieves a file
LISTDIR	Retrieves directory list
OPEN	Opens a cached connection
PUTFILE	Sends a file
REMOVE	Deletes a file
REMOVEDIR	Removes a directory
RENAME	Renames a file

Note

Note that the status variables in Table A.30 are produced only when the STOPONERROR attribute is set to NO. It's set to YES by default.

TABLE A.30 <CFFTP> STATUS VARIABLES

Variable	Description
#CFFTP.ErrorCode#	Error codes (see Table A.31)
#CFFTP.ErrorText#	Error text
#CFFTP.Succeeded#	Success; YES or NO

TABLE A.31 <CFFTP> QUERY COLUMNS

Column	Description
ATTRIBUTES	Comma-delimited list of attributes
ISDIRECTORY	YES if directory; NO if file
LASTMODIFIED	Date and time last modified
LENGTH	File length
MODE	An octal format string listing Unix permissions

Column	Description
NAME	Object name
PATH	Full path to object
URL	Full URL to object

TABLE A.32 <CFFTP> ERROR CODES

Code	Description
0	Operation succeeded.
1	System error (OS or FTP protocol error).
2	An Internet session could not be established.
3	FTP session could not be opened.
4	File transfer mode not recognized.
5	Search connection could not be established.
6	Invoked operation valid only during a search.
7	Invalid timeout value.
8	Invalid port number.
9	Not enough memory to allocate system resources.
10	Cannot read contents of local file.
11	Cannot write to local file.
12	Cannot open remote file for reading.
13	Cannot read remote file.
14	Cannot open local file for writing.
15	Cannot write to remote file.
16	Unknown error.
18	File already exists.
21	Invalid retry count specified.

Note

Because it can be rather dangerous, <CFFTP> can be disabled using the Basic Security settings in the ColdFusion Administrator.

Example: The example in Listing A.6 opens a connection and and caches it using the CONNECTION attribute. It reads the directory and outputs the resulting query (identified by the NAME attribute). This query is then displayed with a simple <CFTABLE>.

The next call to <CFFTP> uses the cached connection named in the first call. It is used to upload a file from the local ColdFusion server to the remote FTP server by using the PUTFILE ACTION. It then displays the value of the CFFILE.Succeeded to verify the success.

If the file was uploaded successfully, the file is deleted and the operation's success is displayed. A new directory listing is displayed.

LISTING A.6 USING <CFFTP>

```
<CFFTP
   SERVER="ftp.someserver.com"
   USERNAME="joeuser"
   PASSWORD="myPassword"
   ACTION="LISTDIR"
   DIRECTORY="c:\temp\"
   NAME="MyDocs"
   CONNECTION="MyConn"
>

<HTML>
<HEAD>
   <TITLE>CFFTP Example</TITLE>
</HEAD>

<BODY>

<P>Directory of files
<CFTABLE BORDER="Yes" HTMLTABLE="yes" QUERY="MyDocs">
<CFCOL HEADER="Name" TEXT="#Name#" ALIGN="LEFT">
<CFCOL HEADER="Length" TEXT="#Length#" ALIGN="RIGHT">
</CFTABLE>

<CFFTP
   ACTION="PUTFILE"
   LOCALFILE="c:\temp\tcmnote.txt"
   REMOTEFILE="c:\temp\tcmnote.txt"
>
<p>Uploaded? <CFOUTPUT> #CFFTP.Succeeded#</CFOUTPUT>

<CFIF CFFTP.Succeeded>
   <CFFTP
      ACTION="REMOVE"
      ITEM="c:\temp\tcmnote.txt"
   >

   <p>Deleted? <CFOUTPUT> #CFFTP.Succeeded#</CFOUTPUT>

   <CFFTP
      ACTION="LISTDIR"
      DIRECTORY="c:\temp\"
      NAME="MyDocs"
   >

   <P>Directory of files
   <CFTABLE BORDER="Yes" HTMLTABLE="yes" QUERY="MyDocs">
```

```
   <CFCOL HEADER="Name" TEXT="#Name#" ALIGN="LEFT">
   <CFCOL HEADER="Length" TEXT="#Length#" ALIGN="RIGHT">
   </CFTABLE>
</CFIF>
</BODY>
</HTML>
```

See also: <CFHTTP>, <CFFILE>

<CFGRAPH>,<CFGRAPHDATA>, </CFGRAPH>

This extremely powerful set of tags enables you to dynamically create Web-based graphs. You can produce dynamic bar (vertical or horizontal), line, and pie charts built from query results. Three sets of similar attributes are available to support the three types of graphs. You also can employ the child tag, <CFGRAPHDATA>, to add individual data points to the graph or to produce graphs entirely from hard-coded data points.

The attributes for <CFGRAPH> are presented in Table A.33; the attributes for <CFGRAPHDATA> are presented in Table A.34.

Syntax for dynamic bar and horizontal bar graphs:

```
<CFGRAPH BACKGROUNDCOLOR="Web color" BARSPACING="Integer number of pixels"
 BORDERCOLOR="Web color" BORDERWIDTH="Integer number of pixels"
 COLORLIST="Web color list" DEPTH="Integer number of pixels"
 FILEFORMAT="Flash or Jpg" GRAPHHEIGHT="Integer number of pixels"
 GRAPHWIDTH="Integer number of pixels" GRIDLINES="Integer number of lines"
 ITEMCOLUMN="Query column" ITEMLABELFONT="Arial or Courier or Times"
 ITEMLABELSIZE="Number of points" ITEMLABELORIENTATION="Horizontal or Vertical"
 QUERY="Query name" SCALEFROM="Integer minimum value"
 SCALETO="Integer maximum value" SHOWITEMLABELS="Yes or No"
 SHOWVALUELABEL="Yes or No or Rollover" TITLE="Title text"
 TITLEFONT="Arial or Courier or Times" TYPE="Bar or HorizontalBar"
 URL="URL string" URLCOLUMN="Query column"
 VALUECOLUMN="Query column" VALUELABELFONT="Arial or Courier or Times"
 VALUELABELSIZE="Number of points"
 VALUELOCATION="On bar or Over bar"> ... </CFGRAPH>
```

Syntax for dynamic line graphs:

```
<CFGRAPH BACKGROUNDCOLOR="Web color" BORDERCOLOR="Web color"
 BORDERWIDTH="Integer number of pixels" DEPTH="Integer number of pixels"
 FILEFORMAT="Flash or Jpg" FILL="Yes or No"
 GRAPHHEIGHT="Integer number of pixels" GRAPHWIDTH="Integer number of pixels"
 GRIDLINES="Integer number of lines" ITEMCOLUMN="Query column"
 ITEMLABELFONT="Arial or Courier or Times"
 ITEMLABELORIENTATION="Horizontal or Vertical" ITEMLABELSIZE="Number of points"
 LINECOLOR="Web color" LINEWIDTH="Integer number of pixels"
 QUERY="Query name" SCALEFROM="Integer minimum value"
 SCALETO="Integer maximum value" SHOWITEMLABELS="Yes or No"
 TITLE="Title text" TITLEFONT="Arial or Courier or Times"
 TYPE="Line" VALUECOLUMN="Query column"> ... </CFGRAPH>
```

Syntax for dynamic pie graphs:

```
<CFGRPH BACKGROUNDCOLOR="Web color" BORDERCOLOR="Web color"
 BORDERWIDTH="Integer number of pixels" COLORLIST="Web color list"
 DEPTH="Integer number of pixels" FILEFORMAT="Flash or Jpg"
 GRAPHHEIGHT="Integer number of pixels" GRAPHWIDTH="Integer number of pixels"
 ITEMCOLUMN="Query column" LEGENDFONT="Arial or Courier or Times"
 QUERY="Query name" SHOWLEGEND="Above or Below or Left or Right or None"
 SHOWVALUELABEL="Yes or No or Rollover" TITLE="Title text"
 TITLEFONT="Arial or Courier or Times" TYPE="Pie" URL="URL string"
 URLCOLUMN="Query column" VALUECOLUMN="Query column"
 VALUELABELFONT="Arial or Courier or Times" VALUELABELSIZE="Number of points"
 VALUELOCATION="Inside or Outside"> ... </CFGRAPH>
```

Syntax for hard-coded graphs:

```
<CFGRAPH TYPE=" ... " other attributes>
 <CFGRAPHDATA COLOR="Web color" ITEM="Data item label" VALUE="Data value"
  URL="URL string">
 ...
</CFGRAPH>
```

TABLE A.33 <CFGRAPH> ATTRIBUTES

Attribute	Description	Notes
BACKGROUNDCOLOR	Color for graph	Optional; can specify from the 256 standard Web colors in any valid HTML format. It defaults to WHITE.
BARSPACING	Integer number of pixels	Optional; spacing between bars in a vertical or horizontal bar chart. Defaults to 0.
BORDERCOLOR	Color of border	Optional; can specify from the 256 standard Web colors in any valid HTML format. Defaults to BLACK.
BORDERWIDTH	Number of pixels	Optional; integer. Defaults to 1. To remove border, set to 0.
COLORLIST	List of colors for each data point	Optional; can specify from the 256 standard Web colors in any valid HTML format. If there are more colors than datapoints, the list starts over.
DEPTH	Depth of 3D effect	Optional; integer depth in pixels. Defaults to 0.
FILEFORMAT	Type of output file	Optional; must be FLASH or JPG. Defaults to FLASH.

Attribute	Description	Notes
FILL	YES or NO	Optional; indicates whether to fill the area below the line (to produce an area chart). Defaults to NO.
GRAPHHEIGHT	Height of graph in pixels	Optional; defaults to 240.
GRAPHWIDTH	Width of graph in pixels	Optional; defaults to 320.
GRIDLINES	Number of lines in grid	Optional; defaults to 0.
ITEMCOLUMN	Query column	Optional; label of query column. Only used when graph is built from a query.
ITEMLABELFONT	Font name	Optional; font name for use with item labels. Must be ARIAL (default), COURIER, or TIMES.
ITEMLABELORIENTATION	Orientation of item labels	Optional; HORIZONTAL (default) or VERTICAL.
ITEMLABELSIZE	Size of item	Optional; point size of item labels. Defaults to 12.
LEGENDFONT	Font name	Optional; font name for use in the legend. Must be ARIAL (default), COURIER, or TIMES.
LINECOLOR	Color of lines	Optional; color for data lines in a line chart. Can specify from the 256 standard Web colors in any valid HTMLformat. Defaults to BLUE.
LINEWIDTH	Number of pixels	Optional; integer width of graph lines in a line chart. Defaults to 1.
QUERY	Query name	Optional; identifies query to build graph from. It is required if you're not using <CFGRAPHDATA> to create data items.
SCALEFROM	Minimum value for vertical axis	Optional; integer, defaults to 0. For line and bar charts only.
SCALETO	Maximum value for vertical axis	Optional integer; defaults to maximum data value. For line and bar charts only.
SHOWITEMLABELS	YES or NO	Optional; indicates whether labels are to be displayed on horizontal axis.

TABLE A.33 CONTINUED

Attribute	Description	Notes
SHOWLEGEND	ABOVE or BELOW or LEFT or RIGHT or NONE	Optional, identifies placement of legend that describes colors used in data labels. Defaults to LEFT.
SHOWVALUELABEL	YES, NO, or ROLLOVER	Optional, indicates whether values are to be displayed for data items or whether they are to be displayed only when the mouse is rolled over the data. Defaults to YES.
TITLE	Title text	Optional; chart title. Appears above or below the chart, depending on legend placement.
TITLEFONT	Font name	Optional; font name for chart title. Must be ARIAL (default), COURIER, or TIMES.
TYPE	PIE, LINE, BAR, or HORIZONTALBAR	Required; identifies type of graph to produce.
URL	URL	Optional; used only with FLASH format. URL to load when user clicks a data item. Data identified in the URLCOLUMN attribute is tacked onto the URL. Not available when TYPE=LINE.
URLCOLUMN	Query column	Optional, name of query column name to be added to URL when the URL attribute is specified. Not available when TYPE=LINE.
VALUECOLUMN	Query column name	Optional; name of query column containing the value to be plotted. Required if you are not using <CFGRAPHDATA> to create data points.
VALUELABELFONT	Font name	Optional; font name for use with values. Must be ARIAL (default), COURIER, or TIMES.
VALUELABELSIZE	Number of points	Optional; size of value font. Defaults to 12.

Attribute	Description	Notes
VALUELOCATION	Location code	Optional; code that identifies where value labels are to be placed. For bar type graphs, it can be either ON BAR or OVER BAR. Defaults to ON BAR. For pie graphs, it can be INSIDE or OUTSIDE. Default to INSIDE. Not used with line graphs.

TABLE A.34 <CFGRAPHDATA> ATTRIBUTES

Attribute	Description	Notes
COLOR	Color for data point	Optional; color for a data point. You can specify from the 256 standard Web colors in any valid HTML format. Default is to use colors specified in the <CFGRAPH> parent tag's COLORLIST attribute. This attribute is ignored in line graphs.
ITEM	Text label	Optional; label for the data point
URL	URL	Optional, used only with FLASH format. URL to load when user clicks a data item. Does not work with line charts.
VALUE	Number	Required; data point value.

Example: The example in Listing A.7 presents a simple pie chart in which the data points are hard-coded.

LISTING A.7 A PIE CHART PRODUCED FROM HARD-CODED VALUES

```
<CFGRAPH TYPE="pie" COLORLIST="pink,blue,red,yellow,orange"
 TITLE="Revenues By Genre">
 <CFGRAPHDATA ITEM="Drama" VALUE="85000">
 <CFGRAPHDATA ITEM="Comedy" VALUE="37500">
 <CFGRAPHDATA ITEM="Action" VALUE="125000">
 <CFGRAPHDATA ITEM="Horror" VALUE="95000">
 <CFGRAPHDATA ITEM="Sci-Fi" VALUE="95000">
</CFGRAPH>
```

The next example, shown in Listing A.8, demonstrates a horizontal bar graph built dynamically from a query.

LISTING A.8 A HORIZONTAL BAR CHART PRODUCED FROM HARD-CODED VALUES

```
<!--- Get the raw data from the database. --->
<CFQUERY NAME="GetExpenses" DATASOURCE="OWS">
  SELECT
   E.FilmID, F.MovieTitle,
   SUM(ExpenseAmount) AS TotalExp
  FROM Expenses E INNER JOIN Films F ON E.FilmID = F.FilmID
  GROUP BY E.FilmID, F.MovieTitle
  HAVING SUM(ExpenseAmount) > 10000
  ORDER BY SUM(ExpenseAmount) DESC
</CFQUERY>

<HTML>
<HEAD>
  <TITLE>Film Expense Analysis</TITLE>
</HEAD>
<BODY>

<H1>Film Expense Analysis</H1>

<!--- Bar graph, from Query of Queries --->
<CFGRAPH
  TYPE="HorizontalBar"
  QUERY="GetExpenses"
  VALUECOLUMN="TotalExp"
  ITEMCOLUMN="MovieTitle"
  TITLE="Total Expenses By Film"
  SCALETO="255000"
  ITEMLABELSIZE="10"
  VALUELABELSIZE="8"
  VALUELOCATION="over bar"
  FILEFORMAT="flash"
  DEPTH="2"
  COLORLIST="red,green,blue,yellow,orange"
>
</CFGRAPH>
</BODY>
</HTML>
```

See also: <CFREPORT>, <CFOBJECT>, <CFSERVLET>

<CFGRID>, <CFGRIDCOLUMN>, <CFGRIDROW>, </CFGRID>

Description: <CFGRID> embeds a Java grid control in your <CFFORM> forms. Grids are similar to spreadsheet-style interfaces, and <CFGRID> grids can be used to browse, select, and even edit data. Grids can be populated either by a query or by specifying each row using the <CFGRIDROW> tag. <CFGRIDCOLUMN> can be used to configure the individual columns with a grid. <CFGRID> attributes are presented in Table A.35. <CFGRIDCOLUMN> attributes are presented in Table A.36, and <CFGRIDROW> attributes are presented in Table A.37.

<CFGRID> must be used between <CFFORM> and </CFFORM> tags.

Syntax:

```
<CFGRID ALIGN="Alignment" APPENDKEY="Yes or No"
 AUTOWIDTH="Yes or No"
 BGCOLOR="Color" BOLD="Yes or No"
 COLHEADERALIGN="Alignment" COLHEADERBOLD="Yes or No" COLHEADERFONT="Font Face"
 COLHEADERFONTSIZE="Font Size" COLHEADERITALIC="Yes or No" COLHEADERS="Yes or No"
COLHEADERSTEXTCOLOR="Color specification"
 DELETE="Yes or No"
 DELETEBUTTON="Button Text" FONT="Font Face" FONTSIZE="Font Size"
 GRIDDATAALIGN="Alignment" GRIDLINES="Yes or No"
 HEIGHT="Control Height"
 HIGHLIGHTHREF="Yes or No" "Yes or No" HREF="URL" HREFKEY="Key"
 HSPACE="Horizontal Spacing"
 INSERT="Yes or No" INSERTBUTTON="Button Text"
 ITALIC="Yes or No" MAXROWS="Number" NAME="Field Name"
 NOTSUPPORTED="Non Java Browser Code" ONERROR="Error Function"
 ONVALIDATE="Validation Function" PICTUREBAR="Yes or No" QUERY="Query
 ROWHEADER="Yes or No" ROWHEADERALIGN="Alignment"
 ROWHEADERBOLD="Yes or No"
 ROWHEADERFONT="Font Face" ROWHEADERFONTSIZE="Font Size" ROWHEADERWIDTH="Width"
 ROWHEADERITALIC="Yes or No" ROWHEADERTEXTCOLOR="Color specification"
 SELECTCOLOR="Color" SELECTMODE="Mode" SORT="Yes or No"
 SORTASCENDINGBUTTON="Button Text" SORTDESCENDINGBUTTON="Button Text"
 TARGET="Target Window" TEXTCOLOR="Color specification" VSPACE="Vertical
➥Spacing" WIDTH="Control Width">

<CFGRIDCOLUMN BOLD="Yes or No"  BGCOLOR="Color specification"
 COLHEADERTEXTCOLOR="Color specification"
 DATAALIGN="Alignment"
 DISPLAY="Yes or No"
 FONT="Font Face" FONTSIZE="Font Size" HEADER="Header Text"
 HEADERALIGN="Alignment" HEADERBOLD="Yes or No" HEADERFONT="Font Face"
 HEADERFONTSIZE="Font Size" HEADERITALIC="Yes or No"
 HREF="URL" HREFKEY="Key"
 ITALIC="Yes or No" NAME="Column Name"
 NUMBERFORMAT="Format Mask"
 SELECT="Yes or No"
 TARGET="Target Window" TYPE="Type" WIDTH="Column Width"
➥TEXTCOLOR="Color specification"
 VALUES="Values list" VALUESDISPLAY="List of values"
 VALUESDELIMITER="Delimiter character"
>

<CFGRIDROW DATA="Data">
```

TABLE A.35 <CFGRID> ATTRIBUTES

Attribute	Description	Notes
ALIGN	Control alignment	Optional; possible values are TOP, LEFT, BOTTOM, BASELINE, TEXTTOP, ABSBOTTOM, MIDDLE, ABSMIDDLE, and RIGHT.

TABLE A.35 CONTINUED

Attribute	Description	Notes
APPENDKEY	YES or NO	Optional; appends item key to URL. If the value is YES, a variable named GRIDKEY is appended to the URL containing the item selected. It defaults to YES.
AUTOWIDTH	YES or NO	Optional; sizes grid based on the width of the data, within the HEIGHT and WIDTH attributes.
BGCOLOR	Background color	Optional; possible values are BLACK, CYAN, DARKGRAY, GRAY, LIGHTGRAY, MAGENTA, ORANGE, PINK, WHITE, YELLOW, or any color specified in RGB form.
BOLD	YES or NO	Optional; boldfaces grid control text; defaults to NO.
COLHEADERALIGN	Column header alignment	Optional; can be LEFT, CENTER, or RIGHT; default is LEFT.
COLHEADERBOLD	YES or NO	Optional; boldfaces column header text; defaults to NO.
COLHEADERFONT	Column header font	Optional font to use for column header.
COLHEADERFONTSIZE	Column header font size in points	Optional font size to use for column header.
COLHEADERITALIC	YES or NO	Optional; italicizes column header text; defaults to NO.
COLHEADERS	YES or NO	Optional; displays column headers; default is YES.
COLHEADERSTEXTCOLOR	Color for text	Optional; can be specified as BLACK (default), MAGENTA, CYAN, ORANGE, DARKGRAY, PINK, GRAY, WHITE, LIGHTGRAY, or YELLOW. It can also be specified as a hex value, such as ##999999 (a hex value preceded by two pound signs).
DELETE	YES or NO	Optional; if YES, allows records to be deleted from the grid. Default is NO.

Attribute	Description	Notes
DELETEBUTTON	Delete button text	Optional text to use for the Delete button;default is Delete.
FONT	Font face	Optional font face to use.
FONTSIZE	Font size	Optional font size.
GRIDDATAALIGN	Data alignment	Data alignment; can be LEFT, RIGHT, or CENTER; can be overridden at the column level.
GRIDLINES	YES or NO	Optional; displays grid lines; default is YES.
HEIGHT	Control height	Optional height in pixels.
HIGHLIGHTHREF	YES or NO	Optional attribute; if Yes, links are highlighted and underlined; defaults to YES.
HREF	URL	Optional URL to go to upon item selection; if populated by a query, this can be a query column.
HREFKEY	Primary key column	Optional name of column to be used as the primary key.
HSPACE	Control horizontal spacing	Optional horizontal spacing in pixels.
INSERT	YES or NO	Optional; if YES, allows records to be added to the grid; default is NO.
INSERTBUTTON	Insert button text	Optional text to use for the Insert button; default is INSERT.
ITALIC	Italic face text	Optional attribute; must be YES or NO if specified; defaults to NO.
MAXROWS	Number	Optional number of rows you want to show in the grid.
NAME	Unique control	Required.
NOTSUPPORTED	Text to be used for non-Java browsers	Optional text (or HTML code) to be displayed on non–Java-capable browsers.
ONERROR	JavaScript error function	Optional override to your own JavaScript error message function.

TABLE A.35 CONTINUED

Attribute	Description	Notes
ONVALIDATE	JavaScript validation function	Optional override to your own JavaScript validation function.
PICTUREBAR	Displays picture bar with icons	Optional; if YES, a button bar with icons is displayed for insert, delete, and sort; default is NO.
QUERY	Query to populate grid	Optional name of query to be used to populate the grid.
ROWHEADER	YES or NO	Optional; displays row header if YES; default is YES.
ROWHEADERALIGN	Row header alignment	Optional attribute; can be LEFT, CENTER, or RIGHT; default is LEFT.
ROWHEADERBOLD	YES or NO	Optional; displays row header in bold font; defaults to NO.
ROWHEADERFONT	Row header font	Optional font to use for row header.
ROWHEADERFONTSIZE	Row header font size	Optional font size to use for row header.
ROWHEADERITALIC	YES or NO	Optional; displays row header in italics; default is NO.
ROWHEADERTEXTCOLOR	Color for text	Optional; can be specified as BLACK (default), MAGENTA, CYAN, ORANGE, DARKGRAY, PINK, GRAY, WHITE, LIGHTGRAY, or YELLOW. Can also be specified as a hex value, such as ##999999 (a hex value preceded by two pound signs).
ROWHEADERWIDTH	Row header width	Optional row header width in pixels.
ROWHEIGHT	Row height	Optional height of row in pixels.
SELECTCOLOR	Selection color	Optional attribute; possible values are BLACK, CYAN, DARKGRAY, GRAY, LIGHTGRAY, MAGENTA, ORANGE, PINK, WHITE, YELLOW, or any color specified in RGB form.

Attribute	Description	Notes
SELECTMODE	Selection mode	Optional attribute; can be EDIT SINGLE ROW COLUMN or BROWSE; default is BROWSE.
SORT	YES or NO	Optional; if YES, allows grid data to be sorted; defaults to NO.
SORTASCENDINGBUTTON	Sort ascending button text	Optional text to use for the sort ascending button; default is A -> Z.
SORTDESCENDINGBUTTON	Sort descending button text	Optional text to use for the sort descending button; default is Z -> A.
TARGET	URL to target window	Optional name of target window for HREF URL.
TEXTCOLOR	Color for text	Optional; can be specified as BLACK (default), MAGENTA, CYAN, ORANGE, DARKGRAY, PINK, GRAY, WHITE, LIGHTGRAY, or YELLOW. Can also be specified as a hex value, such as ##999999 (a hex value preceded by two pound signs).
VSPACE	Control vertical spacing	Optional vertical spacing in pixels.
WIDTH	Control width	Optional width in pixels.

TABLE A.36 <CFGRIDCOLUMN> ATTRIBUTES

Attribute	Description	Notes
BGCOLOR	Color for text	Optional; can be specified as BLACK (default), MAGENTA, CYAN, ORANGE, DARKGRAY, PINK, GRAY, WHITE, LIGHTGRAY, or YELLOW. Can also be specified as a hex value, such as ##999999 (a hex value preceded by two pound signs).
BOLD	Boldface text	Optional; must be YES or NO if specified; defaults to NO.

TABLE A.36 CONTINUED

Attribute	Description	Notes
COLHEADERTEXTCOLOR	Color for text	Optional; can be specified as BLACK (default), MAGENTA, CYAN, ORANGE, DARKGRAY, PINK, GRAY, WHITE, LIGHTGRAY, or YELLOW. Can also be specified as a hex value, such as ##999999 (a hex value preceded by two pound signs).
DATAALIGN	Data alignment	Optional; can be LEFT, CENTER, or RIGHT; default is LEFT.
DISPLAY	Display column	Optional; if NO, column is hidden; default is YES.
FONT	Font face	Optional font face to use.
FONTSIZE	Font size	Optional font size.
HEADER	Header text	Optional header text; defaults to column name. Value is significant only when the <CFGRID> COLHEADERS attribute is YES (default).
HEADERALIGN	Header alignment	Optional; can be LEFT, CENTER, or RIGHT; default is LEFT.
HEADERBOLD	Header in bold	Optional; if YES, header is displayed in a bold font; default is NO.
HEADERFONT	Header font	Optional font to use for header.
HEADERFONTSIZE	Header font size	Optional font size to use for header.
HEADERITALIC	Header in italics	Optional; if YES, header is displayed in an italic font; default is NO.
HREF	URL	URL for selection in this column; can be absolute or relative.
HREFKEY	Primary key	Optional primary key to use for this column.
ITALIC	Italic face text	Optional; must be YES or NO if specified; defaults to NO.

Attribute	Description	Notes
NAME	Column name	Required; if using a query to populate the grid, this must be a valid column name.
NUMBERFORMAT	Number formatting	Optional; uses NumberFormat() function masks; see that function for mask details.
SELECT	Allow selection	Optional; if NO, selection or editing is not allowed in this column.
TARGET	Target window	Optional target window for HREF.
TEXTCOLOR	Color for text	Optional; can be specified as BLACK (default), MAGENTA, CYAN, ORANGE, DARKGRAY, PINK, GRAY, WHITE, LIGHTGRAY, or YELLOW. Can also be specified as a hex value, such as ##999999 (a hex value preceded by two pound signs).
TYPE	Data type	Optional; can be image or numeric. If it's an image, an appropriate graphic is displayed for the cell value. Built-in images are in the following bulleted list.
VALUES	List or range	Optional; enables you to format a column as a drop-down box. You specifiy a hard-coded list of delimited values (for example, Joe,Bob,Jenny) or a range of values, such as 1-10.
VALUESDISPLAY	List to dislay	Optional; this is used with the VALUES attribute to specify the list to be displayed.
VALUESDELIMITER	Delimiter character	Optional; character to act as delimiter in VALUES and VALUESDISPLAY lists. Defaults to ,.
WIDTH	Column width	Optional column width in pixels. Columns are sized as wide as the longest value by default.

The following lists the built-in image types for use in the TYPE attribute:

- Image
- CD
- Computer
- Document
- Element
- Folder
- Floppy
- Fixed
- Remote

TABLE A.37 <CFGRIDROW> ATTRIBUTES

Attribute	Description	Notes
DATA	Row data	Comma-delimited list of data to be displayed; one item for each column in the grid

Example: The example in Listing A.9 creates two grids based on query results. The first query is then used to produce a read-only grid containing film budget information. The second query is used to produce a grid in which the user can edit and delete expense records.

LISTING A.9 DISPLAYING QUERY RESULTS WITH <CFGRID>

```
<CFQUERY NAME="GetBigBudgetFlicks" DATASOURCE="ows">
   SELECT FilmID, MovieTitle, AmountBudgeted
   FROM Films
</CFQUERY>

<CFQUERY NAME="GetExpenses" DATASOURCE="ows">
   SELECT E.ExpenseID, E.FilmID, F.MovieTitle, E.ExpenseAmount, E.Description
   FROM Expenses E INNER JOIN Films F ON E.FilmID = F.FilmID
</CFQUERY>

<HTML>
<HEAD>
   <TITLE>CFGRID Example</TITLE>
</HEAD>

<BODY>
<CFFORM ACTION="CFGRID_Action.cfm" METHOD="POST" NAME="GridForm">

<H3>Films Budgets</H3>
<CFGRID
   NAME="FilmBudgets"
   QUERY="GetBigBudgetFlicks"
   COLHEADERBOLD="Yes"
```

```
   COLHEADERFONT="Gill Sans MT"
   SELECTMODE="BROWSE"
   WIDTH="400"
   HEIGHT="300"
>
   <CFGRIDCOLUMN NAME="FilmID" HEADER="Film ID">
   <CFGRIDCOLUMN NAME="MovieTitle" HEADER="Title">
   <CFGRIDCOLUMN
      NAME="AmountBudgeted"
      NUMBERFORMAT="___,___,___,___.__"
      DATAALIGN="RIGHT"
      HEADER="Budget"
      HEADERALIGN="RIGHT"
   >
</CFGRID>

<h3>Film Expenses</H3>
<!--- We'll make this table editable using SELECTMODE
 and the SELECT attribute in the CFGRIDCOLUMN tags --->
<CFGRID
   NAME="FilmExpenses"
   QUERY="GetExpenses"
   COLHEADERBOLD="Yes"
   COLHEADERFONT="Gill Sans MT"
   SELECTMODE="EDIT"
   WIDTH="580"
   HEIGHT="300"
   DELETE="Yes"
   DELETEBUTTON="Del?"
>
   <!--- The key column is included so that we can
    update, but is not displayed to user. --->
   <CFGRIDCOLUMN NAME="ExpenseID" DISPLAY="No">
   <CFGRIDCOLUMN SELECT="Yes" NAME="FilmID" HEADER="Film ID">
   <CFGRIDCOLUMN SELECT="Yes" NAME="MovieTitle" HEADER="Title">
   <CFGRIDCOLUMN
      NAME="ExpenseAmount"
      SELECT="Yes"
      NUMBERFORMAT="___,___,___,___.__"
      DATAALIGN="RIGHT"
      HEADER="Amount"
      HEADERALIGN="RIGHT"
   >
   <CFGRIDCOLUMN SELECT="yes" NAME="Description">
</CFGRID>

<P><INPUT TYPE="Submit" VALUE="Save">
</CFFORM>

</BODY>
</HTML>
```

The action template (`CFGRID_ACTION.CFM`) for the form developed in the previous example is presented in the following example for `<CFGRIDUPDATE>`.

For more information about using `<CFGRID>`, see Chapter 25, "Enhancing Forms with Client-Side Java."

Note

The <CFGRID> control is accessible only by users with Java-enabled browsers.

See also: <CFFORM>, <CFGRIDUPDATE>, <CFINPUT>, <CFSELECT>, <CFSLIDER>, <CFTEXTINPUT>, <CFTREE>

<CFGRIDUPDATE>

Description: <CFGRIDUPDATE> provides the action backend to support <CFGRID> in edit mode. <CFGRIDUPDATE> performs all inserts, deletes, and updates in one simple operation. <CFGRIDUPDATE> can be used only in an action page to which a form containing a <CFGRID> control was submitted. <CFGRIDUPDATE> attributes are listed in Table A.38.

Syntax:

```
<CFGRIDUPDATE DATASOURCE="ODBC Data Source Name" DBNAME="database name"
 DBPOOL="pool" DBTYPE="type" DBSERVER="dbms" GRID="Grid Name"
 KEYONLY="Yes or No""Yes or No"
 PASSWORD="Password" PROVIDER="provider" PROVIDERDSN="data source"
 TABLENAME="Table Name" TABLEOWNER="Table Owner Name"
 TABLEQUALIFIER="Table Qualifier" USERNAME="User Name">
```

TABLE A.38 <CFGRIDUPDATE> ATTRIBUTES

Attribute	Description	Notes
DATASOURCE	ODBC data source	Required
DBNAME	Sybase database name	Optional; used only if using native Sybase drivers
DBPOOL	Database connection pool name	Optional database pool name
DBSERVER	Database server	Optional database server; only used if using native database drives
DBTYPE	Database type	Optional type; defaults to ODBC; other values are Oracle73, Oracle80, and Sybase11
GRID	Grid name	Required; the name of the grid in the submitted form with which to update the table
KEYONLY	WHERE clause construction	If YES, the WHERE clause generated by <CFGRID> contains just the primary; default is YES

Attribute	Description	Notes
PASSWORD	ODBC login password	Optional ODBC login password
PROVIDER	OLE-DB COM provider	Optional; only used if using OLE-DB
PROVIDERDSN	Data source OLE-DB COM provider	Optional; only used if using OLE-DB
TABLENAME	Table name	Required
TABLEOWNER	Table owner	Optional ODBC table owner
TABLEQUALIFIER	Table qualifier	Optional ODBC table qualifier
USERNAME	ODBC username	Optional ODBC username

Example: The following example updates the `Expenses` table based on a grid in the calling form named `FilmExpenses`. See the example previously in `<CFGRID>`:.

```
<CFGRIDUPDATE DATASOURCE="OWS" TABLENAME="Expenses" GRID="FilmExpenses">
```

For more information about using `<CFGRIDUPDATE>`, see Chapter 23, "Improving the User Experience."

See also: `<CFFORM>`, `<CFGRID>`

<CFHEADER>

Description: `<CFHEADER>` enables you to control the contents of specific HTTP headers. You can either provide values for HTTP header elements or specify an HTTP response code and text. The attributes for this tag are presented in Table A.39.

Syntax:

```
<CFHEADER NAME="Header Name" VALUE="Value">
```

or

```
<CFHEADER STATUSCODE="HTTP code number" STATUSTEXT="Explanation of code">
```

TABLE A.39 `<CFHEADER>` ATTRIBUTES

Attribute	Description	Notes
NAME	Name for header to be set	Required if you're not supplying a STATUSCODE
VALUE	Header value	Used in conjunction with NAME attribute

TABLE A.39 CONTINUED

Attribute	Description	Notes
STATUSCODE	HTTP number code	Required if you're not specifying a NAME
STATUSTEXT	Text that explains the STATUSCODE	Required

Example: The following example sets several header values to prevent the template from being cached:

```
<CFHEADER NAME="Pragma" VALUE="no-cache">
<CFHEADER NAME="Expires" VALUE="0">
<CFHEADER NAME="cache-control" VALUE="no-cache, no-store,
  must-revalidate, max-age=0">
<CFHTMLHEAD TEXT='<META HTTP-EQUIV="Expires"
 CONTENT="Mon, 01 Jan 2001 00:00:01 GMT">'>
```

Note

There is usually little need to use <CFHEADER> because ColdFusion sets the HTTP headers automatically to optimum values.

Note

Because <CFHEADER> needs to write to the HTTP header, you cannot use it if you've already flushed the header from the ColdFusion output buffer with <CFFLUSH>.

<CFHTMLHEAD>

Description: <CFHTMLHEAD> writes text into the header section of your Web page. It can be placed anywhere in your page, effectively enabling you to write your <HEAD> section from in or below the <BODY> section of a page. Attributes for this tag are presented in Table A.40.

Syntax:

```
<CFHTMLHEAD TEXT="Text">
```

TABLE A.40 <CFHTMLHEAD> ATTRIBUTES

Attribute	Description	Notes
TEXT	Text to place in <HEAD>	Required

Note

Because <CFHTMLHEAD> needs to write to the HTML <HEAD>, you cannot use it if you've already flushed the <HEAD> section from the ColdFusion output buffer with <CFFLUSH>.

Example: See <CFHEADER> for example.

<CFHTTP>, <CFHTTPPARAM>, </CFHTTP>

Description: <CFHTTP> enables you to process HTTP GET and POST requests within your ColdFusion code, making ColdFusion perform like a browser. If you're using the POST method, parameters can be passed using the <CFHTTPPARAM> tag. <CFHTTPPARAM> can be used only between <CFHTTP> and </CFHTTP> tags.

<CFHTTP> attributes are listed in Table A.41, and <CFHTTPPARAM> attributes are listed in Table A.42. <CFHTTP> sets special variables upon completion that you can inspect; they are listed in Table A.43.

Syntax:

```
<CFHTTP COLUMNS="Column Names" DELIMITER="Delimiter Character" FILE="File Name"
 METHOD="Get|Post" NAME="Query Name" PASSWORD="Password" PATH="Directory"
 PORT="Port number" PROXYSERVER="Host Name" RESOLVEURL="Yes or No""Yes or No"
 TEXTQUALIFIER="Text Qualifier"
 URL="Host Name" USERNAME="User Name">...</CFHTTP>

<CFHTTPPARAM FILE="File Name" NAME="Field Name" TYPE="Type" VALUE="Value">
```

TABLE A.41 <CFHTTP> ATTRIBUTES

Attribute	Description	Notes
COLUMNS	Query columns	Optional; queries columns for retrieved data.
DELIMITER	Column delimiter	Required if NAME is used; default delimiter is a comma.
FILE	Filename	Required only if PATH is used; file to save.
METHOD	Submission method	Required; must be either GET or POST; use POST to use <CFHTTPPARAM>.
NAME	Query name	Optional; name of query to be constructed with HTTP results.
PASSWORD	User password	Optional; user password if required by server.
PATH	File path	Optional; path to save file if method is POST.
PORT	TCP/IP port number	Optional; defaults to 80.
PROXYSERVER	Server name	Optional name of proxy server to use.
PROXYPORT	TCP/IP port number	Optional; defaults to 80.
REDIRECT	YES or NO	Optional; defaults to NO; indicates whether execution is to be redirected upon tag failure.
RESOLVEURL	Resolve URL	Optional; defaults to NO; if YES, fully resolves embedded URLs.

TABLE A.41 CONTINUED

Attribute	Description	Notes
TEXTQUALIFIER	Text qualifier	Required if NAME is used; delimiter indicating start and end of column.
TIMEOUT	Timeout period in seconds	Optional; timeout period can be defined in the browser URL, the CF Administrator, and this tag.
THROWONERROR	YES or NO	Optional; defaults to NO; indicates whether an exception should be thrown on an error; enables you to trap an error with a <CFTRY> <CFCATCH> block. The error codes are found in the CFHTTP.StatusCode variable.
URL	Host URL	Required; must be DNS name or IP address of a valid host.
USERAGENT	User agent request	Optional; enables your <CFHTTP> call to spoof a specific browser.
USERNAME	Username	Optional; username if required by server.

TABLE A.42 <CFHTTPPARAM> ATTRIBUTES

Attribute	Description	Notes
FILE	Filename	Required if TYPE is File.
NAME	Field name	This attribute is required.
TYPE	Field type	This attribute is required; must be URL, FORMFIELD, COOKIE, CGI, or FILE.
VALUE	Field value	This attribute is optional unless TYPE is File.

TABLE A.43 <CFHTTP> RETURNED VARIABLES

Field	Description
#CFHTTP.FILECONTENT#	Content returned by HTTP request.
#CFHTTP.MIMETYPE#	MIME type of returned data.
#CFHTTP.RESPONSEHEADER#	Response header name/value pair(s). If there are more than one, they're returned in an array.
#CFHTTP.HEADER#	Raw response header.
#CFHTTP.STATUSCODE#	HTTP error code associated with the error that occurs if THROWONERROR is set to YES.

Example: This example uses Altavista.com's Babblefish to translate the expression "Say 'Hello' in French" into French. It does this by mimicking the Altavista.com form found at `http://world.altavista.com/`. It will work as long as Altavista.com doesn't change the URLs or names of fields in this form.

The example in Listing A.10 uses a `<CFHTTP>` call with `METHOD` set to `POST`, passing the parameters on to the form's action page (identified in the `URL` attribute as `http://world.altavista.com/tr`) using calls to `<CFHTTPPARAM>`.

The action page is itself a form and is returned in the `CFHTTP.FileContent` variable. This variable is parsed looking for two values that were determined to delimit the translated value in the action page.

LISTING A.10 USING <CFHTTP> WITH AN HTML FORM

```
<P>Say 'Hello' in French.
<CFTRY>
   <CFHTTP
      METHOD="POST"
      URL="http://world.altavista.com/tr"
      THROWONERROR="Yes"
   >
      <CFHTTPPARAM TYPE="FORMFIELD" NAME="doit" VALUE="done">
      <CFHTTPPARAM TYPE="FORMFIELD" NAME="tt" VALUE="urltext">
      <CFHTTPPARAM TYPE="FORMFIELD" NAME="lp" VALUE="en_fr">
      <CFHTTPPARAM TYPE="FORMFIELD" NAME="urltext"
       ➥VALUE="Say 'Hello' in French.">
   </CFHTTP>

   <CFCATCH TYPE="Any">
      <CFOUTPUT>#cfhttp.StatusCode#</CFOUTPUT>
      <P>Failure!<CFABORT>
   </CFCATCH>
</CFTRY>

<P>Success!

<!--- Now parse the French out of the action page's form.
 The value we're looking for is inside a textarea named 'q'
 in the action page. --->
<CFSET nStart=Find('name="q"', CFHTTP.FileContent) +9>
<CFSET nEnd=Find('</textarea>', CFHTTP.FileContent, nStart+1) >
<CFSET French=Mid(CFHTTP.FileContent, nStart, nEnd-nStart)>

<P><EM><CFOUTPUT>#French#</CFOUTPUT></EM>
```

See also: `<CFFTP>`

<CFIF>, <CFELSEIF>, <CFELSE>, </CFIF>

Description: The `<CFIF>` set of tags is used to provide conditional branching logic (along with `<CFSWITCH>`, `<CFCASE>`, and related tags).

Every <CFIF> tag must have a matching </CFIF> tag. The <CFELSEIF> and <CFELSE> tags are entirely optional. You can use as many <CFELSEIF> tags as necessary in a <CFIF> statement, but only one <CFELSE>. <CFELSE> must always be the last compare performed if it is used.

<CFIF> uses operators to compare values. These are presented in Table A.44. Conditions can also be combined to perform more complex comparisons using the Boolean operators presented in Table A.45.

You can compare any values, including static text and numbers, ColdFusion fields, database column values, and function results.

Syntax:

```
<CFIF Condition><CFELSEIF Condition><CFELSE></CFIF>
```

TABLE A.44 COLDFUSION CONDITIONAL OPERATORS

Operator	Alternative	Description
IS	EQUAL, EQ	Checks that the right value is equal to the left value
IS NOT	NOT EQUAL, NEQ	Checks that the right value is not equal to the left value
CONTAINS		Checks that the right value is contained within the left value
DOES NOT CONTAIN		Checks that the right value is not contained within the left value
GREATER THAN	GT	Checks that the left value is greater than the right value
LESS THAN	LT	Checks that the left value is less than the right value
GREATER THAN OR EQUAL	GTE	Checks that the left value is greater than or equal to the right value
LESS THAN OR EQUAL	LTE	Checks that the left value is less than or equal to the right value

TABLE A.45 COLDFUSION BOOLEAN OPERATORS

Operator	Description
AND	Conjunction; returns TRUE only if both expressions are true
OR	Disjunction; returns TRUE if either expression is true
NOT	Negation

This first example checks to see whether a FORM variable named LastName exists:

```
<CFIF IsDefined("FORM.LastName")>
   <CFSET Lname=FORM.LastName)>
```

```
<CFELSE>
   <CFSET Lname="">
</CFIF>
```

The following example checks to see whether both the `FirstName` and `LastName` `FORM` variables exist:

```
<CFIF (IsDefined("FORM.FirstName")) AND (IsDefined("FORM.LastName"))>
...
```

You could use the following to check for either a first name or a last name:

```
<CFIF (IsDefined("FORM.FirstName")) OR (IsDefined("FORM.LastName"))>
...
```

Often, you will want to verify that a field is not empty and that it does not contain blank spaces. The following example demonstrates how this can be accomplished:

```
<CFIF Trim(FORM.LastName) IS NOT "">
...
```

You can use the `CONTAINS` operator to check whether a value is within a range of values. Take a look at both of these examples:

```
<CFIF "KY,MI,MN,OH,WI" CONTAINS State>
...
```

```
<CFIF TaxableStates CONTAINS State>
...
```

More complex expressions can be created by combining conditions within parentheses. For example, the following condition checks to see whether payment is by check or credit card; if payment is by credit card, it checks to ensure that there is an approval code:

```
<CFIF (PaymentType IS "Check") OR ((PaymentType IS "Credit")
 AND (ApprovalCode IS NOT ""))>
...
```


The following example is a complete conditional statement that uses `<CFELSEIF>` to perform additional comparisons. It also uses `<CFELSE>` to specify a default for values that pass none of the compares:

```
<CFIF State IS "MI">
 Code for Michigan only goes here
<CFELSEIF State IS "IN">
 Code for Indiana only goes here
<CFELSEIF (State IS "OH") OR (State IS "KY")>
 Code for Ohio or Kentucky goes here
<CFELSE>
 Code for all other states goes here
</CFIF>
```

Note that `<CFPARAM>` can be used as a shortcut for this functionality. The code

```
<CFPARAM NAME="MyVariable" DEFAULT="123">
```

is the same as this code:

```
<CFIF NOT IsDefined("MyVariable")>
```

```
   <CFSET MyVariable=123>
</CFIF>
```

For more information about the <CFIF> tags, Chapter 10, "CFML Basics."

See also: <CFSWITCH>, <CFPARAM>, Iif()

<CFIMPERSONATE>, </CFIMPERSONATE>

Description: <CFIMPERSONATE> enables you to make one user impersonate another user defined in a security context that is in turn defined using ColdFusion Advanced Security. This impersonation is extended to execute the code you put between the start and end portions of the tag. The impersonation follows either the user within the operating system or the application. The TYPE attribute enables you to make the distinction. Table A.46 lists the attributes for <CFIMPERSONATE>.

Syntax:

```
<CFIMPERSONATE SECURITYCONTEXT="SecurityContext" USERNAME="Name"
 PASSWORD="Password" TYPE="CF or OS">
  ...
  HTML or CFML code to execute
  ...
</CFIMPERSONATE>
```

TABLE A.46 <CFIMPERSONATE> ATTRIBUTES

Attribute	Description	Notes
SECURITYCONTEXT	Context name	Required; predefined value in ColdFusion Advanced Security
USERNAME	Name	Required; names the user you want the current user to impersonate
PASSWORD	Password	Required; the password of the user you want to impersonate
TYPE	CF or OS	Required

Example: The following code verifies that a user is authenticated and then enables her to briefly appear as an administrative user:

```
<CFIF NOT IsAuthenticated()>
  <CFTRY>
  <CFAUTHENTICATE SECURITYCONTEXT="ReadOnly" USERNAME="#FORM.UName#"
   PASSWORD="#FORM.PW#">
  <CFCATCH TYPE="Security">
    <CFABORT SHOWERROR="You cannot be authenticated. Please use the
    ➥Back button and try again.">
  </CFCATCH>
  </CFTRY>
<CFELSE>
```

```
  <CFIMPERSONATE SECURITYCONTEXT="ReadWrite" USERNAME="Administrator"
   PASSWORD="#AdminPW#">
</CFIF>
```

See also: <CFAUTHENTICATE>

<CFINCLUDE>

Description: <CFINCLUDE> includes the contents of another template in the one being processed. This is one mechanism ColdFusion developers can employ to reuse code. The attribute is described in Table A.47.

Syntax:

```
<CFINCLUDE TEMPLATE="Template File Name">
```

TABLE A.47 <CFINCLUDE> ATTRIBUTES

Attribute	Description	Notes
TEMPLATE	Name of template to include	This attribute is required. Only relative paths are supported.

Example: The following example includes the footer file in the current directory if it exists and a default footer if not:

```
<CFIF FileExists("FOOTER.CFM")>
 <CFINCLUDE TEMPLATE="FOOTER.CFM">
<CFELSE>
 <CFINCLUDE TEMPLATE="/DEFAULT/FOOTER.CFM">
</CFIF>
```

Tip

<CFINCLUDE> can help you reuse templates. You can use <CFINCLUDE> to break out common components (such as page headers and footers or commonly used queries), which enables you to share them among multiple templates.

Caution

You need to be careful about defining variables in templates that are being included in other templates because their variables share the same scope. Therefore, if the included template defines a variable that is previously defined in the *including* template, the original variable value will be overwritten. Note that ColdFusion custom tags enable you to avoid these problems and reuse code. See Chapter 22, "Building Reusable Components," for more details.

See also: <CFLOCATION>

<CFINDEX>

Description: <CFINDEX> is used to populate Verity collections with index data. A collection must be created with either the ColdFusion Administrator or <CFCOLLECTION> before it can be populated. <CFINDEX> can be used to index physical files (in which case, the filename is returned in searches) or query results (in which case the primary key is returned in searches). <CFINDEX> attributes are listed in Table A.48. Values for the ACTION attribute are described in Table A.49.

Syntax:

```
<CFINDEX ACTION="Action" BODY="Text" COLLECTION="Collection Name" CUSTOM1="Data"
 CUSTOM2="Data" EXTENSIONS="File Extensions" EXTERNAL=""Yes or No"  KEY="Key"
 LANGUAGE="Language from optional International Search Pack"
 QUERY="Query Name" RECURSE="Yes or No" TITLE="Text" TYPE="Type" URLPATH="Path">
```

TABLE A.48 <CFINDEX> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action	Required attribute.
BODY	Body to index	Required if TYPE is CUSTOM; invalid if TYPE is DELETE. ASCII text to be indexed. If indexing a query, this must be the column (or comma-delimited list of columns) to be indexed.
COLLECTION	Collection name	Required; name of the collection to be indexed; if using external collections, this must be a fully qualified path to the collection.
CUSTOM1	Custom data	Optional attribute for storing data during indexing; specify a valid query column name.
CUSTOM2	Custom data	Optional attribute for storing data during indexing; usage is the same as CUSTOM1.
EXTENSIONS	File extensions	Optional; comma-delimited list of extensions of files to be indexed; only used if TYPE is PATH.
EXTERNAL	External collection	Optional; must be YES if indexing a collection created outside ColdFusion using Verity's native functions.
KEY	Unique key	Optional; used to indicate what makes each record unique. If TYPE is FILE, this should be the document filename; if TYPE is PATH, this should be a full path to the document; if TYPE is CUSTOM, this should be any unique identifier (for example, key field in a query result).

Attribute	Description	Notes
LANGUAGE	Specify a language	Optional; requires installation of the International Search Pack.
QUERY	Query name to be indexed	Optional; use when TYPE=CUSTOM.
RECURSE	YES or NO	Optional; used when TYPE=PATH; if it's YES then all subdirectories are indexed, too.
TITLE	Document title	Required if TYPE is Custom; specified title for collection or query column name. Enables searching collections by title and the display of a title other than the actual key.
TYPE	Index type	Optional attribute, must be FILE, PATH, or CUSTOM.
URLPATH	URL path	Optional attribute; specifies the URL path for files when TYPE=File or TYPE=Path.

TABLE A.49 <CFINDEX> ACTIONS

Action	Description
DELETE	Deletes a key from a collection
OPTIMIZE	Optimizes a collection
PURGE	Clears all data from a collection
REFRESH	Clears all data from a collection and repopulates it
UPDATE	Updates a collection and adds a key if it does not exist

Example: The first example updates an existing collection built from a query. This index enables users to search for Orange Whip Studios merchandise based on merchandise name, description, or film name:

```
<CFQUERY NAME="GetFilmsMerchandise" DATASOURCE="ows">
   SELECT M.MerchID, F.MovieTitle, M.MerchName, M.MerchDescription
   FROM Merchandise M, Films F
   WHERE M.FilmID = F.FilmID
</CFQUERY>
<CFINDEX
   ACTION="UPDATE"
   BODY="MerchName, MerchDescription, MovieTitle"
   COLLECTION="FilmsMerchandise"
   KEY="MerchID"
   QUERY="GetFilmsMerchandise"
   TYPE="CUSTOM"
>
```

The second example creates a collection and populates it with the contents of documents in a specific path:

```
<CFCOLLECTION ACTION="CREATE" COLLECTION="CFInstallTest"
 PATH="C:\CFUSION\Verity\Collections">
<CFINDEX
   ACTION="UPDATE"
   TYPE="PATH"
   COLLECTION="CFInstallTest"
   EXTENSIONS=".htm, .html, .cfm"
   KEY="C:\Inetpub\wwwroot\CFDOCS\testinstallation"
   RECURSE="Yes"
   URLPATH="http://localhost/CFDOCS/testinstallation"
>
```

For more information about using <CFINDEX> and Verity collections, see Chapter 29, "Online Commerce."

See also: <CFCOLLECTION>, <CFSEARCH>

<CFINPUT>

Description: <CFINPUT> is an enhancement to the standard HTML <INPUT> tag. <CFINPUT> enables you to automatically embed JavaScript client-side validation code in your HTML forms. <CFINPUT> must be used between <CFFORM> and </CFFORM> tags; it is not a Java control. Attributes are presented in Table A.50.

Syntax:

```
<CFINPUT CHECKED MAXLENGTH="Length" MESSAGE="Message Text" NAME="Field Name"
 ONERROR="JavaScipt Error Function" ONVALIDATE="JavaScript Validation Function"
 PASSTHROUGH="HTML attributes"
 RANGE="Range Values" REQUIRED="Yes or No" "Yes or No"
 SIZE="Field Size" TYPE="Type"
 VALIDATE="Validation Type" VALUE="Initial Value">
```

TABLE A.50 <CFINPUT> ATTRIBUTES

Attribute	Description	Notes
CHECKED	Checked state	Optional; only valid if type is RADIO or CHECKBOX and if present radio button or check box is prechecked
MAXLENGTH	Maximum number of characters	Optional
MESSAGE	Validation failure message	Optional message to display upon validation failure
NAME	Unique control name	Required
ONERROR	JavaScript error function	Optional override to your own JavaScript error message function
ONVALIDATE	JavaScript validation function	Optional override to your own JavaScript validation function
PASSTHROUGH	HTML attributes	Optional; for including HTML attributes not explicitly supported by <CFINPUT>

Attribute	Description	Notes
RANGE	Range minimum and maximum	Optional range for numeric values only; must be specified as two numbers separated by a comma
REQUIRED	YES or NO	Optional; indicates that value must be supplied; defaults to NO
SIZE	Field size	Optional number of characters to display before needing horizontal scrolling
TYPE	Input type	Must be TEXT, RADIO, CHECKBOX, or PASSWORD
VALIDATE	Field validation	Optional field validation type (see Table A.51)
VALUE	Initial value	Optional initial field value

TABLE A.51 <CFINPUT> VALIDATION TYPES

Type	Description
CREDITCARD	Correctly formatted credit card number verified using mod10 algorithm
DATE	Date in mm/dd/yy format
EURODATE	European date in dd/mm/yy format
FLOAT	Number with decimal point
INTEGER	Number with no decimal point
SOCIAL_SECURITY_NUMBER	Social security number formatted as 999-99-9999 (using hyphens or spaces as separators)
TELEPHONE	Phone number in 999-999-9999 format (using hyphens or spaces as separators); area code and exchange must not begin with 0 or 1
TIME	Time in hh:mm or hh:mm:ss format
ZIPCODE	U.S. ZIP code, in either 99999 or 99999-9999 format

Example: The example in Listing A.11 creates a simple form with several fields. These fields employ several types of data validation using <CFINPUT>.

LISTING A.11 USING <CFINPUT> TO VALIDATE DATA ENTRY

```
<CFFORM ACTION="process.cfm">
<P>Enter your name: <CFINPUT TYPE="text" NAME="name" REQUIRED="Yes"
   MESSAGE="NAME is required!">
<P>Enter your phone number:
 <CFINPUT TYPE="text" NAME="phone" VALIDATE="telephone"
  MESSAGE="You entered an invalid phone number!">
<P>Select credit card:
```

LISTING A.11 CONTINUED

```
   <SELECT NAME="ccnumber">
      <OPTION>MasterCard
      <OPTION>Visa
      <OPTION>Amex
   </SELECT>
<P>CC#: <CFINPUT TYPE="text" NAME="ccnumber" VALIDATE="creditcard" MAXLENGTH="12"
    MESSAGE="Please enter a valid credit card number!">
<p><INPUT TYPE="Submit" VALUE="Save">
</CFFORM>
```

Note

<CFINPUT> does not support input fields of type HIDDEN.

For more information about <CFINPUT>, see Chapter 13, "Form Data Validation."

See also: <CFFORM>, <CFGIRD>, <CFSELECT>, <CFSLIDER>, <CFTEXTINPUT>, <CFTREE>

<CFINSERT>

Description: <CFINSERT> adds a single row to a database table. <CFINSERT> requires that the data source and table names be provided. All other attributes are optional. The attributes for this tag are presented in Table A.52.

The data source can be a preconfigured data source or defined dynamically by using the value "__DYNAMIC__" (two underscores followed by the word dynamic, followed by two underscores). In this case, you also must provide a CONNECTSTRING.

Syntax:

```
<CFINSERT CONNECTSTRING="text" DATASOURCE="ODBC Data Source"
 DBNAME="database name"
 DBTYPE="type" DBSERVER="dbms" FORMFIELDS="List of File to Insert"
 PASSWORD="Password" PROVIDER="provider" PROVIDERDSN="data source"
 TABLENAME="Table Name" TABLEOWNER="owner" TABLEQUALIFIER="qualifier"
 USERNAME="User Name">
```

TABLE A.52 <CFINSERT> ATTRIBUTES

Attribute	Description	Notes
CONNECTSTRING	Text database connection value	Values required to connect to a dynamic data source; required when DATASOURCE is "__DYNAMIC__".
DATASOURCE	Name of ODBC data source	Required; can be an existing data source or defined dynamically.

Attribute	Description	Notes
DBNAME	Sybase database name	Optional; only used if using native Sybase drivers.
DBSERVER	Database server	Optional; only used if using native database drivers.
DBTYPE	Database type	Optional; defaults to ODBC; other values are ORACLE73, ORACLE80, SYBASE11, OLEDB, DB2, and INFORMIX73.
FORMFIELDS	List of fields to insert	Optional attribute; specifies which fields are to be inserted if they are present. Any fields present that are not in the list will not be inserted.
PASSWORD	ODBC data source password	Optional; used to override the ODBC login password specified in the ColdFusion Administrator.
PROVIDER	OLE-DB COM provider	Optional; only used if using OLE-DB.
PROVIDERDSN	Data source OLE-DB COM provider	Optional; only used if using OLE-DB.
TABLENAME	Name of table to insert data into	Required; some ODBC data sources require fully qualified table names.
TABLEOWNER	Table owner name	Optional; used by databases that support table ownership.
TABLEQUALIFIER	Table qualifier	Optional; used by databases that support full qualifiers.
USERNAME	ODBC data source login name	Optional; used to override the ODBC login name specified in the ColdFusion Administrator.

Note

Your form field names must match the column names in the destination table for <CFINSERT> to work correctly.

Tip

If your form contains fields that are not part of the table into which you are inserting data, use the FORMFIELDS attribute to instruct ColdFusion to ignore those fields.

Tip

For more control over the insertion of rows into a database table, you can use the <CFQUERY> tag, specifying INSERT as the SQL statement.

Example: In the first example, a simple data entry form is used to collect data to be inserted into the Merchandise table:

```
<FORM ACTION="testinsert_act.cfm" METHOD="post">
<P>film id: <INPUT TYPE="Text" MAXLENGTH="5" NAME="filmid" value="18">
<P>merchandise name: <INPUT TYPE="Text" NAME="MerchName" MAXLENGTH="100">
<P>merchandise desc: <input type="Text" NAME="Merchdescription">
<P>merchandise price: <input TYPE="Text" NAME="merchprice">
<p><INPUT TYPE="Submit">
</FORM>
```

This <CFINSERT> tag inserts this form data:

```
<CFINSERT DATASOURCE="OWS" TABLENAME="Merchandise">
```

If, however, the form contains additional fields that don't correspond to the fields in the Merchandise table, you must use the FORMFIELDS attribute to identify which form fields are to be inserted:

```
<CFINSERT DATASOURCE="OWS" TABLENAME="Merchandise"
  FORMFIELDS="FilmID,MerchName,MerchDescription,MerchPrice">
```

For more information about the <CFINSERT> tag, see Chapter 14, "Using Forms to Add or Change Data."

See also: <CFQUERY>, <CFUPDATE>, <CFSTOREDPROC>

<CFLDAP>

Description: <CFLDAP> is used for all interaction with LDAP servers. It can be used to search an LDAP server, as well as to add, change, or delete data. Table A.53 lists the attributes for <CFLDAP>.

Syntax:

```
<CFLDAP ACTION="Action" ATTRIBUTES="Attributes List" DN="Name" FILTER="Filter"
 FILTERFILE="filename" MAXROWS="Number" MODIFYTYPE="Modification type"
 NAME="Query Name" PASSWORD="Password"
 PORT="Port Number" REBIND="Yes or No" REFERRAL="Hops" SCOPE="Scope"
 SECURE="Security type" SEPARATOR="Separator character"
 SERVER="Server Address" SORT="Sort Order" SORTCONTROL="Ascending or Descending"
 START="Start Position"
 STARTROW="Number" TIMEOUT="Timeout" USERNAME="Name">
```

TABLE A.53 <CFLDAP> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action	Required; specifies one of the actions in Table A.54.
ATTRIBUTES	Desired attributes	Required if ACTION is QUERY, ADD, MODIFY, or MODIFYDN; comma-delimited list of desired attributes; query specified in NAME attribute will contain these columns.

Attribute	Description	Notes
DN	Distinguished name	Required if ACTION is ADD, MODIFY, MODIFYDN, or DELETE.
FILTER	Search filter	Optional search filter used if ACTION is QUERY.
FILTERFILE	Name of filter file and stanza within the file	Optional attribute.
MAXROWS	Maximum rows to retrieve	Optional attribute.
MODIFYTYPE	Adds, deletes, or replaces attribute	Optional; ADD, DELETE, or REPLACE. Used to modify an attribute.
NAME	Query name	Name of query for returned data; required if ACTION is QUERY.
PASSWORD	User password	Optional user password; might be required for update operations.
PORT	Port number	Optional port number; defaults to 389 if not specified.
REBIND	Rebinds referral callback	Optional; YES or NO; reissues query using original credentials.
REFERRAL	Number of hops	Optional; specifies number of hops a referral is limited.
SCOPE	Search scope	Optional search scope if ACTION is QUERY; valid values are ONELEVEL, BASE, and SUBTREE; default is ONELEVEL.
SECURE	Type of security	Optional; CFSSL_BASIC or CFSSL_CLIENT_AUTH. Must include additional information specified in Table A.55.
SEPARATOR	Separator character	Optional; the character LDAP will use to separate values in multivalued attributes.
SERVER	Server name	Required DNS name or IP address of LDAP server.
SORT	Sort order	Optional attribute; used if ACTION is QUERY; specifies the sort order as a comma-delimited list; can use ASC for ascending and DESC for descending; default is ASC.
SORTCONTROL	How to sort query results	Optional; enter NOCASE; default is case sensitive ASC or DESC.
START	Start name	Required if ACTION is QUERY; distinguished name to start search at.
STARTROW	Start row	Optional start row; defaults to 1.

TABLE A.53 CONTINUED

Attribute	Description	Notes
TIMEOUT	Timeout value	Optional timeout value; defaults to 1 minute.
USERNAME	User login name	Optional user login name; might be required for update operations.

TABLE A.54 <CFLDAP> ACTIONS

Action	Description
ADD	Adds an entry to an LDAP server
DELETE	Deletes an entry from an LDAP server
MODIFY	Updates an entry on an LDAP server
MODIFYDN	Updates the distinguished name of an entry on an LDAP server
QUERY	Performs a query against an LDAP server (default action)

TABLE A.55 VARIABLES FOR USE WITH THE SECURE ATTRIBUTE

SECURE Value	Description
CFSSL_BASIC	You must provide the name of the certificate database file (in Netscape cert7.db format).
CFSSL_CLIENT_AUTH	You must provide the certificate database name, the certificate_name, the keyword database (in Netscape key3.db format) that holds the public/private key-pair, and the password to the keyword database.

Example: The following example retrieves a list of names from a public directory:

```
<CFSET Name="John Doe">
<CFLDAP
   SERVER="ldap.bigfoot.com"
   ACTION="QUERY"
   NAME="results"
   START="cn=#Name#,c=US"
   FILTER="(cn=#Name#)"
   ATTRIBUTES="cn,o,mail,p"
   SORT="cn ASC"
>
<CFTABLE QUERY="results" BORDER="yes" HTMLTABLE="Yes">
   <CFCOL HEADER="Name" TEXT="#cn#">
   <CFCOL HEADER="Org" TEXT="#o#">
   <CFCOL HEADER="Email" TEXT="<A HREF='mailto:#mail#'>#mail#</A>">
</CFTABLE>
```

For more information about <CFLDAP> and directory services, see the advanced version of this book.

<CFLOCATION>

Description: <CFLOCATION> is used to redirect a browser to a different URL. See Table A.56 for a description of this tag's attributes.

Syntax:

```
<CFLOCATION ADDTOKEN="Yes|No" URL="URL">
```

TABLE A.56 <CFLOCATION> ATTRIBUTES

Attribute	Description	Notes
ADDTOKEN	Adds session tokens	Optional attribute; default is YES.
URL	URL (or relative URL) to redirect to	This attribute is required.

Example: The following example redirects the user to a login page if she's not already logged in (as indicated by the presence of a session variable named LoggedIn):

```
<CFIF NOT IsDefined("Session.LoggedIn")>
   <CFLOCATION URL="login.cfm">
</CFIF>
```

Note

Because <CFLOCATION> needs to write to the HTTP header, you cannot use it if you've already flushed the header from the ColdFusion output buffer with <CFFLUSH>.

Tip

If your template creates cookies and then redirects the user to another template with <CFLOCATION>, the cookies will not be created. In this situation, use <META HTTP-EQUIV="refresh" ...>, JavaScript, or some other directive to redirect the user.

Note

Unlike <CFINCLUDE>, any text or CFML after the <CFLOCATION> tag is ignored by ColdFusion.

See also: <CFINCLUDE>

<CFLOCK>, </CFLOCK>

Description: <CFLOCK> is used to restrict blocks of code to single threaded access. Once inside a locked block of code, all other threads are queued until the thread with the exclusive lock relinquishes control. Table A.57 lists all the <CFLOCK> attributes.

Syntax:

```
<CFLOCK NAME="lock name" SCOPE="Application or Session or Server"
 TYPE="Readonly or Exclusive" TIMEOUT="timeout" THROWONTIMEOUT="Yes or No"
 > ... </CFLOCK>
```

TABLE A.57 <CFLOCK> ATTRIBUTES

Attribute	Description	Notes
NAME	Name for a lock	Optional; only one request with a given name can execute at a time.
SCOPE	Scope of lock	Optional and should not be used with NAME. This is set to APPLICATION, SERVER, or SESSION. This identifies the scope of the shared item being locked.
TIMEOUT	Timeout interval	This attribute is required.
TYPE	Lock type	Optional; READONLY or EXCLUSIVE.
THROWONTIMEOUT	Timeout handling	This optional attribute specifies how timeouts should be handled; an exception is thrown if YES; processing continues if NO; defaults to YES.

This tag is intended for use over small sections of code in which you are accessing a shared resource, such as certain types of variables (SESSION, APPLICATION, and SERVER variables), server filesystems, or other shared resources (for example, <CFFILE>). Set TYPE to READONLY when you're just checking and not updating a session variable, and use TYPE=EXCLUSIVE when you're writing to a variable.

Example: The following code locks a session variable to check for its existence. Then, if the SESSION variable isn't defined, an exclusive lock is issued for the purpose of writing to it and the user is redirected to the login part of the application:

```
<CFLOCK TYPE="ReadOnly" SCOPE="Session">
   <CFSET LoggedIn=IsDefined("Session.LoggedIn")>
</CFLOCK>
<CFIF NOT LoggedIn>
   <CFLOCK TYPE="Exclusive" SCOPE="Session">
      <CFSET Session.LoggedIn=False>
   </CFLOCK>
   <CFLOCATION URL="/Login/Login_Form.cfm">
</CFIF >
```

Caution

Avoid unnecessary use of <CFLOCK>. Forcing code to single-threaded use only can seriously impact system performance.

See also: <CFCATCH>, <CFTRY>

<CFLOG>

Description: <CFLOG> enables you to produce user-defined log files. They can be targeted to run only for specified applications or tasks. This ability to produce logs on an application basis is intended primarily for ISPs. The attributes for this tag are presented in Table A.58.

Syntax:

```
<CFLOG APPLICATION="Yes or No" TEXT="text" LOG="log type" FILE="filename"
 TYPE="message type" THREAD="thread ID yes or no" DATE="date yes or no"
 TIME="time yes or no" APPLICATION="yes or no">
```

TABLE A.58 <CFLOG> ATTRIBUTES

Attribute	Description	Notes
APPLICATION	YES or NO	Optional; indicates whether to log the application name (from <CFAPPLICATION>) if one was used. Defaults to YES.
DATE	YES or NO	Optional; toggles logging of current system date; defaults to YES.
FILE	Full filename	Optional; specifies the name of a custom log file.
LOG	Log type	Optional; USER, APPLICATION, or SCHEDULER. It defaults to USER if FILE is specified but LOG isn't; it defaults to APPLICATION if neither FILE nor LOG is specified. APPLICATION logs information only for the application named in the current <CFAPPLICATION> tag. SCHEDULER logs execution of tasks in the ColdFusion scheduler.
TEXT	Message for log	Required; text of entry to be written to log.
THREAD	YES or NO	Optional; toggles logging of current system time; defaults to YES.
TIME	YES or NO	Optional; toggles logging of thread ID; defaults to YES.
TYPE	Message type	Optional; see Table A.59 for a list of message types.

TABLE A.59 MESSAGE TYPES IN ORDER OF SEVERITY

Type	Description
INFORMATION	Simple informational message
WARNING	A problem of some sort might have occurred

TABLE A.59 CONTINUED

Type	Description
ERROR	An error has occurred
FATAL INFORMATION	Fatal error has occurred

Example: This example involves writing to a custom log for the current application. It is invoked when a user makes more than three unsuccessful login attempts:

```
<CFIF FORM.LoginAttempts GT 3>
  <CFLOG
    TEXT="Invalid login attempt: #CGI.REMOTE_ADDR#"
    FILE = "C:\INETPUB\WWWROOT\OWS\OWS_SECURITY.LOG"
    TYPE="Information"
  >
</CFIF>
```

<CFLOOP>, <CFBREAK>, </CFLOOP>

Description: <CFLOOP> enables you to create loops within your code. *Loops* are blocks of code that are executed repeatedly until a specific condition is met. <CFBREAK> enables you to unconditionally terminate a loop. ColdFusion supports five types of loops:

- For—These loops repeat a specific number of times.
- While—These loops repeat until a set condition returns FALSE.
- Query—These loops go through the results of a <CFQUERY> once for each row returned.
- List—These loops go through the elements of a specified list.
- Collection—These loops are used to loop over collections.

Attributes for this tag are presented in Table A.60.

Syntax:

For loop:

```
<CFLOOP INDEX="Index" FROM="Loop Start" TO="Loop End" STEP="Step Value">
```

While loop:

```
<CFLOOP CONDITION="Expression">
```

Query loop:

```
<CFLOOP QUERY="Query Name" STARTROW="Start Row Value" ENDROW="End Row Value">
```

List loop:

```
<CFLOOP INDEX="Index" LIST="List" DELIMITERS="Delimiters">
```

Collection loop:

```
<CFLOOP COLLECTION="Collection" ITEM="Item">
```

Note

The syntax and use of <CFLOOP> varies based on the type of loop being executed.

TABLE A.60 <CFLOOP> ATTRIBUTES

Attribute	Description	Notes
COLLECTION	Collection to loop through	This attribute is required for Collection or structure loops.
CONDITION	While loop condition	This attribute is required for While loops and must be a valid condition.
DELIMITERS	List loop delimiters	This is an optional List loop attribute; if it is omitted, the default delimiter of a comma is used.
ENDROW	Query loop end position	This is an optional Query loop attribute; if it is omitted, all rows are processed.
FROM	For loop start position	This attribute is required for For loops and must be a numeric value.
INDEX	Current element	This attribute is required for For loops and List loops and holds the name of the variable that will contain the current element.
ITEM	Current item	This attribute is required for Collection loops.
LIST	List loop list	This attribute is required for List loops and can be a ColdFusion list field or a static string.
QUERY	Query loop query	This attribute is required for Query loops and must be the name of a previously executed <CFQUERY>.
STARTROW	Query loop start position	This is an optional Query loop attribute; if it is omitted, the loop will start at the first row.
STEP	For loop step value	This is an optional For loop attribute; if it is omitted, the default value of 1 is used.
TO	For loop end position	This attribute is required for For loops and must be a numeric value.

Example: The following is a For loop used in a FORM to populate a select field with the years 1901–2000. The alternative would have been to enter 100 OPTION values manually:

```
<SELECT NAME="year">
 <CFLOOP INDEX="YearValue" FROM="1901" TO="2000">
 <OPTION><CFOUTPUT>#YearValue#</CFOUTPUT>
```

```
 </CFLOOP>
</SELECT>
```

The next example does the exact same thing but presents the list in reverse order. This is done by specifying a `STEP` value of `-1`:

```
<SELECT NAME="year">
 <CFLOOP INDEX="YearValue" FROM="2000" TO="1901" STEP="-1">
  <OPTION><CFOUTPUT>#YearValue#</CFOUTPUT>
 </CFLOOP>
</SELECT>
```

This example loops until any random number between 1 and 10, excluding 5, is generated:

```
<CFSET RandomNumber=0>
<CFLOOP CONDITION= "(RandomNumber GTE 0) AND (RandomNumber NEQ 5)">
 <CFSET RandomNumber=RandRange(1, 10)>
</CFLOOP>
```

This example creates a `Query` loop that processes an existing `<CFQUERY>` named `Orders`, but it processes only rows 100–150:

```
<CFLOOP QUERY="Orders" STARTROW="100" ENDROW="150">
 <CFOUTPUT>
  #OrderNum# - #DateFormat(OrderDate)# - #DollarFormat(Total)#<BR>
 </CFOUTPUT>
</CFLOOP>
```

This example loops through a user-supplied list of titles, displaying them one at a time:

```
<CFLOOP INDEX="Title" LIST="#FORM.Titles#">
 <CFOUTPUT>
  Title: #Title#<BR>
 </CFOUTPUT>
</CFLOOP>
```

This example uses `<CFBREAK>` to terminate a loop when a specific row is reached—in this case, an order number greater than 10,000:

```
<CFLOOP QUERY="Orders" >
 <CFIF OrderNum GT 10000>
  <CFBREAK>
 </CFIF>
 <CFOUTPUT>
  #OrderNum# - #DateFormat(OrderDate)# - #DollarFormat(Total)#<BR>
 </CFOUTPUT>
</CFLOOP>
```

This last example, shown in Listing A.12, involves nesting a `COLLECTION` loop inside an `INDEX` loop.

LISTING A.12 LOOPING OVER A COLLECTION

```
<CFSCRIPT>
   Colors = StructNew();
   Colors["Red"] = 1;
   Colors["Green"] = 2;
   Colors["Blue"] = 3;
```

```
   Colors["Yellow"] = 4;
   Colors["Orange"] = 5;
   Colors["Pink"] = 6;
   Colors["Brown"] = 7;
   Colors["Black"] = 8;
   Colors["Tan"] = 9;
   Colors["White"] = 10;
</CFSCRIPT>
<CFLOOP FROM="1" TO="4" INDEX="ii">
   <CFLOOP COLLECTION="#Colors#" ITEM="Color">
      <BR><CFOUTPUT><FONT COLOR="#Color#" SIZE="#ii#">#Color# = #Colors[Color]#
      </FONT></CFOUTPUT>
   </CFLOOP>
</CFLOOP>
```

Tip

Using <CFLOOP> to process queries is substantially slower than using <CFOUTPUT>. Whenever possible, use <CFOUTPUT> to loop through query results.

Note

The <CFLOOP> tag can be nested, and there is no limit placed on the number of nested loops allowed.

<CFMAIL>, </CFMAIL>

Description: <CFMAIL> generates SMTP mail from within ColdFusion templates. <CFMAIL> can be used to output query results, just like <CFOUTPUT>, or on its own. The <CFMAIL> tag itself is used to set up the mail message, and all text between the <CFMAIL> and </CFMAIL> tags is sent as the message body. <CFMAIL> requires that you specify a sender address, recipient address, and subject. All other attributes are optional. The attributes for this tag are presented in Table A.61.

Syntax:

```
<CFMAIL BCC="Blind CC Addresses" CC="Carbon Copy Addresses"
 FROM="Sender Address"
 GROUP="Group Name" GROUPCASESENSITIVE="Yes or No"
 MAXROWS="Maximum Mail Messages"
 MIMEATTACH="Pathname" PORT="SMTP TCP/IP Port" QUERY="Query Name"
 SERVER="SMTP Server Address" SUBJECT="Subject" TIMEOUT="SMTP Connection Timeout"
 TO="Recipient Address" TYPE="Message Type" MAILERID="id">
 Message
</CFMAIL>
```

TABLE A.61 <CFMAIL> ATTRIBUTES

Attribute	Description	Notes
BCC	Blind carbon copy addresses	Optional blind carbon copy addresses.

TABLE A.61 <CFMAIL> ATTRIBUTES

Attribute	Description	Notes
CC	Carbon copy	Optional one or more carbon addresses copy addresses separated by commas.
FROM	Sender's address	Required sender's e-mail address.
GROUP	Query column to group on	Optional attribute that specifies column to group on. See <CFOUTPUT> for more information on grouping data.
GROUPCASESENSITIVE	Enables grouping with respect to case	Optional. If the specified QUERY was generated from case-insensitive SQL, setting this to NO (it defaults to YES) will preserve order of query.
MAILERID	ID for X-Mailer SMTP header	Optional attribute enabling you to specify an X-Mailer ID for the SMTP header. Defaults to Allaire ColdFusion Application Server.
MAXROWS	Maximum message to send	Optional attribute specifying the maximum number of e-mail messages to generate.
MAILERID	Mailer ID	Optional mailer ID; default is Allaire ColdFusion Application Server.
MIMEATTACH	Fully qualified filename	Optional pathname to file that will be attached as a MIME-encoded attachment.
PORT	TCP/IP SMTP port	Optional TCP/IP SMTP port; overrides the default value of 25 if specified.
QUERY	<CFQUERY> to draw data from	E-mail can be generated based on the results of a <CFQUERY>; to do this specify the name of the <CFQUERY> here. This is an optional attribute.
SERVER	SMTP mail server	Optional SMTP mail server name; overrides the default setting if specified.
SUBJECT	Message subject	Required message subject.

Attribute	Description	Notes
TIMEOUT	Connection timeout interval	Optional SMTP connection timeout interval; overrides the default setting if specified.
TO	Recipient's address	Required recipient's e-mail address.
TYPE	Message type	Optional message type; currently the only supported type is HTML, indicating that HTML code is embedded in the message.

Example: The following is a simple e-mail message based on a form submission. It uses form fields in both the attributes and the message body itself:

```
<CFMAIL
 FROM="#FORM.EMail#"
 TO="sales@orangewhipstudios.com"
 SUBJECT="Customer inquiry"
>The following customer inquiry was posted to our Web site:
Name: #FORM.name#
email: #FORM.EMail#

Message:
#FORM.Message#

</CFMAIL>
```

This next example sends an e-mail message based on <CFQUERY> results. The message is sent once for each row retrieved:

```
<CFQUERY NAME="GetInquiries" DATASOURCE="OWS">
  SELECT * FROM WebQueries
</CFQUERY>
<CFMAIL
 FROM=" sales@orangewhipstudios.com "
 TO="sales@orangewhipstudios.com"
 SUBJECT="Customer inquiry"
 QUERY="GetInquiries"
>The following customer inquiry was posted to our Web site:
Name: #GetInquiries.name#
email: #GetInquiries.EMail#

Message:
#GetInquiries.Message#

</CFMAIL>
```

This next example, shown in Listing A.13, sends an e-mail message based on <CFQUERY> results. The message is sent once for each row retrieved. Note that it also specifies a mail server other than the default SMTP server defined in the ColdFusion Administrator.

LISTING A.13 SENDING E-MAIL WITH A SPECIFIED SERVER

```
<CFQUERY NAME="GetMailingList" DATASOURCE="OWS">
   SELECT FirstName, Email
   FROM Contacts
   WHERE MailingList = 1
</CFQUERY>

<CFMAIL
 QUERY="GetMailingList"
 FROM="Sales@orangewhipstudios.com"
 TO="#Email#"
 SUBJECT="Buy our stuff"
 SERVER="mail.orangewhip.com"
 MIMEATTACH="C:\OWS\Catalog\Catalog2001.pdf"
>Dear #FirstName#,

We sure would appreciate it if you'd visit our
Web site and buy some of our junk.

Please find our catalog attached.

Thanks.
--The Management
  Orange Whip Studios
</CFMAIL>
```

Tip

Unlike Web browsers, e-mail programs do not ignore whitespace. Carriage returns are displayed in the e-mail message if you embed carriage returns between the <CFMAIL> and </CFMAIL> tags.

It's also worth noting that if you specify HTML in the TYPE attribute, you can send e-mail consisting of HTML code. This has become popular in recent years because it gives users much more control over the formatting, but users must still consider whether the recipients' e-mail client software supports HTML.

Note

To use <CFMAIL>, the ColdFusion SMTP interface must be set up and working. If e-mail is not being sent correctly, use the ColdFusion Administrator to verify that ColdFusion can connect to your SMTP mail server.

Note

The PORT, SERVER, and TIMEOUT attributes will never be used in normal operation. These are primarily used for debugging and troubleshooting e-mail problems.

Note

E-mail errors are logged to the \CFUSION\MAIL\LOG directory. Messages that cannot be delivered are stored in the \CFUSION\MAIL\UNDELIVER directory.

For more information about the <CFMAIL> tag, see Chapter 28, "Interacting with E-mail."

See also: <CFMAILPARAM>, <CFPOP>

<CFMAILPARAM>

Description: <CFMAILPARAM> is used inside <CFMAIL> </CFMAIL> tags and is used to specify additional headers and file attachments. Note that you can use several <CFMAILPARAM> tags within each <CFMAIL>. Attributes for this tag are presented in Table A.62.

Syntax:

```
<CFMAILPARAM FILE="FileName" NAME="Header name" VALUE="Header value">
```

TABLE A.62 <CFMAILPARAM> ATTRIBUTES

Attribute	Description	Notes
FILE	Full pathname to file to be attached	Required if NAME is not specified
NAME	Name of SMTP mail header	Required if FILE is not specified
VALUE	Header value	Required if NAME is specified

Example: This example sends an e-mail to a specific address, specifying a particular reply-to header and attaching two files:

```
<CFMAIL
 FROM="MrBig@OrangeWhipStudios.com"
 TO="you@domain.org"
 SUBJECT="We need your eyes"
>Please consider putting your eyballs on our Web site:

www.orangewhipstudios.com

Thank you,
--The Management
<CFMAILPARAM FILE="c:\temp\tcmnote.txt">
<CFMAILPARAM FILE="c:\temp\more.HTM">
<CFMAILPARAM NAME="Reply-To" VALUE="John Doe <John@doe.com>">
</CFMAIL>
```


<CFMODULE>

Description: <CFMODULE> is used to call a custom tag explicitly stating its full or relative path. This is necessary when the custom tag template is not in the current directory. Table A.63 lists the <CFMODULE> attributes. Your own tag attributes also can be added to this list.

Syntax:

```
<CFMODULE NAME="Path" TEMPLATE="Path" ATTRIBUTE_n=VALUE_n...>
```

TABLE A.63 <CFMODULE> ATTRIBUTES

Attribute	Description	Notes
NAME	Fixed path to tag file	Either TEMPLATE or NAME must be used, but not both at once; use a period for directory delimiters.
TEMPLATE	Relative path to tag file	Either TEMPLATE or NAME must be used, but not both at once.
*ATTRIBUTE*_n	Your attribute/value pairs	Optional; you add your custom tag's attributes to your call.
ATTRIBUTECOLLECTION	Key/value pairs stored in a structure	Optional; alternative to specifying your attributes one at a time.

Note

When using the NAME attribute, you place the custom tag template either in the ColdFusion custom tags directory (which by default is in C:\CFUSION\CUSTOMTAGS) or in a directory beneath ColdFusion's custom tags directory. Use periods (.) as delimiters between subdirectories.

When using TEMPLATE, you must specify either a path relative to the current directory or a path that has been mapped in the ColdFusion Administrator.

Example: This example calls a custom tag named DUMPQUERY.CFM in the directory C:\CFUSION\CUSTOMTAGS\OUTPUT:

```
<CFQUERY DATASOURCE="OWS" NAME="GetContacts">
   SELECT
      FirstName,
      LastName,
      Address
   FROM Contacts
   WHERE State IS NOT NULL
</CFQUERY>
<CFMODULE NAME="OUTPUT.DUMPQUERY" QUERY="GetContacts" BORDER="1" MAXROWS="2">
```

See also: <CFASSOCIATE>

<CFOBJECT>

Description: <CFOBJECT> enables you to use COM, Java, and CORBA objects within your ColdFusion applications. You need to know an object's ID or filename to use it, as well as its methods and properties. <CFOBJECT> attributes vary depending on the type of object with which you are working. The attributes for each type are listed in the Tables A.64–66.

To use an object with <CFOBJECT>, that object must be already installed on the server.

Syntax:

```
<CFOBJECT ACTION="Action" CLASS="Class ID" CONTEXT="Context" NAME="Name"
 SERVER="Server Name" TYPE="COM or CORBA" LOCALE="Type/value pairs">
```

TABLE A.64 <CFMODULE> ATTRIBUTES FOR COM

Attribute	Description	Notes
ACTION	Action	Required attribute; must be either CREATE to instantiate an object or CONNECT to connect to a running object.
CLASS	Component ProgID	This attribute is required.
CONTEXT	Operation context	Optional attribute; must be INPROC, LOCAL, or REMOTE. User's Registry setting is not specified.
NAME	Object name	Required attribute.
SERVER	Valid server name	Server name as UNC, DNS, or IP address; required only if CONTEXT = "remote".
TYPE	Object type	Required; set to COM.

TABLE A.65 <CFMODULE> ATTRIBUTES FOR CORBA

Attribute	Description	Notes
CLASS	Component ProgID	Required; if CONTEXT is IOR, this names the file containing the IOR; must be readable by ColdFusion; if CONTEXT is NAMESERVICE, this specifies period-delimited class name.
CONTEXT	Operation context	Required; IOR or NAMESERVICE.
LOCALE	Type/value pair arguments	Optional, specific to visi-broker orbs.
NAME	Object name	This attribute is required.
TYPE	Object type	Required; set to CORBA.

TABLE A.66 <CFMODULE> ATTRIBUTES FOR JAVA

Attribute	Description	Notes
ACTION	Action	Required; set to CREATE for creating objects under WebLogic
CLASS	Component ProgID	Required; name of Java class
NAME	Object name	Required; name used in CFML to address object
TYPE	Object type	Required; set to JAVA

Example: The following example instantiates a COM object named NT.Exec and invokes a method:

```
<CFOBJECT TYPE="COM" CLASS="NT.Exec" ACTION="CREATE" NAME="Exec">
<CFSET Exec.Command="DIR C:\">
<CFSET temp=Exec.Run()>
```

Note

For more information about <CFOBJECT> and COM, see Chapter 12, "Interfacing with COM and DCOM Objects," of *Advanced ColdFusion 4 Development* (Que, 1999).

Note

Use of <CFOBJECT> can be disabled in the ColdFusion Administrator program.

<CFOUTPUT>, </CFOUTPUT>

Description: <CFOUTPUT> is used to output the results of a <CFQUERY> or any time text includes variables that are to be expanded. If <CFQUERY> is used to process the results of a <CFQUERY> tag, any code between <CFOUTPUT> and </CFOUTPUT> is repeated once for every row. When outputting query results, an additional set of variables is available. These are documented in Table A.68. <CFOUTPUT> can be used with the GROUP attribute to specify a data group from a query. Data that is grouped together is displayed so that only the first occurrence of each value is output. The attributes for this tag are presented in Table A.67.

Syntax:

```
<CFOUTPUT QUERY="Query Name" MAXROWS="Maximum Rows" STARTROW="Start Row"
 GROUP="Group Column" GROUPCASESENSITIVE="Yes or No">
Code
</CFOUTPUT>
```

TABLE A.67 <CFOUTPUT> ATTRIBUTES

Attribute	Description	Notes
GROUP	Column to group on	This optional attribute allows you to define output groups.
GROUPCASESENSITIVE	YES or NO	Optional; indicates whether the same values but in different case should be treated as separate entires.
MAXROWS	Maximum rows to display	This optional attribute specifies the maximum number of rows to display. If omitted, all rows are displayed.
QUERY	Query name	Optional; query name refers to the query results within <CFOUTPUT> text.
STARTROW	First row to display	Optional; specifies the output start row.

TABLE A.68 <CFOUTPUT> FIELDS AVAILABLE WHEN USING THE QUERY ATTRIBUTE

Field	Description
#COLUMNLIST#	Comma-delimited list of columns with a query
#CURRENTROW#	The number of the current row, starting at 1, and incremented each time a row is displayed
#RECORDCOUNT#	The total number of records to be output

Example: Any time you use variables or fields within your template, you must enclose them within <CFOUTPUT> tags, as shown in this example. Otherwise, the field name is sent as is and is not expanded:

```
<CFOUTPUT>
 Hi #CLIENT.Name#, thanks for dropping by again.<P>
 You have now visited us #NumberFormat(CLIENT.Visits)#
 since your first visit on #DateFormat(CLIENT.FirstVisit)#.
</CFOUTPUT>
```

This example uses <CFOUTPUT> to display the results of a query in an unordered list:

```
<CFQUERY NAME="GetContacts" DATASOURCE="ows">
  SELECT FirstName, LastName, Phone
  FROM Contacts
</CFQUERY>
<UL>
<CFOUTPUT QUERY="GetContacts">
 <LI>#LastName#, #FirstName# - Phone: #Phone#
</CFOUTPUT>
</UL>
```

You can use the GROUP attribute to group output results. This example groups contacts by whether they're on the mailing list:

```
<CFQUERY NAME="GetContacts" DATASOURCE="ows">
  SELECT FirstName, LastName, Phone, MailingList
  FROM Contacts
  ORDER BY MailingList
</CFQUERY>
<UL>
 <CFOUTPUT QUERY="GetContacts" GROUP="MailingList">
  <LI><B>#Iif(MailingList EQ 1, DE("On"), DE("Not On"))# Mailing List</B>
   <UL>
     <CFOUTPUT>
    <LI>#LastName#, #FirstName# - Phone: #Phone#
     </CFOUTPUT>
   </UL>
 </CFOUTPUT>
</UL>
```


Note

There is no limit to the number of nested groups you can use in a <CFOUTPUT>. However, every column used in a GROUP must be part of the SQL statement ORDER BY clause.

Tip

The STARTROW and MAXROWS attributes can be used to implement a "display next n of n" type display. You should note, however, that even though only a subset of the retrieved data is displayed, it has *all* been retrieved by the <CFQUERY> statement. So, although the page might be transmitted to the browser more quickly because it contains less text, the SQL operation itself takes no less time.

For more information about <CFOUTPUT>, see Chapter 9, "Using ColdFusion," and Chapter 11, "Creating Data-Driven Pages."

See also: <CFLOOP>, <CFMAIL>, <CFQUERY>, <CFTABLE>

<CFPARAM>

Description: <CFPARAM> lets you specify default values for parameters and specify parameters that are required. <CFPARAM> requires that you specify a variable name. If a VALUE is passed as well, that value will be used as the default value if the variable is not specified. If VALUE is not specified, <CFPARAM> requires that the named variable be passed; it will generate an error message if it is not. The attributes are documented in Table A.69.

It is commonly used to ensure that variables are defined with default values. Used in this way, it replaces this conditional logic:

```
<CFIF NOT IsDefefined("MyVar")>
  <CFSET MyVar="SomeValue">
</CFIF>
```

with:

```
<CFPARAM NAME="MyVar" DEFAULT="SomeValue">
```

Syntax:

```
<CFPARAM NAME="Parameter Name" DEFAULT="Default">
```

TABLE A.69 <CFPARAM> ATTRIBUTES

Attribute	Description	Notes
NAME	Name of variable	Required; name should be fully qualified with scope
DEFAULT	Default variable value	Optional; the value is used as the default value if the variable is not already defined
TYPE	Data type	Optional; type of data required; see the following list

Here are the valid values for TYPE:

- Any (default)
- Array

- Binary
- Boolean
- Date
- Numeric
- Query
- String
- Struct
- UUID
- variableName (ensures the value is valid as a variable name)

Example: The following specifies a default value for a field that is to be used in a <CFQUERY> tag, making it unnecessary to write conditional code to build a dynamic SQL statement:

```
<CFPARAM NAME="Form.Minimum" DEFAULT="10">
<CFQUERY NAME="OverDue" DATASOURCE= "ows">
 SELECT MerchID, MerchName
 FROM Merchandise
 WHERE MerchPrice <= #Form.Minimum#
</CFQUERY>
```

This example makes the Minimum field required and indicates that it must be numeric; an error is generated if you request the template and the Minimum field is not specified or is non-numeric:

```
<CFPARAM NAME="Minimum" DEFAULT="10" TYPE="numeric">
<CFQUERY NAME="OverDue" DATASOURCE= "ows">
 SELECT MerchID, MerchName
 FROM Merchandise
 WHERE MerchPrice <= #Form.Minimum#
</CFQUERY>
```

See also: <CFSET>

<CFPOP>

Description: <CFPOP> retrieves and manipulates mail in a POP3 mailbox. You must know three things to access a POP mailbox: the POP server name, the POP login name, and the account password. <CFPOP> has three modes of operation: It can be used to retrieve just mail headers and entire message bodies and to delete messages. POP messages are not automatically deleted when they are read and must be deleted explicitly with a DELETE operation. Table A.70 lists the <CFPOP> attributes; Table A.72 lists the columns returned when retrieving mail or mail headers.

Syntax:

```
<CFPOP ACTION="Action" ATTACHMENTSPATH="Path" MAXROWS="Number"
 MESSAGENUMBER="Messages" NAME="Query Name" PASSWORD="Password"
 PORT="Port Number"
 SERVER="Mail Server" STARTROW="Number" TIMEOUT="Timeout" USERNAME="User Name">
```

TABLE A.70 <CFPOP> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action	Optional; one of the values in Table A.71.
ATTACHMENTSPATH	Attachment path	Optional path to store mail attachments.
MAXROWS	Maximum messages to retrieve	Optional attribute; ignored if MESSAGENUMBER is used.
MESSAGENUMBER	Message number	Optional message number (or comma-delimited list of message numbers); required if ACTION is DELETE; specifies the messages to be deleted or retrieved.
NAME	Query name	Required if ACTION is GETALL or GETHEADERONLY; name of query to be returned. Query columns are listed in Table A.72.
PASSWORD	Password	Optional POP account password; most POP servers require this.
PORT	Mail server port	Optional attribute; defaults to port 110.
SERVER	Mail server	Required; DNS name or IP address of the POP mail server.
STARTROW	Start row	Optional start row; defaults to 1; ignored if MESSAGENUMBER is used.
TIMEOUT	Timeout value	Optional timeout value.
USERNAME	Login name	Optional POP login name; most POP servers require this.

TABLE A.71 <CFPOP> ACTIONS

Action	Description
DELETE	Deletes messages from a POP mailbox
GETALL	Gets message headers and body
GETHEADERONLY	Gets just message headers

TABLE A.72 <CFPOP> QUERY COLUMNS

Column	Description
ATTACHMENTFILES	List of saved attachments; only present if ACTION is GETALL and an ATTACHMENT path was specified

Column	Description
ATTACHMENTS	List of original attachment names; only present if ACTION is GETALL and an ATTACHMENT path was specified
BODY	Body of the message
CC	List of any carbon copy recipients
DATE	Message date
FROM	Sender name
HEADER	Mail header
MESSAGENUMBER	Message number for use in calls with future calls
REPLYTO	E-mail address to reply to
SUBJECT	Message subject
TO	Recipient list

Example: This example retrieves a list of waiting mail in a POP mailbox and then displays the message list in an HTML list:

```
<CFPOP SERVER="mail.a2zbooks.com" USERNAME=#username# PASSWORD=#pwd#
 ACTION="GETHEADERONLY" NAME="msg">
<UL>
<CFOUTPUT QUERY="msg">
<LI>From: #from# - Subject: #subject#
</CFOUTPUT>
</UL>
```

Note

<CFPOP> is used to retrieve mail only. Use the <CFMAIL> tag to send mail.

For more information about the <CFPOP> tag, see Chapter 24, "Improving Performance."

See also: <CFMAIL>

<CFPROCESSINGDIRECTIVE>, </CFPROCESSINGDIRECTIVE>

Description: <CFPROCESSINGDIRECTIVE> enables you to suppress all whitespace between the start and end tags. If this tag is nested, the settings on the innermost tag are used.

<CFPROCESSINGDIRECTIVE> attributes are listed in Table A.73.

Syntax:

```
<CFPROCESSINGDIRECTIVE SUPPRESSWHITESPACE="Yes or No">
... CFML ...
</CFPROCESSINGDIRECTIVE>
```

TABLE A.73 <CFPROCESSINGDIRECTIVE> ATTRIBUTES

Attribute	Description	Notes
SUPPRESSWHITESPACE	YES or NO	Required; indicates whether white-space is to be eliminated

Example: The example in Listing A.14 suppresses whitespace. To see the effects, try changing the value of the SUPPRESSWHITESPACE attribute to NO and view the source of the rendered page.

LISTING A.14 SUPPRESSING WHITESPACE

```
<CFPROCESSINGDIRECTIVE SUPPRESSWHITESPACE="Yes">
<CFQUERY NAME="GetContacts" DATASOURCE="OWS">
    SELECT LastName, Phone
    FROM CONTACTS
</CFQUERY>

<CFSET MyVar="This is a variable containing whitespace">
<P><CFOUTPUT>#MyVar#</CFOUTPUT>
<TABLE>
<CFOUTPUT QUERY="GetContacts">
<TR>
  <TD>#LastName#</TD>
  <TD>
  <CFIF Phone IS NOT "">
    #Phone#
  <CFELSE>
    n/a
  </CFIF>
  </TD>
</TR>
</CFOUTPUT>
</TABLE>
</CFPROCESSINGDIRECTIVE>
```

<CFQUERY>, </CFQUERY>

Description: <CFQUERY> submits SQL statements to a data source that is either previously configured or dynamically generated, or to another query. SQL statements can include SELECT, INSERT, UPDATE, and DELETE, as well as calls to stored procedures. <CFQUERY> returns results in a named set if you specify a query name in the NAME attribute. The <CFQUERY> attributes set up the query, and any text between the <CFQUERY> and </CFQUERY> tags becomes the SQL statement that is sent to the ODBC driver. ColdFusion conditional code can be used between the <CFQUERY> and </CFQUERY> tags, allowing you to create dynamic SQL statements. The attributes for this tag are presented in Table A.74.

Syntax:

```
<CFQUERY BLOCKFACTOR="Number of rows" CONNECTSTRING="connect string"
 NAME="Parameter Name"  DATASOURCE="Data Source"
```

```
 DBNAME="database name" DBPOOL="pool" DBTYPE="type" DBSERVER="dbms"
 USERNAME="User Name" PASSWORD="Password" PROVIDER="provider"
 PROVIDERDSN="data source" BLOCKFACTOR="factor" TIMEOUT="timeout value"
 CACHEDAFTER="date" CACHEDWITHIN="time span" DEBUG="Yes or No">
SQL statement
</CFQUERY>
```

TABLE A.74 <CFQUERY> ATTRIBUTES

Attribute	Description	Notes
BLOCKFACTOR	Number of rows to retrieve at once	Optional; available if using ODBC or Oracle drivers. Valid values are 1–100; the default value is 1.
CACHEDAFTER	Cache date	Optional; specifies that query is to be cached and cached copy is to be used after specified date.
CACHEDWITHIN	Cache time span	Optional; specifies that query is to be cached and cached copy is to be used within a relative time span.
CONNECTSTRING	Database connection string	Optional but required when DATASOURCE="__DYNAMIC__".This provides the connection information required to make the database connection dynamically.
DATASOURCE	Data source	Required unless DBTYPE=QUERY; identifies the ODBC or other data source from which the query should retrieve its data. You can also specify "__DYNAMIC__", in which case you must specify a CONNECTSTRING as well.
DBNAME	Sybase database name	Optional; only used with native Sybase and SQLOLEDB drivers.
DBPOOL	Database connection pool name	Optional; database pool name.
DBSERVER	Database server	Optional database server; only used if using native database drivers.
DBTYPE	Database type	Optional type; defaults to ODBC; other values are QUERY, ORACLE73, ORACLE80, SYBASE11, OLEDB, DB2, and INFORMIX73. Set to QUERY when querying a prior query.
DEBUG	Enable query debugging	Optional attribute; turns on query debugging output.
MAXROWS	Number of rows	Optional; specifies maximum number of rows to retrieve.

TABLE A.74 CONTINUED

Attribute	Description	Notes
NAME	Query name	Required; used to refer to the query results in <CFOUTPUT>, <CFMAIL>, or <CFTABLE> tags.
PASSWORD	ODBC data source	Optional; overrides the password specified in the ColdFusion Administrator.
PROVIDER	OLE-DB COM provider	Optional; used only if using OLE-DB.
PROVIDERDSN	Data source OLE-DB COM provider	Optional; only used if using OLE-DB.
TIMEOUT	Timeout value	Optional; timeout value in milli-seconds.
USERNAME	ODBC data source login name	Optional; overrides the login name specified in the ColdFusion Administrator.

The following example is a simple data retrieval query:

```
<CFQUERY DATASOURCE="OWS"  NAME="Contacts">
  SELECT FirstName, LastName, Phone
  FROM Contacts
</CFQUERY>
```

The next example, shown in Listing A.15, demonstrates the technique of querying against an existing query result. The second query uses DBTYPE of QUERY. The table it queries uses the name of the previous query resultset.

LISTING A.15 QUERYING AGAINST A QUERY RESULT

```
<CFQUERY DATASOURCE="OWS" NAME="GetAllContacts">
  SELECT * FROM Contacts
</CFQUERY>

<CFQUERY NAME="GetMailingList" DBTYPE="Query">
  SELECT LastName, FirstName, MailingList
  FROM GetAllContacts
  WHERE MailingList = 1
</CFQUERY>

<H2><CFOUTPUT>#GetAllContacts.RecordCount#</CFOUTPUT> Contact Records</H2>
<CFTABLE QUERY="GetAllContacts" BORDER="Yes" HTMLTABLE="Yes">
<CFCOL HEADER="Last Name" TEXT="#LastName#">
<CFCOL HEADER="First Name" TEXT="#FirstName#">
<CFCOL HEADER="On List" TEXT="#Iif( MailingList EQ 1,DE('Y'), DE(''))#">
</CFTABLE>

<H2><CFOUTPUT>#GetMailingList.RecordCount#</CFOUTPUT> Contact Records
    on Mailing List</H2>
<CFTABLE QUERY="GetMailingList" BORDER="Yes" HTMLTABLE="Yes">
```

```
<CFCOL HEADER="Last Name" TEXT="#LastName#">
<CFCOL HEADER="First Name" TEXT="#FirstName#">
<CFCOL HEADER="On List" TEXT="#Iif( MailingList EQ 1,DE('Y'), DE(''))#">
</CFTABLE>
```

The next example, shown in Listing A.16, demonstrates how dynamic SQL statements can be constructed using the ColdFusion conditional tags.

Listing A.16 Producing SQL Dynamically

```
<CFQUERY NAME="GetContacts" DATASOURCE="OWS">
  SELECT FirstName, LastName, Phone, ContactID
  FROM Contacts
  WHERE 1=1

  <CFIF FORM.FirstName IS NOT "">
    AND FirstName LIKE '#FORM.FirstName#%'
  </CFIF>

  <CFIF FORM.LastName IS NOT "">
    AND LastName LIKE '#FORM.LastName#%'
  </CFIF>

  <CFIF FORM.Phone IS NOT "">
    AND PhoneExtension LIKE 'FORM.#Phone#%'
  </CFIF>

  ORDER BY LastName, FirstName
</CFQUERY>
```

<CFQUERY> also can be used for all basic SQL operations. The next example demonstrates a delete query on a dynamically defined data source:

```
<CFQUERY NAME="DeleteContact" DATASOURCE="__DYNAMIC__"
  CONNECTSTRING = "Driver={Microsoft Access Driver (*.mdb)};
  Dbq=ows.mdb;
  DefaultDir=C:\cfusion\database;
  Uid=Admin;Pwd=;"
>
 DELETE FROM Contacts WHERE ContactID=#FORM.ContactID#"
</CFQUERY>
```

The last example demonstrates the use of <CFQUERY> to execute a stored procedure in a SQL Server database:

```
<CFQUERY NAME="GetContact" DATASOURCE="ContactSystem">
 {Call GetContacts(#URL.ContactID#)}
</CFQUERY>
```

Note that the preferred mechanism for executing stored procedures involves using <CFSTOREDPROC> and related tags. This older technique will work but is limited to returning one recordset.

For more information about the <CFQUERY> tag, see Chapter 11. For more information about creating dynamic SQL statements, see Chapter 12, "ColdFusion Forms." For more information about using <CFQUERY> for INSERT, UPDATE, and DELETE operations, see Chapter 14.

See also: <CFOUTPUT>, <CFMAIL>, <CFTABLE>, <CFSTOREDPROC>

<CFQUERYPARAM>

Description: <CFQUERYPARAM> is embedded within the SQL of a <CFQUERY>. It enables you to define query parameters and their data types. Queries will execute more quickly when passed data-typed parameters.

<CFQUERYPARAM> attributes are listed in Table A.75.

Syntax:

```
<CFQUERYPARAM CFSQLTYPE="Param type"
 MAXLENGTH="Number" NULL="Yes or No"
 SCALE="Number of decimal places"
 SEPARATOR="Delimiter character"
 VALUE="Param value" >
```

TABLE A.75 <CFQUERYPARAM> ATTRIBUTES

Attribute	Description	Notes
CFSQLTYPE	Type	Optional; data type specified from list in Table A.72
MAXLENGTH	Bytes	Optional; maximum length of parameter value
NULL	YES or NO	Optional; YES if passed parameter is null
SCALE	Number of decimals	Optional; applicable for TYPE=CF_SQL_NUMERIC and CF_SQL_DECIMAL. Specifies number of decimals; defaults to 0
SEPARATOR	Character	Optional; character to be used as a delimiter in lists
VALUE	Param value	Required; the parameter's value

TABLE A.76 CF_SQL TYPES

Attribute
CF_SQL_BIGINT
CF_SQL_BIT
CF_SQL_CHAR
CF_SQL_DATE
CF_SQL_DECIMAL

Attribute
CF_SQL_DOUBLE
CF_SQL_FLOAT
CF_SQL_IDSTAMP
CF_SQL_INTEGER
CF_SQL_LONGVARCHAR
CF_SQL_MONEY
CF_SQL_MONEY4
CF_SQL_NUMERIC
CF_SQL_REAL
CF_SQL_REFCURSOR
CF_SQL_SMALLINT
CF_SQL_TIME
CF_SQL_TIMESTAMP
CF_SQL_TINYINT
CF_SQL_VARCHAR

Example: In the following example, <CFQUERYPARAM> is used to validate the data type being passed:

```
<CFPARAM NAME="URL.ContactID" DEFAULT="q">
<CFTRY>
<CFQUERY NAME="GetContacts" DATASOURCE="ows">
  SELECT LastName, FirstName
  FROM Contacts
  WHERE ContactID = <CFQUERYPARAM VALUE="#URL.ContactID#"
  CFSQLTYPE="CF_SQL_INTEGER">
</CFQUERY>
<CFCATCH TYPE="Any">
   <P>An error has occured:
   <CFOUTPUT>
   <P>#CFCATCH.Detail#
   <P>#CFCATCH.Type#</CFOUTPUT>
</CFCATCH>
</CFTRY>
```

See also: <CFQUERY>, <CFSTOREDPROC>, <CFPROCPARAM>

<CFREGISTRY>

Description: <CFREGISTRY> can be used to directly manipulate the system Registry. The <CFREGISTRY> ACTION attribute specifies the action to be performed, and depending on the action, other attributes might or might not be necessary.

<CFREGISTRY> attributes are listed in Table A.77. Values for the ACTION attribute are listed in Table A.78.

Syntax:

```
<CFREGISTRY ACTION="action" BRANCH="branch" ENTRY="entry" NAME="query"
 SORT="sort order" TYPE="type" VALUE="value" VARIABLE="variable">
```

TABLE A.77 <CFREGISTRY> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action	Required; one of the values in Table A.78.
BRANCH	Registry branch	Required.
ENTRY	Branch entry	Required for GET, SET, and DELETE actions.
NAME	Name of record to contain keys and values	Required if ACTION is GETALL.
SORT	Sort order	Optional; can be used if ACTION is GETALL. Enables sorting on specified column(s): ENTRY, TYPE, and VALUE. You also can specify ASC for ascending sorts or DESC for descending sorts.
TYPE	Value type	Optional; can be used for all actions except DELETE; valid types are STRING, DWORD, and KEY; default is STRING.
VALUE	Value to set	Required if ACTION is SET.
VARIABLE	Variable to save value in	Required if ACTION is GET.

TABLE A.78 <CFREGISTRY> ACTIONS

Action	Description
DELETE	Deletes a Registry key
GET	Gets a Registry value
GETALL	Gets all Registry keys in a branch
SET	Sets a Registry value

Example: This example in Listing A.17 retrieves all the keys beneath the Allaire branch.

LISTING A.17 READING A KEY VALUE FROM THE REGISTRY

```
<CFREGISTRY ACTION="GETALL" NAME="reg"
 BRANCH="HKEY_LOCAL_MACHINE\SOFTWARE\Liquid Audio Settings">

<P>Number of Liquid Audio entries: <CFOUTPUT>#reg.RecordCount#</CFOUTPUT>
<TABLE CELLPADDING="3">
```

```
<CFOUTPUT QUERY="reg">
<TR ALIGN="left">
   <TH>Entry</TH>
   <TH>Type</TH>
   <TH>Value</TH>
</TR>
<TR ALIGN="left">
   <TD>#Entry#</TD>
   <TD>#Type#</TD>
   <TD>#Value#</TD>
</TR>
</CFOUTPUT>
</TABLE>
```

Note

Note that only Windows platforms use a concept of a Registry. This tag works only on Windows platforms.

Caution

You should take great care when using this tag, particularly when writing to the Registry. It is not hard to corrupt the integrity of the Registry. You might want to consider turning off the use of this tag in the ColdFusion Administrator, in the Tag Restrictions section under Security.

<CFREPORT>, </CFREPORT>

Description: <CFREPORT> is the ColdFusion interface to reports created with the Crystal Reports Professional report writer. <CFREPORT> requires only a single attribute: the name of the report to be processed. The full list of supported attributes is in Table A.79.

Syntax:

```
<CFREPORT REPORT="Report File" ORDERBY="Sort Order"
 USERNAME="User Name" PASSWORD="Password" FORMULA="Formula">
Optional filter conditions
</CFREPORT>
```

TABLE A.79 <CFREPORT> ATTRIBUTES

Attribute	Description	Notes
FORMULA	Crystal Reports formula	Optional; enables you to specify values for Crystal Reports formulas used in the report.
ORDERBY	Report sort order	Optional; overrides the default sort order specified when the report was created.
PASSWORD	ODBC data source password	Optional; used to override the ODBC login password specified in the ColdFusion Administrator.

TABLE A.79 <CFREPORT> ATTRIBUTES

Attribute	Description	Notes
REPORT	Name of RPT file to process	Required.
USERNAME	ODBC data source login name	The optional attribute is used to override the ODBC login name specified in the ColdFusion Administrator.

Example: The following example processes a report created with Crystal Reports Professional and passes it an optional filter condition:

```
<CFREPORT REPORT="\ows\scripts\Contact.rpt">
 {Contacts.ContacID} = "3"
</CFREPORT>
```

This example processes a report and specifies parameters to override the ODBC data source, user login name and password, and a formula named `title` derived from an HTML form:

```
<CFREPORT REPORT="\ows\scripts\ContactList.rpt" DATASOURCE="OWSInternal"
 USERNAME="Sales" PASSWORD="bigbucks" @Title="#FORM.title#">
 {Contacts.ContactID} = "Sales"
</CFREPORT>
```

<CFRETHROW>

Description: <CFRETHROW> enables you to force the current error to be invoked again within <CFCATCH> ... </CFCATCH> block. It generally is used when you have error trapping logic that traps an error your code isn't capable of handling. In this case, you want to *rethrow* the error.

Syntax:

```
<CFRETHROW>
```

Example: In the following example, a query attempts to insert a new record using key values entered through a form. This can result in a key violation. The <CFCATCH> block is used to trap this type of error, but if some other type of database error occurs, you rethrow the error:

```
<CFTRY>
   <CFQUERY NAME="InsertContactOrder" DATASOURCE="OWS">
      INSERT INTO ContactOrders (ContactID,OrderID)
      VALUES (#FORM.ContactID#,#FORM.OrderID#)
   </CFQUERY>

   <CFCATCH TYPE="DATABASE">
      <!---
      If the database signalled a 23000 error, we can ignore it,
      otherwise rethrow the exception.
      --->
      <CFIF CFCATCH.sqlstate neq 23000>
         <CFRETHROW>
```

```
      </CFIF>
   </CFCATCH>
</CFTRY>
```

See also: <CFCATCH>, <CFTHROW>, <CFTRY>

<CFSAVECONTENT>, </CFSAVECONTENT>

Description: <CFSAVECONTENT> enables you to save the output of a page or portion of a page in a variable. It is valuable when you need to process the output in some way before it is complete. It saves the results of evaluated expressions and custom tag output in the body. The attributes are described in Table A.80.

Syntax:

```
<CFSAVECONTENT VARIABLE="variablename">
```

TABLE A.80 <CFSAVECONTENT> ATTRIBUTES

Attribute	Description	Notes
VARIABLE	CFML variable name	Required; name of the variable in which to save content

Example: In this example, <CFSAVECONTENT> is used to save the page output in a variable. The phrase "Profit and Loss Q1" is then replaced with the phrase "Profit and Loss Q2":

```
<CFSAVECONTENT VARIABLE="ReportTitle">
  <CF_ProducePandL StartDate="4/1/2001" EndDate="6/30/2001">
</CFSAVECONTENT>
<CFOUTPUT>
#Replace(ReportTitle, "Profit and Loss Q1", "Profit And Loss Q2", "all")#
</CFOUTPUT>
```


<CFSCHEDULE>

Description: <CFSCHEDULE> enables you to programmatically create, update, delete, and execute tasks in the ColdFusion Administrator's scheduler. The scheduler enables you to run a specified page at scheduled times and intervals.

You have the option to direct page output to static HTML pages. This enables you to offer users access to pages that publish data (such as reports) without forcing them to wait while a database transaction that populates the data on the page is performed.

ColdFusion-scheduled events must be registered using the ColdFusion Administrator before they can be executed. Information supplied by the user includes the scheduled ColdFusion page to execute, the time and frequency for executing the page, and whether the output from the task should be published. A path and file are specified if the output is to be published.

<CFSCHEDULE> attributes are listed in Table A.81. The values for the ACTION attribute are described in Table A.82.

Syntax:

```
<CFSCHEDULE ACTION="Action" ENDDATE="Date" ENDTIME="Time" FILE="File Name"
 INTERVAL="Interval" LIMITIME="Seconds" OPERATION="HTTPRequest"
 PASSWORD="Password" PATH="Path" PORT="Port Number" PROXYSERVER="Server Name"
 PROXYPORT="port number"PUBLISH="Yes or No" "Yes or No"
 RESOLVEURL="Yes or No" REQUESTIMEOUT="seconds"
 STARTDATE="Date"  STARTTIME="Time" TASK="Task Name" URL="URL"
 USERNAME="User Name">
```

TABLE A.81 <CFSCHEDULE> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action (refer to Table A.82)	Required attribute.
ENDDATE	Event end date	Optional attribute; date the scheduled task should end.
ENDTIME	Event end time	Optional attribute; time the scheduled task should end; enter value in seconds.
FILE	File to create	Required if PUBLISH is Yes.
INTERVAL	Execution interval	Required if ACTION is UPDATE; can be specified as number of seconds; as daily, weekly, or monthly; or as execute.
LIMITTIME	Maximum execution time	Optional attribute; maximum number of seconds allowed for execution.
OPERATION	Operation	Required if ACTION is UPDATE. Currently only HTTPRequest is supported.
PASSWORD	Password	Optional password for protected URLs.
PATH	Path to save published files	Required if PUBLISH is YES.
PORT	Port number on server	Optional; used with RESOLVEURL set to YES to properly execute URLs that specify a port other than 80 (default).
PROXYSERVER	Proxy server name	Optional name of proxy server.
PROXYPORT	Port number on proxy server	Optional; used with RESOLVEURL set to YES to properly execute URLs that specify a port other than 80 (default).
PUBLISH	Publish static files	Optional attribute; YES if the scheduled task should publish files; default is NO.

Attribute	Description	Notes
REQUESTTIMEOUT	Seconds before timeout	Optional; used to extend the default timeout for long tasks.
RESOLVEURL	Resolve URLs	Optional attribute; resolve URLs to fully qualified URLs if YES; default is NO.
STARTDATE	Event start date	Optional attribute; date the scheduled task should start.
STARTTIME	Event start time	Optional attribute; time the scheduled task should start; enter value in seconds.
TASK	Task name	Required attribute; the registered task name.
URL	URL	Required if ACTION is UPDATE; the URL to be executed.
USERNAME	Username	Optional username for protected URLs.

TABLE A.82 <CFSCHEDULE> ACTIONS

Action	Description
DELETE	Deletes a task
UPDATE	Updates a task or creates it if it doesn't exist
RUN	Executes a task

Example: This example creates a recurring task that runs every 10 minutes (600 seconds). The output is saved in C:\INETPUB\WWWROOT\SAMPLE.HTML:

```
<CFSCHEDULE ACTION="UPDATE"
   TASK="TaskName"
   OPERATION="HTTPRequest"
   URL="http://127.0.0.1/testarea/test.cfm"
   STARTDATE="3/13/2001"
   STARTTIME="12:25 PM"
   INTERVAL="600"
   RESOLVEURL="Yes"
   PUBLISH="Yes"
   FILE="sample.html"
   PATH="c:\inetpub\wwwroot\"
   REQUESTTIMEOUT="600"
>
```

This example deletes the task created in the previous example:

```
<CFSCHEDULE ACTION="delete" TASK="TaskName">
```

For more information about <CFSCHEDULE> and scheduled events, see Chapter 37, "Event Scheduling."

Note

Execution of <CFSCHEDULE> can be disabled in the ColdFusion Administrator.

<CFSCRIPT>, </CFSCRIPT>

Description: <CFSCRIPT> and </CFSCRIPT> are used to mark blocks of ColdFusion script. ColdFusion script looks similar to JavaScript and enables you to produce certain functions of ColdFusion tags (and use many ColdFusion functions) and avoid all the wordy syntax associated with tags. Note that one major limitation of <CFSCRIPT> is that you can't execute SQL (or other) queries with it. Each line must be terminated with a semicolon.

Syntax:

```
<CFSCRIPT> script </CFSCRIPT>
```

Example: The example shown in Listing A.18 creates a structure and then uses it in place of a <CFSWITCH> to select a value.

LISTING A.18 USE OF <CFSCRIPT>

```
<CFSCRIPT>
Meals = StructNew();
Meals["Breakfast"]="Bacon,Eggs,Toast,Fruit,Coffee";
Meals["Lunch"]="Soup,Sandwich";
Meals["Dinner"]="Pasta,Garlic bread,Red wine";
WriteOutput("We'll be having " & Meals[Form.Meal] & " for " & Form.Meal);
</CFSCRIPT>
```

<CFSEARCH>

Description: <CFSEARCH> performs searches against Verity collections (in much the same way <CFQUERY> performs searches against ODBC data sources). To use <CFSEARCH>, you must specify the collection to be searched and the name of the query to be returned. You can search more than one collection at once, and you also can perform searches against Verity collections created with applications other than ColdFusion. Table A.83 lists the <CFSEARCH> attributes.

Syntax:

```
<CFSEARCH COLLECTION="Collection Name" CRITERIA="Search Criteria" CUSTOM1="Data"
 CUSTOM2="Data" EXTERNAL="Yes or No" "Yes or No"LANGUAGE="language"
 MAXROWS="Number" NAME="Name" STARTROW="Number" TYPE="Type">
```

TABLE A.82 <CFSEARCH> ATTRIBUTES

Attribute	Description	Notes
COLLECTION	Collection name	Required attribute; the name of the collection or collections to be searched. Multiple collections must be separated by commas; for external collections, specify the full path to the collection.
CRITERIA	Search criteria	Optional attribute; search criteria as shown in Appendix D.
EXTERNAL	External collection	Optional attribute; must be YES if using an external collection; default is NO.
LANGUAGE	Language	Optional; requires installation of International Search Page.
MAXROWS	Maximum rows to retrieve	Optional attribute; defaults to all.
NAME	Value column	Optional attribute; column to be used for OPTION VALUE attribute.
STARTROW	Start row	Optional attribute; default is first row.
TYPE	Search type	Optional attribute; can be SIMPLE or EXPLICIT.

Example: This example in Listing A.19 performs a search with a user-supplied search criteria. It searches the collection built in the example for <CFCOLLECTION>.

LISTING A.19 TEXT SEARCH WITH USER-SUPPLIED CRITERIA

```
<CFIF IsDefined("FORM.Searchfield")>
   <CFSEARCH COLLECTION="FilmsMerchandise" TYPE="SIMPLE"
   CRITERIA="#Form.SearchField#" NAME="GetMerch">
</CFIF>
<FORM ACTION="#CGI.SCRIPT_NAME#" METHOD="post">
<P>Find film merchandise: <INPUT NAME="searchfield" TYPE="Text">
<p><INPUT TYPE="Submit" VALUE="Search">
</FORM>

<CFIF IsDefined("Form.searchfield")>
   <CFIF GetMerch.RecordCount GT 0>
      <UL>
      <CFOUTPUT QUERY="GetMerch">
        <LI>#Key# - #NumberFormat(Score, "__.__" )# - #Summary#
      </CFOUTPUT>
      </UL>
   <CFELSE>
      <P>Nothing matching your criteria
   </CFIF>
</CFIF>
```

For more information about using <CFINDEX> and Verity collections, see Chapter 36. For the complete list of search expressions and instructions, see Appendix D.

See also: <CFCOLLECTION>, <CFINDEX>

<CFSELECT>, </CFSELECT>

Description: <CFSELECT> is used to simplify the process of creating data-driven SELECT controls. <CFSELECT> is not a Java control. <CFSELECT> requires that you pass it the name of a query for use in populating the drop-down list box. <CFSELECT> attributes are listed in Table A.83.

You can add your own options to the SELECT list by adding <OPTION> tags between the <CFSELECT> and </CFSELECT> tags.

Syntax:

```
<CFSELECT DISPLAY="Column Name" MESSAGE="Message Text"
 MULTIPLE="Yes or No""Yes or No"
 NAME="Field Name" ONERROR="JavaScript Error Function" PASSTHROUGH="attributes"
 QUERY="Query Name" REQUIRED="Yes or No" SELECTED="Value" SIZE="Size"
 VALUE="Column Name"></CFSELECT>
```

TABLE A.83 <CFSELECT> ATTRIBUTES

Attribute	Description	Notes
DISPLAY	Column to display	Optional query column to use as the displayed text.
MESSAGE	Validation failure message	Optional message to display upon validation failure.
MULTIPLE	Allow multiple selection	Optional attribute; defaults to NO.
NAME	Unique field name	This attribute is required.
ONERROR	JavaScript error function	Optional override to your own JavaScript error message function.
PASSTHROUGH	HTML <SELECT> attribute	Optional; <SELECT> attributes not supported by <CFSELECT>, such as TABINDEX.
QUERY	Query name	Required attribute; query to be used to populate the SELECT box.
REQUIRED	Field is required	Optional required flag; must be YES or NO if specified; defaults to NO.
SELECTED	Selected value	Value of the OPTION to be preselected.
SIZE	List size	Required attribute; number of options to display without scrolling.
VALUE	Value column	Optional attribute; column to be used for OPTION VALUE attribute.

Example: This example creates a simple data-driven SELECT control such that the user is required to make a selection:

```
<CFQUERY NAME="GetActors" DATASOURCE="OWS">
  SELECT ActorID, NameLast
  FROM Actors
</CFQUERY>

<CFFORM ACTION="process.cfm">
<CFSELECT
  NAME="Actors"
  QUERY="GetActors"
  VALUE="ActorID"
  DISPLAY="NameLast" SIZE="1"
  REQUIRED="Yes"
>
</CFSELECT>
</CFFORM>
```

See also: <CFFORM>, <CFGRID>, <CFINPUT>, <CFSLIDER>, <CFTEXTINPUT>, <CFTREE>

<CFSERVLET>, </CFSERVLET>

Description: <CFSERVLET> enables you to execute Java servlets running on a JRun server. Its child tag, <CFSERVLETPARAM>, enables you to pass parameters to the servlets. The attributes for <CFSERVLET> are presented in Table A.84.

Syntax:

```
<CFSERVLET CODE="class name of servlet" JRUNPROXY="proxy server"
 TIMEOUT="timeout in seconds" WRITEOUTPUT="Yes/No"
 DEBUG="Yes or No">
   <CFSERVLETPARAM ...>
   <CFSERVLETPARAM ...>
   ...
</CFSERVLET>
```


TABLE A.84 <CFSERVLET> ATTRIBUTES

Attribute	Description	Notes
CODE	Class name	Required; this identifies the class to execute.
DEBUG	YES or NO	Optional; enables writing JRun connection status information to JRun error log.
JRUNPROXY	IP:portnumber of remote JRUN server	Optional; by default it's assumed that JRun server is running locally. The default port number is 8081.
TIMEOUT	Duration in seconds	Optional; number of seconds JRun should wait for servlet to complete processing.
WRITEOUTPUT	Only output text within <CFOUTPUT>	This attribute is optional and must be blocked. YES or NO is specified.

<CFSERVLET> creates an output structure named CFSERVLET with the following variables:

- CFSERVLET.Output—If WRITEOUTPUT is NO, output is written to this variable.
- CFSERVLET.*servletResponseHeaderName*—Values of any response headers created by the servlet. Each header has an entry in this structure.

Example: This example calls a servlet named SynchronizeContacts on a remote server. It is passed a ContactID by value and other contact information by reference. This is discussed further in <CFSERVLETPARAM>:

```
<CFSERVLET CODE="SynchronizeContacts" JRUNPROXY="236.3.3.4:8081"
  TIMEOUT="30"
>
   <CFSERVLETPARAM  NAME="ContactID" VALUE="CFVar1">
   <CFSERVLETPARAM  NAME="ContactLName" VARIABLE="FORM.LastName">
   <CFSERVLETPARAM  NAME="ContactFName" VALUE="FORM.FirstName">
   <CFSERVLETPARAM  NAME="ContactAddress" VARIABLE="FORM.Address">
   <CFSERVLETPARAM  NAME="ContactCity" VARIABLE="FORM.City">
   <CFSERVLETPARAM  NAME="ContactState" VARIABLE="FORM.State">
   <CFSERVLETPARAM  NAME="ContactZip" VARIABLE="FORM.Zip">
</CFSERVLET>
```

See also: <CFSERVLETPARAM>

<CFSERVLETPARAM>

<CFSERVLETPARAM> enables you to pass parameters by value or by reference to a Java servlet running on a JRun server. It is nested inside a <CFSERVLET>...</CFSERVLET> block. Attributes for <CFSERVLETPARAM> are presented in Table A.85. Possible values for the TYPE attribute are presented in Table A.86.

Syntax:

```
<CFSERVLET ...>
  <CFSERVLETPARAM NAME="servlet param name"
   VALUE="param value">
  <CFSERVLETPARAM NAME="servlet param name"
   TYPE="data type"
   VARIABLE="ColdFusion var name">
</CFSERVLET>
```

TABLE A.85 <CFSERVLET> ATTRIBUTES

Attribute	Description	Notes
NAME	Name of parameter or attribute	Required; this identifies class to execute.
TYPE	Attribute data type	Optional; used with VARIABLE to specify data type. See Table A.83 for a list of valid types.
VALUE	Value of param	Optional; used to specify a value of the parameter. Passed by value to servlet.

Attribute	Description	Notes
VARIABLE	ColdFusion variable name	Optional; used to specify attribute; passed by reference to servlet.

TABLE A.86 DATA TYPES FOR TYPE ATTRIBUTE

Type	Java Equivalent
INT	Java.lang.Integer
DOUBLE	Java.lang.Double
BOOL	Java.lang.Bool
DATE	Java.util.Date
STRING	Java.lang.String
ARRAY	Java.util.Vector
STRUCTURE	Java.util.Hashtable
QUERY RESULT	com.allaire.util.RecordSet

Note

When the servlet needs to be capable of changing the value you pass, use the VARIABLE attribute. When the servlet should not change the value being passed, use VALUE instead.

Note

You must use the servlet API function, setAttribute, to modify values of ColdFusion variables passed as VARIABLE parameters.

Example: See <CFSERVLET> previously

See also: <CFSERVLETPARAM>

<CFSET>

Description: <CFSET> assigns values to variables. <CFSET> can be used for both client variables (type CLIENT) and standard variables (type VARIABLES). Unlike most other ColdFusion tags, <CFSET> takes no attributes—just the name of the variable being assigned and its value.

Syntax:

```
<CFSET "Variable"="Value">
```

Example: The following example creates a local variable containing a constant value:

```
<CFSET MaxDisplay=25>
```

The following example creates a client variable called #BGColor#, which contains a user-specified value and explicitly states the variable type:

```
<CFSET CLIENT.BGColor=FORM.Color>
```

This example stores tomorrow's date in a variable called Tomorrow:

```
<CFSET Tomorrow =Now() + 1>
```

<CFSET> can also be used to concatenate fields:

```
<CFSET VARIABLES.FullName=FORM.FirstName FORM.LastName>
```

Values of different data types also can be concatenated:

```
<CFSET Sentence=FORM.FirstName FORM.LastName & "is" & FORM.age & "years old">
```

Note that when creating complex variables, such as arrays, structures, and queries, you must use a function to create the variables before you can set their values:

```
<CFSET myBreakfast=ArrayNew(1)>
<CFSET myBreakfast[1]="Chipped Beef on Toast">
<CFSET myLunch=StructNew()>
<CFSET myLunch["MainCourse"]="Chili">
<CFSET myDinner=QueryNew("Monday,Tuesday,Wednesday,Thursday,Friday")>
<CFSET temp=QueryAddRow(myDinner)>
<CFSET temp=QuerySetCell(myDinner, "Monday", "Lasagna")>
```

Tip

If you ever find yourself performing a calculation or combining strings more than once in a specific template, you're better off doing it once and assigning the results to a variable with <CFSET>; you can then use that variable instead.

See also: <CFAPPLICATION>, <CFCOOKIE>, <CFPARAM>

<CFSETTING>

Description: <CFSETTING> is used to control various aspects of page processing, such as controlling the output of HTML code in your pages or enabling and disabling debug output. One benefit is managing whitespace that can occur in output pages that are served by ColdFusion. <CFSETTING> attributes are listed in Table A.87.

When using <CFSETTING> to disable an option, be sure you have a matching enable option later in the file.

Syntax:

```
<CFSETTING ENABLECFOUTPUTONLY="Yes|No" SHOWDEBUGOUTPUT="Yes|No">
```

TABLE A.87 <CFSETTING> ATTRIBUTES

Attribute	Description	Notes
CATCHEXCEPTIONBYPATTERN	YES or NO	Optional; forces ColdFusion to override structured exception handling. Defaults to NO.
ENABLECFOUTPUTONLY	YES or NO	Required; forces ColdFusion to only output content in <CFOUTPUT> blocks.
SHOWDEBUGOUTPUT	YES or NO	Optional; when set to NO, suppresses debugging information that normally appears at bottom of page. Defaults to YES.

Note

For each use of <CFSETTING> with ENABLECFOUTPUTONLY set to YES, you must include a matching ENABLECFOUTPUTONLY set to NO. That is, if you used it twice with ENABLECFOUTPUTONLY set to YES, you must use it two more times set to NO to get ColdFusion to display regular HTML output.

Example:

The following demonstrates how <CFSETTING> can be used to control generated whitespace:

```
This text will be displayed
<CFSETTING ENABLECFOUTPUTONLY="Yes">
This text will not be displayed as it is not in a CFOUTPUT block
<CFOUTPUT>This will be displayed</CFOUTPUT>
<CFSETTING ENABLECFOUTPUTONLY="No">
This text will be displayed even though it is not in a CFOUTPUT block
```

See also: <CFSILENT>

<CFSILENT>

Description: Similar to <CFSETTING ENABLECFOUTPUTONLY="YES">, <CFSILENT> is a mechanism for suppressing output. However, it simply suppresses all output that ColdFusion produces within the tag's scope.

Syntax:

```
<CFSILENT>
```

Example: This example demonstrates that <CFSILENT> suppresses all included output in <CFOUTPUT> blocks. It does not, however, suppress the execution of code in <CFOUTPUT> blocks, so the variables #X# and #SENTENCE# are still created and processed:

```
<CFSILENT>
<CFSET x="value">
<CFSET sentence="This is a #x# to be output.">
<P><CFOUTPUT>#sentence#</CFOUTPUT>
</CFSILENT>
<P><CFOUTPUT>#sentence#</CFOUTPUT>
```

See also: <CFSETTING>

<CFSLIDER>

Description: <CFSLIDER> embeds a Java slider control in your HTML forms. Slider controls typically are used to select one of a range of numbers. <CFSLIDER> must be used between <CFFORM> and </CFFORM> tags. Table A.88 lists the entire set of <CFSLIDER> attributes.

Syntax:

```
<CFSLIDER ALIGN="Alignment" BGCOLOR="Background Color" BOLD="Yes or No"
 FONT="Font Face" FONTSIZE="Font Size" GROOVECOLOR="Groove Color"
 HEIGHT="Control Height" HSPACE="Horizontal Spacing" IMG="Groove Image"
 IMGSTYLE="Groove Image Style" ITALIC="Yes or No" LABEL="Slider Label"
 LOOKANDFEEL="Motif or Windows or Metal"
 MESSAGE="Error Message" NAME="Field Name" NOTSUPPORTED="Non Java Browser Code"
 ONERROR="Error Function" ONVALIDATE="Validation Function" RANGE="Numeric Range"
 REFRESHLABEL="Yes or No" SCALE="Increment Value" TEXTCOLOR="Text Color"
 TICKMARKIMAGES="URL list" TICKMARKLABELS="Yes or No or Numeric or label list"
 TICKMARKMAJOR="Yes or No"
 TICKMARKMINOR="Yes or No" VALUE="Initial Value" VERTICAL="Yes or No"
VSPACE="Vertical Spacing" WIDTH="Control Width">
```

TABLE A.88 <CFSLIDER> ATTRIBUTES

Attribute	Description	Notes
ALIGN	Control alignment	Optional; possible values are TOP, LEFT, BOTTOM, BASELINE, TEXTTOP, ABSBOTTOM, MIDDLE, ABSMIDDLE, AND RIGHT
BGCOLOR	Background color	Optional; possible values are BLACK, CYAN, DARKGRAY, GRAY, LIGHTGRAY, MAGENTA, ORANGE, PINK, WHITE, YELLOW, or any color specified in hex format
BOLD	Bold face text	Optional attribute; must be YES or NO if specified; defaults to NO
FONT	Font face	Optional font face to use
FONTSIZE	Font size	Optional font size
GROOVECOLOR	Groove color	Optional attribute; possible values are BLACK, CYAN, DARKGRAY, GRAY, LIGHTGRAY, MAGENTA, ORANGE, PINK, WHITE, YELLOW, or any color specified in RGB form

Attribute	Description	Notes
HEIGHT	Control height	Optional height in pixels
HSPACE	Control horizontal spacing	Optional horizontal spacing in pixels
IMG	Groove image	Optional; filename of image to be used for the slider groove
IMGSTYLE	Grove image style	Optional; can be CENTERED, TILED, or SCALED; default is SCALED
ITALIC	Italic face text	Optional attribute; must be YES or NO if specified; defaults to NO
LABEL	Slider label	Optional; can contain the variable %VALUE%, in which case the current value is displayed as the slider is moved
LOOKANDFEEL	Look style name	Optional; can be MOTIF, WINDOWS, or METAL (Java Swing style). Defaults to WINDOWS.
MESSAGE	Validation failure message	Optional message to display upon validation failure
NAME	Unique control name	Required
NOTSUPPORTED	Text to be used for non-Java browsers	Optional text (or HTML code) to be displayed on non–Java-capable browsers
ONERROR	JavaScript error function	Optional override to your own JavaScript error message function
ONVALIDATE	JavaScript validation function	Optional override to your own JavaScript validation function
RANGE	Range minimum and maximum	Optional range for numeric values only; must be specified as two numbers separated by a comma; defaults to "0,100"
REFRESHLABEL	YES or NO	Optional; if NO and slider is moved, label is not refreshed; default is YES
SCALE	Increment scale integer	Optional; increment amount to use when slider is moved; defaults to 1
TEXTCOLOR	Text color	Optional; possible values are BLACK, CYAN, DARKGRAY, GRAY, LIGHTGRAY, MAGENTA, ORANGE, PINK, WHITE, YELLOW, or any color specified in hex format
TICKMARKIMAGES	URL list	Optional; comma-separated list of URLs of images to use for tick marks.
TICKMARKLABELS	YES or NO, or NUMERIC or list	Optional; YES or NUMERIC results in tick marks based on values of RANGE and SCALE attributes. NO (default) prevents the display of label tick marks. It can also provide a list of labels to be used, such as TWENTY FIVE, FIFTY, SEVENTY FIVE, ONE HUNDRED.

TABLE A.88 CONTINUED

Attribute	Description	Notes
TICKMARKMAJOR	YES or NO	Optional; defaults to NO. Renders major tick marks based on value of SCALE attribute.
TICKMARKMINOR	YES or NO	Optional; defaults to NO. Renders major tick marks based on value of SCALE attribute.
VALUE	Initial value	Optional; initial field value; this value must be within the specified range if RANGE is used
VSPACE	Control vertical spacing	Optional vertical spacing in pixels
WIDTH	Control width	Optional width in pixels

Example: The following example displays a form for searching for contacts based on last name. <CFSLIDER> enables the user to set the maximum rows that will be returned by the query.

```
<H2>Search Contacts</H2>
<CFFORM ACTION="ContactSearch.cfm">

<P>Last name: <CFTEXTINPUT FONT="Verdana" BGCOLOR="white" TEXTCOLOR="Blue"
  NAME="LastName" REQUIRED="Yes">

<P><CFSLIDER NAME="volume" HEIGHT="100" WIDTH="200" FONT="Verdana"
 BGCOLOR="lightgray" TEXTCOLOR="Blue" GROOVECOLOR="White"
 LABEL="Maximum rows %value%"
 RANGE="0,50" SCALE="5">

<P><INPUT TYPE="Submit" VALUE="Search">

</CFFORM>
```

For more information about using <CFSLIDER>, see Chapter 25.

Note

The <CFSLIDER> control is accessible only by users with Java-enabled browsers and must be used in a <CFFORM>.

See also: <CFFORM>, <CFGRID>, <CFINPUT>, <CFSELECT>, <CFTEXTINPUT>, <CFTREE>

<CFSTOREDPROC>, <CFPROCPARAM>, <CFPROCRESULT>, </CFSTOREDPROC>

Description: <CFSTOREDPROC> provides sophisticated support for database-stored procedures. Unlike <CFQUERY> (which can also call stored procedures), <CFSTOREDPROC> and its

supporting tags can pass and retrieve parameters and access multiple resultsets. <CFSTOREDPROC> attributes are listed in Table A.89; <CFPROCPARAM> attributes are listed in Table A.90; and <CFPROCRESULT> attributes are listed in Table A.91.

Syntax:

```
<CFSTOREDPROC CONNECTSTRING="connect string" DATASOURCE="ODBC Data Source"
DBNAME="database name" DBTYPE="type" DBSERVER="dbms"
USERNAME="User Name" PASSWORD="Password" PROCEDURE="Procedure"
PROVIDER="provider" PROVIDERDSN="data source" BLOCKFACTOR="factor"
DEBUG="Yes or No" RETURNCODE="Yes or No">

<CFPROCPARAM TYPE="In|Out|Inout" VARIABLE="variable" DBVARNAME="variable"
 VALUE="value" CFSQLTYPE="type" MAXLENGTH="length" NULL="Yes or No"
 SCALE="decimal places">

<CFPROCRESULT NAME="name" RESULTSET="set" MAXROWS="rows">


</CFSTOREDPROC>
```

TABLE A.89 <CFSTOREDPROC> ATTRIBUTES

Attribute	Description	Notes
BLOCKFACTOR	Number of rows to retrieve at once	Optional; available if using ODBC or Oracle drivers; valid values are 1 to 100; default value is 1.
CONNECTSTRING	Text database connection value	Values required to connect to a dynamic data source; required when DATASOURCE= "__DYNAMIC__".
DATASOURCE	ODBC data source	This optional attribute is used to override the ODBC data source specified when the report was created.
DBNAME	Sybase database name	Optional attribute; only used if using native Sybase drivers.
DBSERVER	Database server	Optional database server; only used if using native database drivers.
DBTYPE	Database type	Optional type; defaults to ODBC; other values are Oracle73, Oracle80, and Sybase11.
DEBUG	YES or NO	Optional attribute; turns on query debugging output.
PASSWORD	ODBC data source password	This optional attribute is used to override the ODBC login password specified in the ColdFusion Administrator.
PROCEDURE	Stored procedure name	Name of stored procedure to execute.
PROVIDER	OLE-DB COM provider	Optional attribute; only used if using OLE-DB.

TABLE A.89 CONTINUED

Attribute	Description	Notes
PROVIDERDSN	Data source OLE-DB COM provider	Optional attribute; only used if using OLE-DB.
RETURNCODE	YES or NO	Optional; indicates whether to populate CFSTOREDPROC.STATUSCODE returned by stored procedure. Defaults to NO.
USERNAME	ODBC data source login name	The optional attribute is used to override the ODBC login name specified in the ColdFusion Administrator.

TABLE A.90 <CFPROCPARAM> ATTRIBUTES

Attribute	Description	Notes
CFSQLTYPE	Variable type	Required. See Table A.92 for a list of supported types.
DBVARNAME	Database variable name	Required to support named notation; corresponds to name of parameter in stored procedure.
MAXLENGTH	Number	Optional; maximum parameter length.
NULL	YES or NO	Optional; indicates whether parameter passed is NULL.
SCALE	Number	Optional; number of decimal places in parameter.
TYPE	Parameter type	Optional; valid values are IN, OUT, and INOUT; defaults to IN.
VALUE	Parameter value	Required for IN and INOUT parameters.
VARIABLE	ColdFusion variable name	Required for OUT or INOUT parameters.

TABLE A.91 <CFPROCRESULT> ATTRIBUTES

Attribute	Description	Notes
MAXROWS	Maximum number of rows	Optional
NAME	Query name	Required
RESULTSET	Resultset number	Optional attribute; specifies the desired resultset; defaults to 1

TABLE A.92 CFSQLTYPE TYPES

Type
CF_SQL_BIGINT
CF_SQL_BITCF_SQL_CHAR
CF_SQL_DATE
CF_SQL_DECIMAL
CF_SQL_DOUBLE
CF_SQL_FLOAT
CF_SQL_IDSTAMP
CF_SQL_INTEGER
CF_SQL_LONGVARCHAR
CF_SQL_MONEY
CF_SQL_MONEY4
CF_SQL_NUMERIC
CF_SQL_REAL
CF_SQL_SMALLINT
CF_SQL_TIME
CF_SQL_TIMESTAMP
CF_SQL_TINYINT
CF_SQL_VARCHAR

For more information about using stored procedures, see Chapter 32, "Working with Stored Procedures."

Example: The first example executes a simple stored procedure named GETEMPLOYEES. This is just a SELECT query that builds a resultset named GETEMPLOYEES:

```
<CFSTOREDPROC DATASOURCE="OWS" PROCEDURE="GetEmployees">
   <CFPROCRESULT NAME="GetEmployees">
</CFSTOREDPROC>
```

The next example, shown in Listing A.20, invokes a stored procedure named GETNEXTNUMBER, which increments a value in a table by one and then sends the new number back to the calling routine. It takes an input parameter named TABLENAME and generates a value named BUDGETCATEGORYID, which you can then use in your application.

LISTING A.20 EXECUTING A STORED PROCEDURE WITH PARAMETERS

```
<CFSTOREDPROC
   PROCEDURE="GetNextNumber"
   DATASOURCE="OWS"
   RETURNCODE="YES"
```

LISTING A.20 CONTINUED

```
>
<CFPROCPARAM
   TYPE="IN"
   DBVARNAME=@TblName
   VALUE=#KeyName#
   CFSQLTYPE=CF_SQL_CHAR
   MAXLENGTH="20"
   NULL="no"
>
<CFPROCPARAM
   TYPE="OUT"
   VARIABLE=BudgetCategoryID
   DBVARNAME=@intNextNmbr
   CFSQLTYPE=CF_SQL_INTEGER
>
</CFSTOREDPROC>
```

See also: <CFQUERY>

<CFSWITCH>, <CFCASE>, </CFCASE>, <CFDEFAULTCASE>, </CFDEFAULTCASE>, </CFSWITCH>

Description: <CFSWITCH> is used to create case statements in ColdFusion. Every <CFSWITCH> must be terminated with a </CFSWITCH>. The individual case statements are specified using the <CFCASE> tag; a default case can be specified using the <CFDEFAULTCASE> tag. <CFSWITCH> attributes are listed in Table A.93; <CFCASE> attributes are listed in Table A.94.

Syntax:

```
<CFSWITCH EXPRESSION="expression">
<CFCASE VALUE="value">
… HTML or CFML code here …
</CFCASE>
<CFDEFAULTCASE>… HTML or CFML code here …
</CFDEFAULTCASE>
</CFSWITCH>
```

TABLE A.93 <CFSWITCH> ATTRIBUTES

Attribute	Description	Notes
EXPRESSION	Case expression	This attribute is required.

TABLE A.94 <CFCASE> ATTRIBUTES

Attribute	Description	Notes
VALUE	Case value	This attribute is required.

Example: The following example checks to see whether a state is a known state and displays an appropriate message:

```
<CFSWITCH EXPRESSION="#Ucase(state)#">
  <CFCASE VALUE="CA">California</CFCASE>
  <CFCASE VALUE="FL">Florida</CFCASE>
  <CFCASE VALUE="MI">Michigan</CFCASE>
  <CFCASEDEFAULT>One of the other 47 states</CFCASEDEFAULT>
</CFSWITCH>
```

Tip

You can replace some long <CFSWITH><CFCASE...> statements with a structure. Look at this code:

```
<CFSWITCH EXPRESSION="#FORM.Meal#">
<CFCASE VALUE="Breakfast">
   <CFSET Drink="Coffee">
</CFCASE>
<CFCASE VALUE="Lunch">
   <CFSET Drink="Iced Tea">
</CFCASE>
<CFCASE VALUE="Snack">
   <CFSET Drink="Coke">
</CFCASE>
<CFCASE VALUE="Dinner">
   <CFSET Drink="Wine">
</CFCASE>
    </CFSWITCH>
```

Now, assume you have built this struture:

```
<CFSET Beverages=StructNew()>
<CFSET Beverages["Breakfast"]="Coffee">
<CFSET Beverages["Lunch"]="Iced Tea">
<CFSET Beverages["Snack"]="Coke">
<CFSET Beverages["Dinner"]="Wine">
<CFSET meal=FORM.Lunch>
```

Given this structure, the following single line of code eliminates the need for the lengthy switch/case logic shown previously:

```
<CSET Drink=Beverages["#meal#"]>
```

See also: <CFIF>

<CFTABLE>, <CFCOL>, </CFTABLE>

Description: <CFTABLE> enables you to easily create tables in which dynamically generated query data is displayed. <CFTABLE> can create HTML tables (using the <TABLE> tag) or pre-formatted text tables (using <PRE>, </PRE>) that display on all browsers. Using <CFTABLE> involves two tags: <CFTABLE> defines the table itself, and one or more <CFCOL> tags define the table columns. The <CFTABLE> attributes are listed in Table A.95; the <CFCOL> attributes are listed in Table A.96.

Syntax:

```
<CFTABLE QUERY="Query Name" MAXROWS="Maximum Rows" COLSPACING="Column Spacing"
 COLHEADERS HEADERLINES ="Header Lines" HTMLTABLE>
<CFCOL HEADER="Header Text" WIDTH ="Width" ALIGN ="Alignment" TEXT="Body Text">
</CFTABLE>
```

TABLE A.95 <CFTABLE> ATTRIBUTES

Attribute	Description	Notes
BORDER	Adds border to HTML table	Optional; use when specifying HTMLTABLE.
COLHEADERS	Displays column headers	Optional; column headers are displayed; as specified in <CFCOL>.
COLSPACING	Spaces between columns	Optional; overrides the default column spacing of 2 if present.
HEADERLINES	Number of header lines	Optional; defaults to 2; one for the header and a blank row between the header and the body. You can increase this number if needed.
HTMLTABLE	Creates an HTML table	Optional; an HTML table is created if this attribute is present. If not, a preformatted text table is created.
MAXROWS	Maximum number of table rows	Optional; specifies the maximum number of rows to be displayed in the table.
QUERY	<CFQUERY> name	Required; the name of the query from which to derive the table body text.

TABLE A.96 <CFCOL> ATTRIBUTES

Attribute	Description	Notes
HEADER	Header text	Optional; text for header.
WIDTH	Number of characters	Data wider than this for column width value will be truncated.
ALIGN	Column alignment	Left, Right, or Center.
TEXT	Text that is to be displayed in the column	This can be a combination of text and ColdFusion variables.

Example: This example queries a database and then displays the output using <CFTABLE> and <CFCOL>:

```
<CFQUERY NAME="GetFilms" DATASOURCE="OWS">
  SELECT FilmID, MovieTitle
  FROM Films
</CFQUERY>
```

```
<H1>Use of CFTABLE and CFCOL</H1>
<CFTABLE QUERY="GetFilms">
  <CFCOL HEADER="Film ID" WIDTH="8" ALIGN="Right" TEXT="<EM>#FilmID#</EM>">
  <CFCOL HEADER="Name" WIDTH="30" ALIGN="Left" TEXT="#MovieTitle#">
</CFTABLE>
```

Tip

The <CFTABLE> tag is an easy and efficient way to create tables for displaying query results. You should create HTML tables manually for greater control over table output, including cell spanning, text and background colors, borders, background images, and nested tables.

Note

See Chapter 11 for more information on using tables to display data.

See also: <CFOUTPUT>, <CFQUERY>

<CFTEXTINPUT>

Description: <CFTEXTINPUT> embeds a highly configurable Java text input control in your HTML forms. <CFTEXTINPUT> must be used between <CFFORM> and </CFFORM> tags. Unlike the standard HTML INPUT, <CFTEXTINPUT> lets you configure the exact height and width of the edit control, as well as color, font, size, and spacing. <CFTEXTINPUT> also can automatically generate field JavaScript validation code. Table A.97 lists the entire set of <CFTEXTINPUT> attributes.

Syntax:

```
<CFTEXTINPUT ALIGN="Alignment" BGCOLOR="Background Color" BOLD="Yes or No"
 FONT="Font Face" FONTSIZE="Font Size" HEIGHT="Control Height"
 HSPACE="Horizontal Spacing" ITALIC="Yes or No" MAXLENGTH="Maximum Length"
 MESSAGE="Error Message" NAME="Field Name" NOTSUPPORTED="Non Java Browser Code"
 ONERROR="Error Function" ONVALIDATE="Validation Function" RANGE="Numeric Range"
 REQUIRED="Yes or No" SIZE="Field Size" TEXTCOLOR="Text Color"
 VALIDATE="Validation Type" VALUE="Initial Value" VSPACE="Vertical Spacing"
 WIDTH="Control Width">
```


TABLE A.97 <CFTEXTINPUT> ATTRIBUTES

Attribute	Description	Notes
ALIGN	Control alignment	Optional. Possible values are TOP, LEFT, BOTTOM, BASELINE, TEXTTOP, ABSBOTTOM, MIDDLE, ABSMIDDLE, and RIGHT.
BGCOLOR	Background color	Optional; possible values are BLACK, CYAN, DARKGRAY, GRAY, LIGHTGRAY, MAGENTA, ORANGE, PINK, WHITE, YELLOW, or any color specified in RGB form.

TABLE A.97 <CFTEXTINPUT> ATTRIBUTES

Attribute	Description	Notes
BOLD	YES or NO	Optional; specifies use of bold-faced text; defaults to NO.
FONT	Font face	Optional; font face to use.
FONTSIZE	Font size number	Optional; font size.
HEIGHT	Number of pixels	Optional; controls height.
HSPACE	Number of pixels	Optional; specifies horizontal spacing in pixels.
ITALIC	YES or NO	Optional; specifies use of italic font; defaults to NO.
MAXLENGTH	Number of characters	Optional; maximum number of characters to allow.
MESSAGE	Validation failure message	Optional; message to display upon validation failure.
NAME	Unique control name	Required.
NOTSUPPORTED	Text to be used for non-Java browsers	Optional; text (or HTMLcode) to be displayed on non–Java-capable browsers.
ONERROR	JavaScript error function	Optional; specifies your own JavaScript error message function to override ColdFusion's.
ONVALIDATE	JavaScript validation function	Optional; specifies your own JavaScript validation function to override ColdFusion's.
RANGE	Min number, Max number	Optional; range for numeric values only; specified as two numbers separated by a comma.
REQUIRED	YES or NO	Optional; YES indicates that a value is required. Defaults to NO.
SIZE	Number of characters	Optional; field size; number of characters to display before needing horizontal scrolling.
TEXTCOLOR	Text color	Optional attribute; possible values are BLACK, CYAN, DARKGRAY, GRAY, LIGHTGRAY, MAGENTA, ORANGE, PINK, WHITE, YELLOW, or any color specified in RGB form.
VALIDATE	Field validation	Optional field validation. If specified, must be any of the validation types listed in Table A.98.
VALUE	Initial value	Optional.

Attribute	Description	Notes
VSPACE	Number of pixels	Optional; vertical spacing for control.
WIDTH	Number of pixels	Optional; controls width.

TABLE A.98 <CFTEXTINPUT> VALIDATION TYPES

Type	Description
CREDITCARD	Correctly formatted credit card number verified using mod10 algorithm
DATE	Date in mm/dd/yy format
EURODATE	European date in dd/mm/yy format
FLOAT	Number with decimal point
INTEGER	Number with no decimal point
SOCIAL_SECURITY_NUMBER	Social security number formatted as 999-99-9999 (using hyphens or spaces as separators)
TELEPHONE	Phone number in 999-999-9999 format (using hyphens or spaces as separators); area code and exchange must not begin with 0 or 1
TIME	Time in hh:mm or hh:mm:ss format
ZIPCODE	U.S. ZIP code, either 99999 or 99999-9999 format

Example: The example shown in Listing A.21 displays a form in which the user is to enter a credit card number and expiration date. Two <CFTEXTINPUT> controls are used—one for each field. Attributes are used to make the fields required and control their sizes and font colors. In addition, a custom validation function, CheckExpDate(), is specified to validate the expiration date.

LISTING A.21 USING <CFTEXTINPUT> TO VALIDATE DATA

```
<SCRIPT LANGUAGE="JavaScript">
 function CheckExpDate() {
    if (blah blah blah) {return true} else {return false};
 }
</SCRIPT>
<CFFORM ACTION="process.cfm" METHOD="POST">

<P>Enter your credit card number:
<CFTEXTINPUT NAME="CCnumber" HEIGHT="25" WIDTH="200" FONT="Verdana"
 BGCOLOR="white"
 TEXTCOLOR="Blue" VALIDATE="creditcard" REQUIRED="Yes"
 MESSAGE="Please enter a valid credit card number">

<P>Enter the expiration date (mm/yy):
<CFTEXTINPUT NAME="ExpDate" HEIGHT="25" WIDTH="90" FONT="Verdana" BGCOLOR="white"
 TEXTCOLOR="Blue" MAXLENGTH="5" REQUIRED="Yes"  ONVALIDATE="CheckExpDate()"
 MESSAGE="Please enter a valid expiration date in the form mm/yy">
```

LISTING A.21 CONTINUED

```
<INPUT TYPE="Submit" VALUE="Buy it!">
</CFFORM>
```

For more information about using <CFTEXTINPUT>, see Chapter 23.

Note

The <CFTEXTINPUT> control is accessible only by users with Java-enabled browsers.

See also: <CFFORM>, <CFGRID>, <CFINPUT>, <CFSELECT>, <CFSLIDER>, <CFTREE>

<CFTHROW>

Description: <CFTHROW> is used to force an error condition in a <CFTRY>, </CFTRY> block. Program control is then handed to a <CFCATCH> in which TYPE is set to APPLICATION, ANY, or a custom type. <CFTHROW> attributes are listed in Table A.99.

Syntax:

```
<CFTHROW DETAIL="Error description" ERRORCODE="Code"
 EXTENDEDINFO="More error information" MESSAGE="message"
 TYPE="Error type">
```

TABLE A.99 <CFTHROW> ATTRIBUTES

Attribute	Description	Notes
DETAIL	Description of error	Optional; detailed description of error.
ERRORCODE	Custom error code	Optional; developer-specified error code.
EXTENDEDINFO	Additional info	Optional; additional information on the error.
MESSAGE	Error message	Optional; developer-specified description.
TYPE	Type of error condition	Optional; must be either APPLICATION or a custom type. If you use APPLICATION, you do not have to specify a TYPE for <CFCATCH>.

Example: This code checks for the existence of a specified SESSION variable. If it doesn't exist, a custom, developer-specified error is thrown and trapped in the <CFCATCH> of the specified, custom type:

```
<CFTRY>
   <CFIF NOT IsDefined("SESSION.UserID")>
      <CFTHROW TYPE="AppSecurity" MESSAGE="Invalid auhthorization"
      ERRORCODE="210" DETAIL="Access is restricted to authenticated users">
   </CFIF>
```

```
   ...
   ...
   <CFCATCH TYPE="AppSecurity">
      <CFOUTPUT>
      <p>(#CFCATCH.ErrorCode#) <B>#CFCATCH.Message#</B>
      <p>#CFCATCH.Detail#
      </CFOUTPUT>
   </CFCATCH>
</CFTRY>
...
```

See also: <CFTRY>, <CFCATCH>

<CFTRANSACTION>, </CFTRANSACTION>

Description: <CFTRANSACTION> enables you to group multiple <CFQUERY> uses into a single transaction. Any <CFQUERY> tags placed between <CFTRANSACTION> and </CFTRANSACTION> tags are rolled back if an error occurs. The <CFTRANSACTION> attributes are list in Table A.100.

Syntax:

```
<CFTRANSACTION ISOLATION="Lock Type" ACTION="Action">
Queries
</CFTRANSACTION>
```

TABLE A.100 <CFTRANSACTION> ATTRIBUTES

Attribute	Description	Notes
ACTION	Type of action	Optional; BEGIN (default), COMMIT, or ROLLBACK
ISOLATION	Type of ODBC lock	Optional lock type; possible values are READ_UNCOMMITTED, READ_COMMITTED, REPEATABLE_READ, and SERIALIZABLE

ACTION of BEGIN signifies the start transaction. You can nest <CFTRANSACTION> tags and force commits or rollbacks by setting ACTION to COMMIT or ROLLBACK in your nested transactions.

Note

When <CFTRANSACTION> is used in combination with ColdFusion's error handling, you can tell when a database transaction fails. This gives you control over whether queries grouped into transactions are to be committed or rolled back.

Note

Not all lock types are supported by all ODBC drivers. Consult your database documentation before using the ISOLATION attribute.

Example: The first example demonstrates a simple use of <CFTRANSACTION> to ensure that either both queries succeed or the transaction is canceled:

```
<CFTRANSACTION>
 <CFQUERY NAME="InsertContact" DATASOURCE="OWS">
  INSERT INTO Contacts (FirstName, LastName, Phone, UserLogin)
  VALUES ('Joe', 'Blow', '333-112-1212', 'JoeBlow')
 </CFQUERY>

 <CFQUERY NAME="InsertActor" DATASOURCE="OWS">
  INSERT INTO Actors (NameFirst, NameLast, Gender)
  VALUES ('Joe', 'Blow', 'M')
 </CFQUERY>
</CFTRANSACTION>
```

The second example, shown in Listing A.22, does the same thing but demonstrates the use of `<CFTRANSACTION ACTION="Rollback">` and `<CFTRANSACTION ACTION="Commit">`. Note that these aren't really required in this simple example; it's just being done this way to demonstrate the technique.

LISTING A.22 USING <CFTRANSACTION> TO CREATE DATABASE TRANSACTIONS

```
<CFSET DoCommit="Yes">
<CFTRANSACTION ACTION="BEGIN">
<CFTRY>
 <CFQUERY NAME="InsertContact" DATASOURCE="OWS">
  INSERT INTO Contacts (FirstName, LastName, Phone, UserLogin)
  VALUES ('Joe', 'Blow', '333-112-1212', 'JoeBlow')
 </CFQUERY>

 <CFQUERY NAME="InsertActor" DATASOURCE="OWS">
  INSERT INTO Actors (NameFirst, NameLast, Gender)
  VALUES ('Joe', 'Blow', 'M')
 </CFQUERY>

 <CFCATCH TYPE="DATABASE">
   <CFTRANSACTION ACTION="ROLLBACK"/>
   <CFSET DoCommit="No">
 </CFCATCH>
</CFTRY>

<CFIF DoCommit>
   <CFTRANSACTION ACTION="COMMIT"/>
<CFELSE>
   <p>Failure
</CFIF>

</CFTRANSACTION>
```

Note

Note the use of abbreviated ending tag syntax in Listing A.22. This enables you to omit the ending tag. It's valid in this situation because there is no body between the beginning and ending tag.

See also: `<CFSTOREDPROC>`

<CFTREE>, <CFTREEITEM>, </CFTREE>

Description: <CFTREE> embeds a Java tree control in your HTML forms constructed with <CFFORM>. The tree control is similar to the Explorer window used in several versions of Windows and is in fact used in the ColdFusion Administrator (when browsing for an ODBC data source). The tree is made up of root entries and branches that can be expanded or closed. Branches can be nested. Each branch has a graphic displayed next to it; you can select from any of the supplied graphics or use any of your own.

<CFTREE> trees are constructed using two tags. <CFTREE> creates the tree control, and <CFTREEITEM> adds the entries into the tree. Trees can be populated one branch at a time or by using query results. <CFTREEITEM> must be used between <CFTREE> and </CFTREE> tags. <CFTREE> attributes are listed in Table A.101; <CFTREEITEM> attributes are listed in Table A.102.

Syntax:

```
<CFTREE ALIGN="Alignment" APPENDKEY="Yes or No" BOLD="Yes or No" BORDER="Yes|No|
 COMPLETEPATH="Yes or No" DELIMITER="Delimiter Character" FONT="Font Face"
 FONTSIZE="Font Size" HEIGHT="Control Height" HIGHLIGHTHREF="Yes or No"
 HSPACE="Horizontal Spacing" HSCROLL="Yes or No" ITALIC="Yes or No"
 MESSAGE="Error Message" NAME="Field Name" NOTSUPPORTED="Non Java Browser Code"
 ONERROR="Error Function" ONVALIDATE="Validation Function" REQUIRED="Yes or No"
 VSPACE="Vertical Spacing" WIDTH="Control Width">
<CFTREEITEM DISPLAY="Display Text EXPAND="Yes or No" HREF="URL" IMG="Images"
 IMGOPEN="Images" QUERY="Query Name" QUERYASROOT="Yes or No" TARGET="Target Name"
 PARENT="Parent Branch" VALUE="Values">
```

TABLE A.101 <CFTREE> ATTRIBUTES

Attribute	Description	Notes
ALIGN	Control alignment	Optional. Possible values are TOP, LEFT, BOTTOM, BASELINE, TEXTTOP, ABSBOTTOM, MIDDLE, ABSMIDDLE, and RIGHT.
APPENDKEY	YES or NO	Optional; appends item key to URL. If YES, variable named CFTREEITEMKEY is appended to the URL containing the item selected; defaults to YES.
BOLD	YES or NO	Optional; makes text bold; defaults to NO.
BORDER	YES or NO	Optional; displays border; defaults to YES.
COMPLETEPATH	YES or NO	Optional; passes the full tree path to the selected item when set to YES; defaults to NO.

TABLE A.101 CONTINUED

Attribute	Description	Notes
DELIMITER	Path delimiter character	Optional; defaults to \.
FONT	Font face name	Optional font face to use.
FONTSIZE	Number of pixels	Optional font size.
HEIGHT	Number of pixels	Optional height in pixels.
HIGHLIGHTHREF	YES or NO	Optional; links are highlighted and underlined if YES; defaults to YES.
HSPACE	Number of pixels	Optional; horizontal spacing in pixels.
HSCROLL	YES or NO	Optional; displays horizontal scrollbar; default is YES.
ITALIC	YES or NO	Optional; italicizes text; defaults to NO.
MESSAGE	Validation failure message	Optional; message to display upon validation failure.
NAME	Unique control name	Required.
NOTSUPPORTED	Text message	Optional; text (or HTML code) to be displayed on non-Java–capable browsers.
ONERROR	JavaScript error function	Optional override to your own JavaScript error message function.
ONVALIDATE	JavaScript validation function	Optional override to your own JavaScript validation function.
REQUIRED	YES or NO	Optional; a selection will be required when set to YES; defaults to NO.
VSPACE	Number of pixels	Optional; vertical spacing in pixels.
VSCROLL	YES or NO	Optional; displays a vertical scrollbar when set to YES; default is YES.
WIDTH	Number of pixels	Optional; width of control.

TABLE A.102 <CFTREEITEM> ATTRIBUTES

Attribute	Description	Notes
DISPLAY	Display text	Optional attribute. Defaults to value is not specified. If populating with a query resultset, this value should be a comma-delimited list of values—one for each tree item.
EXPAND	YES or NO	Optional; branch is initially expanded if YES; defaults to NO.

Attribute	Description	Notes
HREF	Item URL	Optional; URL to go to when an item is selected; if populating with a query resultset, this value can be a comma-delimited list of URLs (one for each tree item), or it can be a column name. In that case, it is populated dynamically.
IMG	Image	Optional; image to be displayed; if populating with a query resultset, this value should be a comma-delimited list of images, one for each tree level; images can be CD, COMPUTER, DOCUMENT, ELEMENT, FIXED, FOLDER, FLOPPY, REMOTE, or any image file of your own.
IMGOPEN	Open image	Optional; image to be displayed when branch is open; if populating with a query resultset, this value should be a comma-delimited list of images, one for each tree level; same selections as IMG. If omitted, the IMG image is used.
PARENT	Parent item name	Optional; name of parent item to attach this branch to.
QUERY	Query name	Optional query name to be used to populate the list.
QUERYASROOT	YES or NO	Optional; if YES, query name itself is the tree root branch; defaults to NO. Prevents having to create a parent tree item.
TARGET	Target for HREF	Optional; the window in which to open the link; this value can be a comma-delimited list of targets if populating with a query resultset, one for each tree item.
VALUE	Value to be returned	Required; value to be returned when item is selected. Value should be a comma-delimited list of values, one for each tree item, if populating with a query resultset.

Example: This example creates a simple Java tree control with three branches:

```
<CFFORM ACTION="process.cfm">
<CFTREE NAME="states">
<CFTREEITEM VALUE="US">
<CFTREEITEM VALUE="CA" DISPLAY="California" PARENT="US">
<CFTREEITEM VALUE="MI" DISPLAY="Michigan" PARENT="US">
<CFTREEITEM VALUE="NY" DISPLAY="New York" PARENT="US">
```

```
</CFTREE>
<INPUT TYPE="Submit" VALUE="Select a State">
</CFFORM>
```

This next example populates a tree with a query called `Users`:

```
<CFQUERY NAME="GetUsers" DATASOURCE="ows">
   SELECT ContactID, FirstName & ' ' & LastName as userName, UserLogin
   FROM Contacts
</CFQUERY>
<CFFORM ACTION="process.cfm">
<CFTREE NAME="peopletree" HSPACE="20" HSCROLL="no" VSCROLL="Yes" DELIMITER="?"
 BORDER="Yes">
<CFTREEITEM VALUE="UserName" QUERYASROOT="Yes" QUERY="GetUsers"
 IMG="folder,document" HREF="EditUser.cfm">
</CFTREE>
</CFFORM>
```

The last example, shown in Listing A.23, demonstrates the use of `<CFTREE>` to produce a tree with nested branches. In this case, it displays a list of movies and the actors in the movies.

LISTING A.23 PRODUCING A TREE WITH NESTED BRANCHES

```
<CFQUERY NAME="GetFilmActors" DATASOURCE="ows">
  SELECT FA.FilmID, A.NameFirst, A.NameLast, F.MovieTitle
  FROM (Actors A INNER JOIN FilmsActors FA ON A.ActorID = FA.ActorID)
  INNER JOIN Films F ON FA.FilmID = F.FilmID
</CFQUERY>

<CFFORM ACTION="process.cfm">
<CFTREE NAME="FilmActors" HEIGHT="150" WIDTH="300" HIGHLIGHTHREF="Yes">
<CFOUTPUT QUERY="GetFilmActors" GROUP="FilmID">
   <CFTREEITEM VALUE="#MovieTitle#" EXPAND="No">
   <CFOUTPUT>
      <CFTREEITEM IMG="Document" VALUE="#NameFirst# #NameLast#"
HREF="ViewActor.cfm"
PARENT="#MovieTitle#">
   </CFOUTPUT>
</CFOUTPUT>
</CFTREE>
</CFFORM>
```

For more information about using `<CFTREE>`, see Chapter 23.

Note

The `<CFTREE>` control is accessible only by users with Java-enabled browsers.

See also: `<CFFORM>`, `<CFGRID>`, `<CFINPUT>`, `<CFSELECT>`, `<CFSLIDER>`, `<CFTEXTINPUT>`

<CFTRY>, <CFCATCH>, </CFCATCH>, </CFTRY>

Description: <CFTRY> is used to catch exceptions thrown by ColdFusion or explicitly with <CFTHROW> or <CFRETHROW>. All code between <CFTRY> and </CFTRY> can throw exceptions, and exceptions are caught by <CFCATCH> blocks. Explicit <CFCATCH> blocks can be created for various error types, or one block can catch all errors. <CFCATCH> attributes are listed in Table A.103. A special structure variable, CFCATCH, is available in your <CFCATCH> blocks. Its elements are described in Table A.105.

Syntax:

```
<CFTRY> <CFCATCH TYPE="type"> </CFCATCH> </CFTRY>
```

TABLE A.103 <CFCATCH> ATTRIBUTES

Attribute	Description	Notes
TYPE	Exception type	Optional; values listed in Table A.104

TABLE A.104 <CFCATCH> TYPE VALUES

Type
ANY
APPLICATION
Custom_Type
DATABASE
EXPRESSION
LOCK
MISSINGINCLUDE
OBJECT
SECURITY
SYNCHRONIZATION
TEMPLATE

TABLE A.105 CFCATCH VARIABLE ELEMENTS

Type	Only for Type	Description
DETAIL		A detailed error message; helps determine which tag threw the exception
ERRNUMBER	EXPRESSION	Internal expression error number
ERRORCODE	*Custom type*	Developer-specified error code
EXTENDEDINFO	APPLICATION *Custom type*	Developer's custom error message

TABLE A.105 CONTINUED

Type	Only for Type	Description
LOCKNAME	LOCK	Name of the affected lock; set to anonymous if the lock was unnamed
LOCKOPERATION	LOCK	Operation that failed; TIMEOUT, CREATE MUTEX, or UNKNOWN
MESSAGE		Diagnostic message; can be null
MISSINGFILENAME	MISSINGINCLUDE	Name of file that could not be included
NATIVEERRORCODE	DATABASE	The native error code from the database driver; -1 if no native code provided
SQLSTATE	DATABASE	Another error code from the database driver; -1 if no native code provided
TAGCONTEXT		Tag stack; name and position of each tag in the stack
TYPE		Exception type, as specified in <CFCATCH>

Example: This example traps for an error when attempting to create a new directory programmatically:

```
<CFTRY>
<CFDIRECTORY ACTION="CREATE" DIRECTORY="#FORM.UserDir#">
<CFCATCH TYPE="ANY">
   <CFIF CFCATCH.Detail CONTAINS "when that file already exists">
      <P>Cannot create directory: this directory already exists.
   <CFELSE>
      <CFOUTPUT>#CFCATCH.Detail#</CFOUTPUT>
   </CFIF>
   <CFABORT>
</CFCATCH>
</CFTRY>
<P>Directory created.
```

You will find other useful examples in the entries for <CFRETHROW>, <CFAUTHENTICATE>, and <CFTRANSACTION>.

See also: <CFTHROW>, <CFRETHROW>

<CFUPDATE>

Description: <CFUPDATE> updates a single row to a database table; it requires that the database and table names be provided. All other attributes are optional. The full list of <CFUPDATE> attributes is explained in Table A.106.

Syntax:

```
<CFUPDATE CONNECTSTRING="ODBC connection string" DATASOURCE="ODBC Data Source"
 DBNAME="database name"
 DBTYPE="type" DBSERVER="dbms" FORMFIELDS="List of File to Update"
 PASSWORD="Password" PROVIDER="provider" PROVIDERDSN="data source"
 TABLENAME="Table Name" TABLEOWNER="owner" TABLEQUALIFIER="qualifier"
 USERNAME="User Name">
```

TABLE A.106 <CFUPDATE> ATTRIBUTES

Attribute	Description	Notes
CONNECTSTRING	Text database connection value	Values required to connect to a dynamic data source; required when DATASOURCE= "__DYNAMIC__".
DATASOURCE	Name of ODBC data source	Required; can be an existing data source or defined dynamically.
DBNAME	Sybase database name	Optional; only used if using native Sybase drivers.
DBSERVER	Database server	Optional database server; only used if using native database drivers.
DBTYPE	Database type	Optional; defaults to ODBC; other values are ORACLE73, ORACLE80, SYBASE11, OLEDB, DB2, and INFORMIX73.
FORMFIELDS	List of fields to insert	Optional; specifies which fields are to be updated if they are present. Any fields present that are not in the list will not be updated.
PASSWORD	ODBC data source password	Optional; used to override the ODBC login password specified in the ColdFusion Administrator.
PROVIDER	OLE-DB COM provider	Optional; used only if using OLE-DB.
PROVIDERDSN	Data source OLE-DB COM provider	Optional; used only if using OLE-DB.
TABLENAME	Name of table to insert data into	Required; some ODBC data sources require fully qualified table names.
TABLEOWNER	Table owner name	Optional; used by databases that support table ownership.
TABLEQUALIFIER	Table qualifier	Optional; used by databases that support full qualifiers.
USERNAME	ODBC data source login name	Optional; used to override the ODBC login name specified in the ColdFusion Administrator.

Note

For <CFUPDATE> to work correctly, your form field names must match the column names in the destination table, and the primary key value of the row to be updated must be specified.

Tip

If your form contains fields that are not part of the table you are updating, use the FORMFIELDS attribute to instruct ColdFusion to ignore those fields.

Tip

For more control over updating rows in a database table, you can use the <CFQUERY> tag specifying UPDATE as the SQL statement.

Example: In the first example, a simple data entry form is used to collect data to be updated in the Merchandise table:

```
<FORM ACTION="update_act.cfm" METHOD="post">
<P>film id: <INPUT TYPE="Text" MAXLENGTH="5" NAME="filmid" value="18">
<P>merchandise name: <INPUT TYPE="Text" NAME="MerchName" MAXLENGTH="100">
<P>merchandise desc: <INPUT TYPE="Text" NAME="MerchDescription">
<P>merchandise price: <INPUT TYPE="Text" NAME="MerchPrice">
<p><INPUT TYPE="Submit">
</FORM>
```

This <CFUPDATE> tag updates the table from this form data:

```
<CFUPDATE DATASOURCE="OWS" TABLENAME="Merchandise">
```

If, however, the form contains additional fields that don't correspond to the fields in the Merchandise table, you must use the FORMFIELDS attribute to identify which form fields are to be inserted:

```
<CFUPDATE DATASOURCE="OWS" TABLENAME="Merchandise"
  FORMFIELDS="FilmID,MerchName,MerchDescription,MerchPrice">
```

For more information about the <CFUPDATE> tag, see Chapter 14.

See also: <CFUPDATE>, <CFQUERY>

<CFWDDX>

Description: <CFWDDX> is used to serialize and deserialize ColdFusion data structures to the XML-based WDDX format. The attributes for this tag are presented in Table A.107. The ACTION attribute specifies the action to be performed. The values for the ACTION attribute are presented in Table A.108.

Syntax:

```
<CFWDDX ACTION="action" INPUT="input" OUTPUT="output" TOPLEVELVARIABLE="name"
 USETIMEZONEINFO="Yes or No">
```

TABLE A.107 <CFWDDX> ATTRIBUTES

Attribute	Description	Notes
ACTION	Action	Required; actions are listed in Table A.87
INPUT	Input value	Required
OUTPUT	Output variable	Required if ACTION is WDDX2CFML
TOPLEVELVARIABLE	JavaScript top-level variable	Required if ACTION is WDDX2JS or CFML2JS
USETIMEZONEINFO	YES or NO	Optional; indicates whether to include timezone information (in ISO8601 format) when data is being serialized; defaults to YES

TABLE A.108 <CFWDDX> ACTIONS

Action	Description
CFML2JS	Serializes CFML to JavaScript format
CFML2WDDX	Serializes CFML to WDDX format
WDDX2CFML	Deserializes WDDX to CFML
WDDX2JS	Deserializes WDDX to JavaScript

Example: The example shown in Listing A.109 demonstrates serializing and deserializing a ColdFusion query result.

LISTING A.24 SERIALIZING/DESERIALIZING A QUERY WITH <CFWDDX>

```
<CFQUERY NAME="GetContacts" DATASOURCE="OWS">
   SELECT ContactID, FirstName, LastName
   FROM Contacts
</CFQUERY>

<P>The recordset data is:
<UL>
<CFOUTPUT QUERY="GetContacts">
   <li>#ContactID#, #FirstName#, #LastName#
</CFOUTPUT>
</UL>

<P>Serializing CFML data
<CFWDDX ACTION="cfml2wddx" INPUT="#GetContacts#" OUTPUT="Contacts">

<P>Resulting WDDX packet is:
<XMP><CFOUTPUT>#Contacts#</CFOUTPUT></XMP>

<P>Deserializing WDDX packet<P>
<CFWDDX ACTION="WDDX2CFML" INPUT="#Contacts#" OUTPUT="NewContacts">
```

LISTING A.24 CONTINUED

```
<P>The recordset data is:
<UL>
<CFOUTPUT QUERY="GetContacts">
   <LI>#ContactID#, #FirstName#, #LastName#<BR>
</CFOUTPUT>
</UL>
```