ADOBE® COLDFUSION™ 8
CFML Reference

Adobe® ColdFusion® CFML Reference

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

Chapter 4: ColdFusion Functions

Chapter 5: AJAX JavaScript Functions

Chapter 6: ColdFusion Flash Form Style Reference

Chapter 1: Introduction

The *CFML Reference* is your primary ColdFusion Markup Language (CFML) reference. Use this manual to learn about CFML tags and functions, ColdFusion expressions, and using JavaScript objects for WDDX in Adobe® ColdFusion® 8. It also provides detailed references for Java™ and C++ CFX interfaces.

About Adobe ColdFusion 8 documentation

The ColdFusion documentation is designed to provide support for the complete spectrum of participants.

Documentation set

The ColdFusion documentation set includes the following titles:

Manual	Description
Installing and Using ColdFusion	Describes system installation and basic configuration for Windows®, Macintosh®, Solaris™, Linux®, and AIX®.
Configuring and Administering ColdFusion	Part I describes how to manage the ColdFusion environment, including connecting to your data sources and configuring security for your applications. Part II describes Verity search tools and utilities that you can use for configuring the Verity K2 Server search engine, as well as creating, managing, and troubleshooting Verity collections.
ColdFusion Developer's Guide	Describes how to develop your dynamic web applications, including retrieving and updating your data, using structures, and forms.
CFML Reference	Provides descriptions, syntax, usage, and code examples for all ColdFusion tags, functions, and variables.

Viewing online documentation

All ColdFusion documentation is available online in HTML and Adobe PDF files. Go to the documentation home page for ColdFusion on the Adobe website: www.adobe.com/support/documentation/en/coldfusion/. Also, you can view the documentation in LiveDocs, which lets you add comments to pages and view the latest comments added by Adobe, by going to www.adobe.com/go/livedocs_cf8docs.

Chapter 2: Reserved Words and Variables

Adobe ColdFusion language includes reserved words and scope variables.

Contents

Reserved words

The following list indicates words you must not use for ColdFusion variables, user-defined function names, or custom tag names. Although you can safely use some of these words in some situations, you can prevent errors by avoiding them entirely.

- Any name starting with cf. However, when you call a CFML custom tag directly, you prefix the custom tag page name with cf_.
- Built-in function names, such as Now or Hash
- Scope names, such as Form or Session
- Operators, such as NE or IS
- The names of any built-in data structures, such as Error or File
- The names of any built-in variables, such as RecordCount or CGI variable names
- CFScript language element names such as for, default, or continue

Remember that ColdFusion is not case-sensitive. For example, all of the following are reserved words: IS, Is, iS, and is.

Reserved words in forms

You must also not create form field names that end in any of the following, except to specify a form field validation rule by using a hidden form field name.

- _integer
- _float
- _range
- _date
- _time
- _eurodate

Reserved words in queries

The following table lists SQL keywords that are reserved in ColdFusion queries of queries. This list includes all reserved words in the SQL standard, and should be avoided in variables used in all queries. Do not use these keywords as variable names in any queries.

Note: *Many database management systems have additional reserved words that you cannot use as variable names in queries to their databases. For a detailed list, see your DBMS documentation.*

ABSOLUTE	ACTION	ADD	ALL	ALLOCATE
ALTER	AND	ANY	ARE	AS
ASC	ASSERTION	AT	AUTHORIZATION	AVG
BEGIN	BETWEEN	BIT	BIT_LENGTH	BOTH
BY	CASCADE	CASCADED	CASE	CAST

CATALOG	CHAR	CHARACTER	CHARACTER_ LENGTH	CHAR_LENGTH
CHECK	CLOSE	COALESCE	COLLATE	COLLATION
COLUMN	COMMIT	CONNECT	CONNECTION	CONSTRAINT
CONSTRAINTS	CONTINUE	CONVERT	CORRESPONDING	COUNT
CREATE	CROSS	CURRENT	CURRENT_DATE	CURRENT_TIME
CURRENT_ TIMESTAMP	CURRENT_ USER	CURSOR	DATE	DAY
DEALLOCATE	DEC	DECIMAL	DECLARE	DEFAULT
DEFERRABLE	DEFERRED	DELETE	DESC	DESCRIBE
DESCRIPTOR	DIAGNOSTICS	DISCONNECT	DISTINCT	DOMAIN
DOUBLE	DROP	ELSE	END	END-EXEC
ESCAPE	EXCEPT	EXCEPTION	EXEC	EXECUTE
EXISTS	EXTERNAL	EXTRACT	FALSE	FETCH
FIRST	FLOAT	FOR	FOREIGN	FOUND
FROM	FULL	GET	GLOBAL	GO
GOTO	GRANT	GROUP	HAVING	HOUR
IDENTITY	IMMEDIATE	IN	INDICATOR	INITIALLY
INNER	INPUT	INSENSITIVE	INSERT	INT
INTEGER	INTERSECT	INTERVAL	INTO	IS
ISOLATION	JOIN	KEY	LANGUAGE	LAST
LEADING	LEFT	LEVEL	LIKE	LOCAL
LOWER	MATCH	MAX	MIN	MINUTE
MODULE	MONTH	NAMES	NATIONAL	NATURAL
NCHAR	NEXT	NO	NOT	NULL
NULLIF	NUMERIC	OCTET_LENGTH	OF	ON
ONLY	OPEN	OPTION	OR	ORDER
OUTER	OUTPUT	OVERLAPS	PAD	PARTIAL
POSITION	PRECISION	PREPARE	PRESERVE	PRIMARY
PRIOR	PRIVILEGES	PROCEDURE	PUBLIC	READ
REAL	REFERENCES	RELATIVE	RESTRICT	REVOKE
RIGHT	ROLLBACK	ROWS	SCHEMA	SCROLL
SECOND	SECTION	SELECT	SESSION	SESSION_USER
SET	SIZE	SMALLINT	SOME	SPACE
SQL	SQLCODE	SQLERROR	SQLSTATE	SUBSTRING

SUM	SYSTEM_USER	TABLE	TEMPORARY	THEN
TIME	TIMESTAMP	TIMEZONE_ HOUR	TIMEZONE_ MINUTE	TO
TRAILING	TRANSACTION	TRANSLATE	TRANSLATION	TRIM
TRUE	UNION	UNIQUE	UNKNOWN	UPDATE
UPPER	USAGE	USER	USING	VALUE
VALUES	VARCHAR	VARYING	VIEW	WHEN
WHENEVER	WHERE	WITH	WORK	WRITE
YEAR	ZONE			

Scope-specific built-in variables

ColdFusion returns variables, such as those returned in a `cfdirectory` or `cfftp` operation. A variable is usually referenced by *scoping* it according to its type: naming it according to the code context in which it is available; for example, Session.varname, or Application.varname. For more information on ColdFusion scopes, see "Using ColdFusion Variables" on page 24 in the *ColdFusion Developer's Guide*

You use the `cflock` tag to limit the scope of CFML constructs that modify shared data structures, files, and CFXs, to ensure that modifications occur sequentially. For more information, see "cflock" on page 366, and "Using Persistent Data and Locking" on page 273 in the *ColdFusion Developer's Guide*.

Variable scope

ColdFusion supports the Variables scope. Unscoped variables created with the `cfset` tag acquire the Variables scope by default. For example, the variable created by the statement `<CFSET linguist = Chomsky>` can be referenced as `#Variables.linguist#`.

Caller scope

History

ColdFusion MX: The Caller scope is accessible as a structure. (In earlier releases, it was not.)

CGI variables

see "CGI environment (CGI Scope) variables" on page 14

Client variables

The following client variables are reserved:

```
Client.CFID
Client.CFToken
Client.HitCount
Client.LastVisit
Client.TimeCreated
Client.URLToken
```

Server variables

Use the Server prefix to reference server variables, as follows:

```
Server.ColdFusion.ProductName
Server.ColdFusion.ProductVersion
Server.ColdFusion.ProductLevel
Server.ColdFusion.SerialNumber
Server.ColdFusion.SupportedLocales
Server.ColdFusion.AppServer
Server.ColdFusion.Expiration
Server.ColdFusion.RootDir
Server.OS.Name
Server.OS.AdditionalInformation
Server.OS.Version
Server.OS.BuildNumber
```

Application and session variables

To enable application and session variables, use the `cfapplication` tag or Application.cfc. Reference them as follows:

```
Application.myvariable
Session.myvariable
```

To ensure that modifications to shared data occur in the intended sequence, use the `cflock` tag. For more information, see "cflock" on page 366.

ColdFusion provides the following predefined application and session variables:

```
Application.ApplicationName
Session.CFID
Session.CFToken
Session.URLToken
```

Custom tag variables

A ColdFusion custom tag returns the following variables:

```
ThisTag.ExecutionMode
ThisTag.HasEndTag
ThisTag.GeneratedContent
ThisTag.AssocAttribs[index]
```

A custom tag can set a Caller variable to provide information to the caller. Set the Caller variable as follows:

```
<cfset Caller.variable_name = "value">
```

The calling page can access the variable with the `cfoutput` tag, as follows:

```
<cfoutput>#variable_name#</cfoutput>
```

Request variable

Request variables store data about the processing of one page request. Request variables store data in a structure that can be passed to nested tags, such as custom tags, and processed once.

To provide information to nested tags, set a Request variable, as follows:

```
<CFSET Request.field_name1 = "value">
<CFSET Request.field_name2 = "value">
<CFSET Request.field_name3 = "value">
...
```

Each nested tag can access the variable with the `cfoutput` tag, as follows:

```
<CFOUTPUT>#Request.field_name1#</CFOUTPUT>
```

Form variable

ColdFusion supports the Form variable FieldNames. FieldNames returns the names of the fields on a form. You use it on the action page associated with a form, as follows:

```
Form.FieldNames
```

ColdFusion tag-specific variables

Some ColdFusion tags return data as variables. For example, the `cffile` tag returns file size information in the FileSize variable, referenced as CFFILE.FileSize.

The following tags return data that you can reference in variables:

```
cfcatch
cfdirectory
cferror
cffile
cfftp
cfhttp
cfindex
cfldap
cfpop
cfquery
cfregistry
cfsearch
cfstoredproc
```

ColdFusion query variables

A ColdFusion tag that returns a query object supports the following variables, where *queryname* is the value of the `name` attribute:

```
queryname.CurrentRow
queryname.RecordCount
queryname.ColumnList
```

CFCATCH variables

Within a `cfcatch` block, the active exception properties can be accessed as the following variables:

```
CFCATCH.Type
CFCATCH.Message
CFCATCH.Detail
CFCATCH.ErrNumber
CFCATCH.NativeErrorCode
CFCATCH.SQLState
CFCATCH.LockName
CFCATCH.LockOperation
CFCATCH.MissingFileName
CFCATCH.TagContext
CFCATCH.ErrorCode
CFCATCH.ExtendedInfo
```

Within a `cfcatch` block, database exception properties can be accessed as the following variables:

```
CFCATCH.QueryError
CFCATCH.SQL
CFCATCH.Where
CFCATCH.Datasource
```

Within a `cfcatch` block, undefined variable exception properties can be accessed as the following variable:

```
CFCATCH.Name
```

Within a `cfcatch` block, syntax and parsing exception properties can be accessed as the following variables:

```
CFCATCH.TokenText
```

```
CFCATCH.Snippet
CFCATCH.Column
CFCATCH.KnownColumn
CFCATCH.Line
CFCATCH.KnownLine
```

CFDIRECTORY variables

The `cfdirectory` tag, with `action=list`, returns a query object as follows, where *queryname* is the `name` attribute value:

```
queryname.Name
queryname.Size
queryname.Type
queryname.DateLastModified
queryname.Attributes
queryname.Mode
```

CFERROR variables

When `cferror` generates an error page, the following error variables are available if `type="request"` or `"exception"`.

```
Error.Diagnostics
Error.MailTo
Error.DateTime
Error.Browser
Error.GeneratedContent
Error.RemoteAddress
Error.HTTPReferer
Error.Template
Error.QueryString
```

The following error variables are available if `type="validation"`.

```
Error.ValidationHeader
Error.InvalidFields
Error.ValidationFooter
```

Any `cfcatch` variable that applies to exception type can be accessed within the Error scope, as follows:

```
Error.Type
Error.Message
Error.Detail
Error.ErrNumber
Error.NativeErrorCode
Error.SQLState
Error.LockName
Error.LockOperation
Error.MissingFileName
Error.TagContext
Error.ErrorCode
Error.ExtendedInfo
```

Note: *You can substitute the prefix `CFERROR` for `Error`, if `type = "Exception"`; for example, `CFERROR.Diagnostics`, `CFERROR.Mailto`, or `CFERROR.DateTime`.*

CFFILE ACTION=Upload variables

File variables are read-only. Use the `CFFILE` prefix to reference file variables, for example, `CFFILE.ClientDirectory`. The `File` prefix is deprecated in favor of the `CFFILE` prefix.

```
CFFILE.AttemptedServerFile
CFFILE.ClientDirectory
CFFILE.ClientFile
CFFILE.ClientFileExt
CFFILE.ClientFileName
CFFILE.ContentSubType
CFFILE.ContentType
CFFILE.DateLastAccessed
CFFILE.FileExisted
CFFILE.FileSize
CFFILE.FileWasAppended
CFFILE.FileWasOverwritten
CFFILE.FileWasRenamed
CFFILE.FileWasSaved
CFFILE.OldFileSize
CFFILE.ServerDirectory
CFFILE.ServerFile
CFFILE.ServerFileExt
CFFILE.ServerFileName
CFFILE.TimeCreated
CFFILE.TimeLastModified
```

CFFTP error variables

When you use the `cfftp stoponerror` attribute, the following variables are populated:

```
CFFTP.Succeeded
CFFTP.ErrorCode
CFFTP.ErrorText
```

CFFTP ReturnValue variable

Some `cfftp` file and directory operations provide a return value, in the variable `CFFTP.ReturnValue`. Its value is determined by the results of the `action` attribute. When you specify any of the following actions, `cfftp` returns a value:

```
GetCurrentDir
GetCurrentURL
ExistsDir
ExistsFile
Exists
```

CFFTP query object columns

When you use the `cfftp` tag with the `listdir` action, `cfftp` returns a query object, where *`queryname`* is the `name` attribute value, and *`row`* is the row number of each file or directory entry:

```
queryname.Name[row]
queryname.Path[row]
queryname.URL[row]
queryname.Length[row]
queryname.LastModified[row]
queryname.Attributes
queryname.IsDirectory
queryname.Mode
```

CFHTTP variables

A `cfhttp get` operation can return text and binary files. Files are downloaded and the contents stored in a variable or file, depending on the MIME type, as follows:

```
CFHTTP.FileContent
CFHTTP.MimeType
CFHTTP.Header
CFHTTP.ResponseHeader[http_hd_key]
CFHTTP.StatusCode
```

CFLDAP variables

The `cfldap action=query` tag returns information about the LDAP query, as follows:

```
queryname.CurrentRow
queryname.RecordCount
queryname.ColumnList
```

CFPOP variables

The `cfpop` tag returns the following result columns, depending on the `action` attribute value and the use of other attributes, such as `attachmentpath`, where *queryname* is the `name` attribute value:

```
queryname.Date
queryname.From
queryname.Body
queryname.Header
queryname.MessageNumber
queryname.ReplyTo
queryname.Subject
queryname.CC
queryname.To
queryname.CurrentRow
queryname.RecordCount
queryname.ColumnList
queryname.Attachments
queryname.AttachmentFiles
```

CFQUERY and CFSTOREDPROC variables

The `cfquery` tag returns information about the query in this variable:

```
CFQUERY.ExecutionTime
```

The `cfquery` tag uses the query name to scope the following data about the query:

```
queryname.CurrentRow
queryname.RecordCount
queryname.ColumnList
```

The `cfstoredproc` tag returns the following variables:

```
CFSTOREDPROC.ExecutionTime
CFSTOREDPROC.StatusCode
```

CFREGISTRY variables

The `cfregistry` tag returns a query record set that you can reference after executing the `GetAll` action, as follows, where *queryname* is the `name` attribute value:

```
queryname.Entry
queryname.Type
queryname.Value
```

CFSEARCH variables

A `cfsearch` operation returns the following variables, where *searchname* is the `name` attribute value:

```
searchname.URL
searchname.Key
searchname.Title
searchname.Score
searchname.Custom1 and Custom2
searchname.Summary
searchname.RecordCount
searchname.CurrentRow
searchname.RecordsSearched
searchname.ColumnList
```

CGI environment (CGI Scope) variables

When a browser makes a request to a server, the web server and the browser create environment variables. In ColdFusion, these variables are referred to as *CGI environment variables*. CGI Environment variables contain data about the transaction between the browser and the server, such as the IP Address, browser type, and authenticated username. The available CGI variables depend on the browser and server software.

The CGI variables are available to ColdFusion pages in the CGI scope. They take the CGI prefix regardless of whether the server uses a server API or CGI to communicate with the ColdFusion server. You can reference CGI environment variables for a given page request anywhere in the page. CGI variables are read-only.

By default, when you use the `cfdump` tag to display the CGI scope, or when you request debug output of the CGI scope, ColdFusion attempts to display a fixed list of standard CGI environment variables. Because the available variables depend on the server, browser, and the types of interactions between the two, not all variables are normally available, and are represented by empty strings in the debug output. You can request any CGI variable in your application code, including variables that are not in the list variables displayed by dump and debug output.

ColdFusion checks for the following variables for the `cfdump` tag and debug output:

```
AUTH_PASSWORD
AUTH_TYPE
AUTH_USER
CERT_COOKIE
CERT_FLAGS
CERT_ISSUER
CERT_KEYSIZE
CERT_SECRETKEYSIZE
CERT_SERIALNUMBER
CERT_SERVER_ISSUER
CERT_SERVER_SUBJECT
CERT_SUBJECT
CF_TEMPLATE_PATH
CONTENT_LENGTH
CONTENT_TYPE
CONTEXT_PATH
GATEWAY_INTERFACE
HTTPS
HTTPS_KEYSIZE
HTTPS_SECRETKEYSIZE
HTTPS_SERVER_ISSUER
HTTPS_SERVER_SUBJECT
HTTP_ACCEPT
HTTP_ACCEPT_ENCODING
HTTP_ACCEPT_LANGUAGE
HTTP_CONNECTION
HTTP_COOKIE
HTTP_HOST
HTTP_REFERER
HTTP_USER_AGENT
QUERY_STRING
REMOTE_ADDR
REMOTE_HOST
REMOTE_USER
REQUEST_METHOD
SCRIPT_NAME
SERVER_NAME
SERVER_PORT
SERVER_PORT_SECURE
```

```
SERVER_PROTOCOL
SERVER_SOFTWARE
WEB_SERVER_API (This value is always blank; retained for compatibility.)
```

The following sections describe how to test for CGI environment variables and provide information on some of the more commonly used CGI environment variables

Testing for CGI variables

Because some browsers do not support some CGI variables, ColdFusion always returns `true` when it tests for the existence of a CGI variable, regardless of whether the browser supports the variable. To determine if the CGI variable is available, test for an empty string, as the following example shows:

```
<cfif CGI.varname IS NOT "">
    CGI variable exists
<cfelse>
    CGI variable does not exist
</cfif>
```

CGI server variables

The following table describes common CGI environment variables that the server creates (some of these are not available with some servers):

CGI server variable	Description
SERVER_SOFTWARE	Name and version of the information server software answering the request (and running the gateway). Format: name/version.
SERVER_NAME	Server's hostname, DNS alias, or IP address as it appears in self-referencing URLs.
GATEWAY_INTERFACE	CGI specification revision with which this server complies. Format: CGI/revision.
SERVER_PROTOCOL	Name and revision of the information protocol this request came in with. Format: protocol/revision.
SERVER_PORT	Port number to which the request was sent.
REQUEST_METHOD	Method with which the request was made. For HTTP, this is Get, Head, Post, and so on.
PATH_INFO	Extra path information, as given by the client. Scripts can be accessed by their virtual pathname, followed by extra information at the end of this path. The extra information is sent as PATH_INFO.
PATH_TRANSLATED	Translated version of PATH_INFO after any virtual-to-physical mapping.
SCRIPT_NAME	Virtual path to the script that is executing; used for self-referencing URLs.
QUERY_STRING	Query information that follows the ? in the URL that referenced this script.
REMOTE_HOST	Hostname making the request. If the server does not have this information, it sets REMOTE_ADDR and does not set REMOTE_HOST.
REMOTE_ADDR	IP address of the remote host making the request.
AUTH_TYPE	If the server supports user authentication, and the script is protected, the protocol-specific authentication method used to validate the user.
REMOTE_USER AUTH_USER	If the server supports user authentication, and the script is protected, the username the user has authenticated as. (Also available as AUTH_USER.)

CGI server variable	Description
REMOTE_IDENT	If the HTTP server supports RFC 931 identification, this variable is set to the remote username retrieved from the server. Use this variable for logging only.
CONTENT_TYPE	For queries that have attached information, such as HTTP POST and PUT, this is the content type of the data.
CONTENT_LENGTH	Length of the content as given by the client.

CGI client variables

The following table describes common CGI environment variables the browser creates and passes in the request header:

CGI client variable	Description
HTTP_REFERER	The referring document that linked to or submitted form data.
HTTP_USER_AGENT	The browser that the client is currently using to send the request. Format: software/version library/version.
HTTP_IF_MODIFIED_SINCE	The last time the page was modified. The browser determines whether to set this variable, usually in response to the server having sent the LAST_MODIFIED HTTP header. It can be used to take advantage of browser-side caching.

CGI client certificate variables

ColdFusion makes available the following client certificate data. These variables are available when running Microsoft IIS 4.0 or Netscape Enterprise under SSL if your web server is configured to accept client certificates.

CGI client certificate variable	Description
CERT_SUBJECT	Client-specific information provided by the web server. This data typically includes the client's name, e-mail address, and so on, for example: `O = "VeriSign, Inc.", OU = VeriSign Trust Network, OU = "www.verisign.com/repository/RPA Incorp. by Ref.,LIAB.LTD(c)98", OU = Persona Not Validated, OU = Digital ID Class 1 - Microsoft, CN = Matthew Lund, E = mlund@.com`
CERT_ISSUER	Information about the authority that provided the client certificate, for example: `O = "VeriSign, Inc.", OU = VeriSign Trust Network, OU = "www.verisign.com/repository/RPA Incorp. By Ref.,LIAB.LTD(c)98", CN = VeriSign Class 1 CA Individual Subscriber-Persona Not Validated`

Chapter 3: ColdFusion Tags

ColdFusion Markup Language (CFML) includes a set of tags that you use in ColdFusion pages to interact with data sources, manipulate data, and display output. CFML tag syntax is similar to HTML element syntax.

The following categorized and alphabetical lists of the tags are followed by the detailed tag descriptions.

Contents

Tag summary

The following table briefly describes CFML tags:

CFML tag	Category	Description
cfabort	Flow-control tags	Stops the processing of a ColdFusion page at the tag location
cfajaximport	Internet protocol tags	Controls importation of JavaScript files used for ColdFusion AJAX-based features
cfajaxproxy	Internet protocol tags	Generates an AJAX proxy class on the client page for a ColdFusion component
cfapplet	Forms tags	Embeds Java applets in a cfform tag
cfapplication	Application framework tags	Defines an application name; activates client variables; specifies client variable storage mechanism
cfargument	Extensibility tags	Creates a parameter definition within a component definition; defines a function argument
cfassociate	Application framework tags	Enables subtag data to be saved with a base tag
cfbreak	Flow-control tags	Breaks out of a CFML looping construct
cfcache	Page processing tags	Caches ColdFusion pages
cfcalendar	Forms tags	Provides a calendar from which to select a date
cfcase	Flow-control tags	Used with the cfswitch and cfdefaultcase tags
cfcatch	Exception handling tags, Flow-control tags	Catches exceptions in ColdFusion pages
cfchart	Data output tags	Generates and displays a chart
cfchartdata	Data output tags	Defines chart data points
cfchartseries	Data output tags	Defines style in which chart data displays
cfcol	Data output tags	Defines table column header, properties
cfcollection	Extensibility tags	Administers Verity collections

CFML tag	Category	Description
cfcomponent	Extensibility tags	Creates and defines a component object
cfcontent	Data output tags, Page processing tags	Defines content type and filename of a file to be downloaded by the current page
cfcookie	Variable manipulation tags	Defines and sets cookie variables, including expiration and security options
cfdbinfo	Database manipulation tags	Lets you retrieve information about a data source
cfdefaultcase	Flow-control tags	Receives control if there is no matching cfcase tag value
cfdirectory	File management tags	Performs typical directory-handling tasks from within a ColdFusion application
cfdiv	Display management tags	Creates an HTML tag with that is populated using a bind expressions.
cfdocument	Data output tags	Creates PDF or Adobe® FlashPaper® output from a text block that contains CFML and HTML
cfdocumentitem	Data output tags	Specifies action items, such as header, footer, and page break, for a PDF or FlashPaper document
cfdocumentsection	Data output tags	Divides a PDF or FlashPaper document into sections
cfdump	Debugging tags, Variable manipulation tags	Outputs variables for debugging
cfelse	Flow-control tags	Creates IF-THEN-ELSE constructs
cfelseif	Flow-control tags	Creates IF-THEN-ELSE constructs
cferror	Exception handling tags, Application framework tags	Displays custom HTML error pages when errors occur
cfexchangecalendar	Communications tags	Gets, creates, deletes, modifies, or responds to Microsoft Exchange calendar events
cfexchangeconnection	Communications tags	Opens or closes a persistent connection with an Exchange server
cfexchangecontact	Communications tags	Gets, creates, deletes, or modifies Exchange contacts
cfexchangefilter	Communications tags	Sets filter conditions used in Exchange tag get operations
cfexchangemail	Communications tags	Gets and deletes Exchange mail messages and sets message properties
cfexchangetask	Communications tags	Gets, creates, deletes, or modifies an Exchange user task
cfexecute	Flow-control tags, Extensibility tags	Executes developer-specified process on server computer
cfexit	Flow-control tags	Aborts processing of executing CFML tag
cffeed	Communications tags, Internet protocol tags	Reads, creates, and converts, Atom and RSS syndication feeds
cffile	File management tags	Performs typical file-handling tasks from within ColdFusion application
cfflush	Data output tags, Page processing tags	Flushes currently available data to client
cfform	Forms tags	Builds input form; performs client-side input validation

CFML tag	Category	Description
cfformgroup	Forms tags	Groups form control into a containing object
cfformitem	Forms tags	Adds text and dividing rules to Adobe® Flash® forms
cfftp	Forms tags, Extensibility tags, Internet protocol tags	Permits FTP file operations
cffunction	Extensibility tags	Defines function that you build in CFML
cfgrid	Forms tags	Displays tabular grid control, in cfform tag
cfgridcolumn	Forms tags	Used in cfform; defines columns in a cfgrid
cfgridrow	Forms tags	Defines a grid row; used with cfgrid
cfgridupdate	Forms tags	Directly updates ODBC data source from edited grid data
cfheader	Data output tags, Page processing tags	Generates HTTP headers
cfhtmlhead	Page processing tags	Writes text and HTML to HEAD section of page
cfhttp	Internet protocol tags	Performs GET and POST to upload file or post form, cookie, query, or CGI variable directly to server
cfhttpparam	Internet protocol tags	Specifies parameters required for a cfhttp POST operation; used with cfhttp
cfif	Flow-control tags	Creates IF-THEN-ELSE constructs
cfimage	Other tags	Creates a cfimage, a ColdFusion data type that can be operated by image functions.
cfimport	Application framework tags	Imports JSP tag libraries into a CFML page
cfinclude	Flow-control tags	Embeds references to ColdFusion pages
cfindex	Extensibility tags	Creates Verity search indexes
cfinput	Forms tags	Creates an input element (radio button, check box, text entry box); used in cfform
cfinsert	Database manipulation tags	Inserts records in a data source
cfinterface	Application framework tags, Extensibility tags	Defines an interface that a ColdFusion component can implement
cfinvoke	Extensibility tags	Invokes component methods from a ColdFusion page or component
cfinvokeargument	Extensibility tags	Passes a parameter to a component method or a web service
cflayout	Display management tags	Creates a region of its container with a specific layout behavior
cflayoutarea	Display management tags	Defines a display region within a cflayout tag body
cfldap	Internet protocol tags	Provides access to LDAP directory servers
cflocation	Flow-control tags	Controls execution of a page
cflock	Application framework tags	Ensures data integrity and synchronizes execution of CFML code

CFML tag	Category	Description
cflog	Data output tags, Other tags	Writes a message to a log file
cflogin	Security tags	Defines a container for user login and authentication code
cfloginuser	Security tags	Identifies an authenticated user to ColdFusion
cflogout	Security tags	Logs the current user out
cfloop	Flow-control tags	Repeats a set of instructions based on conditions
cfmail	Communications tags, Internet protocol tags	Assembles and posts an e-mail message
cfmailparam	Communications tags, Internet protocol tags	Attaches a file or adds a header to an e-mail message
cfmailpart	Communications tags, Internet protocol tags	Contains one part of a multipart mail message
cfmenu	Display management tags	Creates a top-level menu or a tool bar.
cfmenuitem	Display management tags	Defines an entry in a menu, including an item that is the head of a submenu.
cfmodule	Application framework tags	Invokes a custom tag for use in a ColdFusion page
cfNTauthenticate	Security tags	Authenticates user information against an NT domain
cfobject	Extensibility tags	Creates COM, component, CORBA, Java, and web service objects
cfobjectcache	Database manipulation tags	Flushes the query cache
cfoutput	Data output tags	Displays the output of a database query or other operation
cfparam	Variable manipulation tags	Defines a parameter and its default value
cfpdf	Forms tags	Manipulates existing PDF documents.
Usagecfpdfform	Forms tags	Creates and manipulates PDF forms.
cfpdfformparam	Forms tags	Creates interactive fields on a PDF form.
cfpdfparam	Forms tags	Child tag of the cfpdf tag. Used only with the merge action to merge multiple pages or PDF documents into one file
cfpdfsubform	Forms tags	Creates subforms within a PDF form.
cfpod	Display management tags	Creates a an area of the browser or layout area with an optional title bar and a body
cfpop	Communications tags, Internet protocol tags	Gets and deletes messages from POP mail server
cfpresentation	Data output tags	Creates a presentation dynamically from an HTML page or SWF files
cfpresentationslide	Data output tags	Creates a slide dynamically from an HTML page or SWF source files (child tag of the cfpresentation tag)
cfpresenter	Data output tags	Describes a presenter in a slide presentation
cfprint	Data output tags	Prints PDF documents. Used for automated print jobs
cfprocessingdirective	Data output tags	Suppresses white space and other output

CFML tag	Category	Description
cfprocparam	Database manipulation tags	Holds parameter information for stored procedure
cfprocresult	Database manipulation tags	Result set name that ColdFusion tags use to access result set of a stored procedure
cfproperty	Extensibility tags	Defines components
cfquery	Database manipulation tags	Passes SQL statements to a database
cfqueryparam	Database manipulation tags	Checks data type of a query parameter
cfregistry	Other tags, Variable manipulation tags	Reads, writes, and deletes keys and values in a Windows system registry
cfreport	Exception handling tags	Embeds a ColdFusion Report Builder or Crystal Reports report
cfreportparam	Exception handling tags	Passes an input parameter to a ColdFusion Report Builder report
cfrethrow	Exception handling tags	Rethrows currently active exception
cfreturn	Extensibility tags	Returns results from a component method
cfsavecontent	Variable manipulation tags	Saves generated content inside tag body in a variable
cfschedule	Variable manipulation tags	Schedules page execution; optionally, produces static pages
cfscript	Application framework tags	Encloses a set of cfscript statements
cfsearch	Extensibility tags	Executes searches against data indexed in Verity collections, using cfindex
cfselect	Forms tags	Creates a drop-down list box form element; used in cfform tag
cfset	Variable manipulation tags	Defines a variable
cfsetting	Other tags, Variable manipulation tags	Defines and controls ColdFusion settings
cfsilent	Data output tags, Page processing tags	Suppresses CFML output within tag scope
cfslider	Forms tags	Creates slider control; used in cfform
cfsprydataset	Internet protocol tags	Creates a spry data set
cfstoredproc	Database manipulation tags	Holds database connection information; identifies a stored procedure to execute
cfswitch	Flow-control tags	Evaluates passed expression; passes control to matching cfcase tag
cftable	Data output tags	Builds a table in a ColdFusion page
cftextarea	Forms tags	Puts a multiline text box in a form
cfthread	Application framework tags	Creates and manages ColdFusion threads, independent streams of execution.

CFML tag	Category	Description
cfthrow	Exception handling tags, Flow-control tags	Throws a developer-specified exception
cftimer	Debugging tags	Displays execution time for a block of code
cftooltip	Display management tags	Specifies text to display when the mouse pointer hovers over the tag body elements
cftrace	Debugging tags	Displays and logs application debugging data
cftransaction	Database manipulation tags	Groups cfquery operations into one transaction; performs rollback processing
cftree	Forms tags	Creates tree control element; used in cfform
cftreeitem	Forms tags	Populates a tree control element in a form; used with cftree
cftry	Exception handling tags, Flow-control tags	Catches exceptions in ColdFusion pages
cfupdate	Database manipulation tags	Updates rows in a database data source
cfwddx	Extensibility tags	Serializes and deserializes CFML data structures to XML-based WDDX format
cfwindow	Display management tags	Creates a pop-up window in the browser
cfxml	Extensibility tags	Creates an XML document object
cfzip	File management tags	Manipulates ZIP and JAR files
cfzipparam	File management tags	Manipulates ZIP and JAR files

Tags by function

The following tables list tags by their function or purpose.

Application framework tags

cfapplication	cfimport	cfmodule
cfassociate	cfinterface	cfscript
cferror	cflock	cfthread

Communications tags

cfexchangecalendar	cfexchangemail	cfmailparam
cfexchangeconnection	cfexchangetask	cfmailpart
cfexchangecontact	cffeed	cfpop
cfexchangefilter	cfmail	

Database manipulation tags

cfdbinfo	cfprocresult	cftransaction
cfinsert	cfquery	cfupdate
cfobjectcache	cfqueryparam	
cfprocparam	cfstoredproc	

Data output tags

cfchart
cfchartdata
cfchartseries
cfcol
cfcontent
cfdocument
cfdocumentitem
cfdocumentsection
cfflush
cfheader
cflog
cfoutput
cfpresentation
cfpresentationslide
cfpresenter
cfprocessingdirective
cfprint
cfreport
cfreportparam
cfsilent
cftable

Debugging tags

cfdump
cftimer
cftrace

Display management tags

cfdiv
cflayout
cflayoutarea
cfmenu
cfmenuitem
cfpod
cftooltip
cfwindow

Exception handling tags

cfcatch
cferror
cfrethrow
cfthrow
cftry

Extensibility tags

cfchart
cfchartdata
cfchartseries
cfcollection
cfcomponent
cfexecute
cfftp
cffunction
cfindex
cfinterface
cfinvoke
cfinvokeargument
cfobject
cfproperty
cfreport
cfreportparam
cfreturn
cfsearch
cfwddx
cfxml

File management tags

cfdirectory
cffile
cfftp
cfzip
cfzipparam

Flow-control tags

cfabort
cfbreak
cfcase
cfdefaultcase
cfelse
cfelseif
cfexecute
cfexit
cfif
cfinclude
cflocation
cfloop
cfrethrow
cfswitch
cfthrow
cftry

Forms tags

cfapplet
cfcalendar
cfform
cfformgroup
cfformitem
cfgrid
cfgridcolumn
cfgridrow
cfgridupdate
cfinput
cfpdf
Usagecfpdfform
cfpdfformparam
cfpdfparam
cfpdfsubform
cfselect
cfslider
cftextarea
cftree
cftreeitem

Internet protocol tags

cfajaximport
cfajaxproxy
cfftp
cffeed
cfhttp
cfhttpparam
cfldap
cfmail
cfmailparam
cfmailpart
cfpop
cfsprydataset

Page processing tags

cfcache
cfcontent
cfflush
cfheader
cfhtmlhead
cfinclude
cfprocessingdirective
cfsetting
cfsilent

Security tags

cflogin
cfloginuser
cflogout
cfNTauthenticate

Variable manipulation tags

cfcookie
cfdump
cfparam
cfregistry
cfsavecontent
cfschedule
cfset
cfsetting

Other tags

cfimage
cflog
cfregistry

Tag changes since ColdFusion 5

The following tables list tags, attributes, and values that have changed since ColdFusion 5, and indicate the specific release in which the change was made.

New tags, attributes, and values

This table lists tags, attributes, and attribute values that have been added since the ColdFusion MX release:

Tag	Attribute or value	Added in this ColdFusion release
Multiple tags	`attributeCollection`	ColdFusion 8
`cfajaximport`	All	ColdFusion 8
`cfajaxproxy`	All	ColdFusion 8
`cfapplication`	scriptProtect	ColdFusion MX 7
	loginStorage	ColdFusion MX 6.1
`cfargument`	`component` value of `type` attribute	ColdFusion 8
	`xml` value of `type` attribute	ColdFusion MX 7
	All	ColdFusion MX
`cfcache`	`cachedirectory`, `timespan` attributes	ColdFusion MX
`cfcalendar`	`onBlur` and `onFocus` attributes	ColdFusion MX 7.01
	All	ColdFusion MX 7
`cfchart`	`style`,`title` attributes	ColdFusion MX 7
	`xAxisType`, `yAxisType` attributes	ColdFusion MX 6.1
	All	ColdFusion MX
`cfchartdata`	All	ColdFusion MX
`cfchartseries`	`datalabelstyle` attribute	ColdFusion MX 7
	`horizontalbar` value of `type` attribute	
	All	ColdFusion MX
`cfcollection`	`categories` attribute	ColdFusion MX 7
	New values of the `language` attribute	
	`list` and `categoryList` values of `action` attribute	
	`name` attribute	ColdFusion MX

Tag	Attribute or value	Added in this ColdFusion release
cfcomponent	implements, serviceaddress attributes	ColdFusion 8
	component value of extends attribute	
	style, namespace, serviceportname, porttypename, wsdlfile, bindingname, and output attributes	ColdFusion MX 7
	Extended functionality for the hint and displayname attributes when publishing document-literal style web services	
	All	ColdFusion MX
cfcontent	variable attribute	ColdFusion MX 7
cfdbinfo	All	ColdFusion 8
cfdirectory	listinfo and type attributes	ColdFusion 8
	recurse attribute for list and delete actions	ColdFusion MX 7
cfdiv	All	ColdFusion 8
cfdocument	bookmark, authPassword, authUser, localUrl, proxyHost, proxyPassword, proxyPort, proxyUser, saveAsName and userAgent attributes totalsectionpagecount and currentsectionpagenumber scope variables	ColdFusion 8
	src, srcfile, and mimetype attributes	ColdFusion MX 7.01
	All	ColdFusion MX 7
cfdocumentitem	All	ColdFusion MX 7
cfdocumentsection	name, authPassword, authUser, and userAgent attributes	ColdFusion 8
	All	ColdFusion MX 7
cfdump	show, format, hide, keys, metainfo, output, and showUDFs attributes	ColdFusion 8
cfexchangecalendar	All	ColdFusion 8
cfexchangeconnection	All	ColdFusion 8
cfexchangecontact	All	ColdFusion 8
cfexchangefilter	All	ColdFusion 8
cfexchangemail	All	ColdFusion 8
cfexchangetask	All	ColdFusion 8
cfexecute	variable attribute	ColdFusion MX 6.1
cffeed	All	ColdFusion 8
cffile	result attribute for action="upload" action	ColdFusion MX 7
	fixnewline attribute for action="append" and action="write" actions	

Tag	Attribute or value	Added in this ColdFusion release
cfform	onSuccess attribute support in AJAX controls for the onError attribute	ColdFusion 8
	name and action attributes are optional	ColdFusion MX 7
	accessible, format, height, width , method, onError, onReset, preloader, scriptsrc, skin, style, timeout, wMode attributes	
cfformgroup	All	ColdFusion MX 7
cfformitem	script value of type attribute	ColdFusion MX 7.01
	All	ColdFusion MX 7
cfftp	fingerprint, key, paraphrase, and secure attributes quote, site, allo, and acct values to the action attribute	ColdFusion 8
	result attribute	ColdFusion MX 7
cffunction	description attribute; the XML value to the returntype attribute	ColdFusion MX 7
	All	ColdFusion MX
cfgrid	bind, bindOnLoad, pageSize, preservePageOnSort, stripeRows, stripeRowColor attributes, and HTML value of format attribute.	ColdFusion 8
	onBlur and onFocus attributes	ColdFusion MX 7.01
	format attribute and support for Flash and XML output enabled, onChange, style, tooltip, visible attributes (Flash format only)	ColdFusion MX 7
cfgridcolumn	mask attribute	ColdFusion MX 7
	currency value of type attribute	ColdFusion MX 7
cfhttp	clientCert and clientCertPassword attributes	ColdFusion 8
	never value of GetAsBinary attribute	ColdFusion MX 7.01
	result attribute	ColdFusion MX 7
	HEAD, PUT, DELETE, OPTIONS, and TRACE values of method attribute	ColdFusion MX 6.1
	multipart, getasbinary, proxyUser, proxyPassword attributes	
	charset,firstrowasheaders attributes	ColdFusion MX
cfhttpparam	header and body values of type attribute	ColdFusion MX 6.1
	encoded, mimeType attributes	
cfimage	All	ColdFusion 8
cfimport	All	ColdFusion MX

Tag	Attribute or value	Added in this ColdFusion release
cfindex	prefix attribute	ColdFusion MX 7.01
	custom3, custom4, category, and categorytree attributes for update and refresh actions	ColdFusion MX 7
	status attribute for update, refresh, delete, and purge actions	
	New values of the language attribute	
cfinput	autosuggest, autosuggestBindDelay, autosuggestMinLength, delimiter, maxResultsDisplayed, showAutosuggestLoadingIcon, sourceForTooltip, and typeahead attributes.	ColdFusion 8
	support for the bind attribute in HTML forms and the bindAttribute, bindOnload, and onBindError attributes.	
	datefield value of the type attribute in HTML forms	
	height and width attributes (all except checkbox and radiobutton)	ColdFusion MX 7
	bind attribute (text and password)	
	label attribute (all but button, image, reset, and submit)	
	mask attribute (text only)	
	validateAt attribute (all but button, image, reset, and submit)	
	datefield, button, file, hidden, image, reset, and submit values of type attribute	
	daynames and monthnames attributes (type="datefield" only)	
	boolean, email, guid, maxlength, noblanks, range, submitOnce, URL, USdate, and uuid values of the validate attribute	
	tooltip, visible, and enabled attributes (Flash forms only)	
	src attribute (image only)	
cfinterface	All	ColdFusion 8
cfinvoke	refreshWSDL, wsdl2java arguments	ColdFusion 8
	servicePort attribute for web services	ColdFusion MX 7
	All	ColdFusion MX
cfinvokeargument	omit attribute	ColdFusion MX 7
	All	ColdFusion MX
cflayout	All	ColdFusion 8

Tag	Attribute or value	Added in this ColdFusion release
cflayoutarea	All	ColdFusion 8
cfldap	returnAsBinary attribute	ColdFusion MX 7
cflocation	statusCode attribute	ColdFusion 8
cflock	Request value of scope attribute	ColdFusion 8
cflogin	All	ColdFusion MX
cfloginuser	All	ColdFusion MX
cflogout	All	ColdFusion MX
cfloop	characters, file, and array attributes	ColdFusion 8
cfmail	prioirity, useSSL, and useTLS	ColdFusion 8
	The cfmail tag no longer lets you send multipart mail by embedding the entire MIME-encoded message in the tag body. Use the cfmailpart tag, instead.	ColdFusion MX 7
	charset, failto, replyTo, userName, password, wrapText attributes	ColdFusion MX 6.1
	spoolEnable attribute	ColdFusion MX
cfmailparam	contentID, dispositionattributes	ColdFusion MX 7
	type attribute	ColdFusion MX 6.1
cfmailpart	All	ColdFusion MX 6.1
cfmenu	All	ColdFusion 8
cfmenuitem	All	ColdFusion 8
cfNTauthenticate	All	ColdFusion MX 7
cfobject	.net value of type attribute and related assembly, port, protocol, and secure attributes	ColdFusion 8
	password, proxyPassword, proxyPort, proxyServer, proxyUser, refreshWSDL, userName, wsdl2JavaArgs, and wsportname attributes for web services	
	component and webservice attributes	ColdFusion MX
cfobjectcache	All	ColdFusion MX
cfparam	min, max, pattern attributes	ColdFusion MX 7
	creditcard, email, eurodate, float, integer, range, regex, regular_expression, ssn, social_security_number, time, URL, USdate, XML, zipcode values of the type attribute	
cfpdf	All	ColdFusion 8
Usagecfpdfform	All	ColdFusion 8
cfpdfformparam	All	ColdFusion 8
cfpdfparam	All	ColdFusion 8

Tag	Attribute or value	Added in this ColdFusion release
cfpdfsubform	All	ColdFusion 8
cfpod	All	ColdFusion 8
cfpop	cids query variable	ColdFusion MX 7.01
cfpresentation	All	ColdFusion 8
cfpresentationslide	All	ColdFusion 8
cfpresenter	All	ColdFusion 8
cfprint	All	ColdFusion 8
cfprocessingdirective	pageEncoding attribute	ColdFusion MX
cfproperty	All	ColdFusion MX
cfquery	result attribute	ColdFusion MX 7
cfreturn	All	ColdFusion MX
cfreport	HTML, XML values of format attribute, resourceTimespan, style attributes	ColdFusion 8
	RTF value of format attribute	ColdFusion MX 7.01
	template, format, name, filename, query, overwrite attribute	ColdFusion MX 7
cfreportparam	chart, query, series, style, subreport attributes	ColdFusion 8
	name, value attributes	ColdFusion MX 7
cfsearch	category, categoryTree, status, suggestions, contextPassages, contextBytes, contextHighlightBegin, contextHighlightEnd, previousCriteria attributes	ColdFusion MX 7
	natural, internet, and internet_basic values of type attribute	
cfselect	support for the bind attribute in HTML forms and the bindAttribute, bindOnload, and onBindError attributes.	ColdFusion 8
	Support for tooltips in HTML forms including the sourceForTooltip attribute	
	selected attribute can take a list	ColdFusion MX 7
	enabled, group, height, label, onKeyUp, onKeyDown, onMouseUp, onMouseDown, onChange, onClick, queryPosition, tooltip, visible, and width attributes	
cfsetting	requestTimeOut attribute	ColdFusion MX
cfsprydataset	All	ColdFusion 8
cfstoredproc	result attribute	ColdFusion MX 7

Tag	Attribute or value	Added in this ColdFusion release
cftextarea	Rich text editor support including the following attributes (HTML format only): richtext, basepath, fontFormats, fontNames, fontSizes, skin, stylesXML, templatesXML, toolbar, toolbarOnFocus, and support for the height and width attributes in HTML format	ColdFusion 8
	support for the bind attribute and bindAttribute, bindOnLoad, and onBindError attributes in HTML format	
	support for tooltips in HTML format including tooltip and sourceForToolTip attribute	
	html attribute	ColdFusion MX 7.01
	All	ColdFusion MX 7
cfthread	All	ColdFusion 8
cfthrow	object attribute	ColdFusion MX
cftimer	All	ColdFusion MX 7
cftooltip	All	ColdFusion 8
cftree	onBlur and onFocus attributes	ColdFusion MX 7.01
	format, onChange, style attributes	ColdFusion MX 7
cftrace	All	ColdFusion MX
cfwindow	ALL	ColdFusion 8
cfxml	All	ColdFusion MX
cfzip	All	ColdFusion 8
cfzipparam	All	ColdFusion 8

Deprecated tags, attributes, and values

The following tags, attributes, and attribute values are deprecated. Do not use them in ColdFusion applications. They might not work, and might cause an error, in releases later than ColdFusion MX.

Tag	Attribute or value	Deprecated as of this ColdFusion release
cfcache	cachedirectory, timeout attributes	ColdFusion MX
cfcollection	map and repair options of the action attribute	ColdFusion MX 7
cferror	monitor option of the exception attribute	ColdFusion MX
cffile	system value for attributes attribute	ColdFusion MX
	temporary value for attributes attribute	ColdFusion MX
cfform	passthrough attribute	ColdFusion MX 7
	enableCAB attribute	ColdFusion MX
cfftp	agentname attribute	ColdFusion MX

Tag	Attribute or value	Deprecated as of this ColdFusion release
cfgraph	All	ColdFusion MX
cfgraphdata	All	ColdFusion MX
cfgridupdate	connectString, dbName, dbServer, dbType, provider, providerDSN attributes	ColdFusion MX
cfinput	passthrough attribute	ColdFusion MX 7
cfinsert	connectString, dbName, dbServer, dbType, provider, providerDSN attributes	ColdFusion MX
cfldap	filterFile attribute	ColdFusion MX
cflog	date, thread, time attributes	ColdFusion MX
cfquery	connectString, dbName, dbServer, provider, providerDSN, sql attributes	ColdFusion MX
	The following dbType attribute values: dynamic, ODBC, Oracle73, Oracle80, Sybase11, OLEDB, DB2	ColdFusion MX (The value query is valid.)
cfregistry	All, on UNIX only	ColdFusion MX
cfsearch	external, language attributes	ColdFusion MX
cfselect	passthrough attribute	ColdFusion MX 7
cfservlet	All	ColdFusion MX
cfservletparam	All	ColdFusion MX
cfslider	img, imgStyle, grooveColor, refreshLabel, tickmarkimages, tickmarklabels, tickmarkmajor, tickmarkminor attributes	ColdFusion MX
cfstoredproc	connectString, dbName, dbServer, dbtype, provider, providerDSN attributes	ColdFusion MX
cftextinput	All	ColdFusion MX 7
cfupdate	connectString, dbName, dbServer, dbtype, provider, providerDSN attributes	ColdFusion MX

Obsolete tags, attributes, and values

The following tags, attributes, and attribute values are obsolete. Do not use them in ColdFusion applications. They do not work, and might cause an error, in releases later than ColdFusion 5.

Tag	Attribute or value	Obsolete as of this ColdFusion release
cfauthenticate	All	ColdFusion MX
cfchart	rotated attribute	ColdFusion MX 7
cffile	attributes attribute value archive	ColdFusion MX
cfimpersonate	All	ColdFusion MX

Tag	Attribute or value	Obsolete as of this ColdFusion release
`cfindex`	`action` attribute value `optimize`	ColdFusion MX
	`external` attribute	
`cfinternaladminsecurity`	All	ColdFusion MX This tag did not appear in *CFML Reference*.
`cfldap`	`filterConfig` and `filterFile` attributes	ColdFusion MX
`cfnewinternaladminsecurity`	All	ColdFusion MX This tag did not appear in *CFML Reference*.
`cfsetting`	`catchExceptionsByPattern` attribute	ColdFusion MX

cfabort

Description

Stops the processing of a ColdFusion page at the tag location. ColdFusion returns everything that was processed before the tag. The tag is often used with conditional logic to stop processing a page when a condition occurs.

Category

Flow-control tags

Syntax

```
<cfabort
    showError = "error message">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *whose value is a structure. Specify the structure name in the* `attributeCollection` *and use the tag's attribute names as structure keys.*

See also

`cfbreak`, `cfexecute`, `cfexit`, `cfif`, `cflocation`, `cfloop`, `cfswitch`, `cfthrow`, `cftry`; "cfabort and cfexit" on page 20 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`showError`	Optional		Error to display, in a standard ColdFusion error page, when tag executes.

Usage

When you use the `cfabort` and `cferror` tags together, the `cfabort` tag halts processing immediately; the `cferror` tag redirects output to a specified page.

If this tag does not contain a `showError` attribute value, processing stops when the tag is reached and ColdFusion returns the page contents up to the line that contains the `cfabort` tag.

When you use this tag with the `showError` attribute, but do not define an error page using `cferror`, page processing stops when the `cfabort` tag is reached. The message in `showError` displays to the client.

When you use this tag with the `showError` attribute and an error page using `cferror`, ColdFusion redirects output to the error page specified in the `cferror` tag.

Example

This example shows the use of `cfabort` to stop processing. In the second example, where cfabort is used, the result never displays.

```
<h3>Example A: Let the instruction complete itself</h3>
<!--- first, set a variable --->
<cfset myVariable = 3>
<!--- now, perform a loop that increments this value --->
<cfloop from = "1" to = "4" index = "Counter">
    <cfset myVariable = myVariable + 1>
</cfloop>

<cfoutput>
<p>The value of myVariable after incrementing through the loop #Counter# times is:
    #myVariable#</p>
</cfoutput>
```

```
<h3>Example B: Use cfabort to halt the instructions with showmessage attribute and
    cferror</h3>
<!--- Reset the variable and show the use of cfabort. --->
<cfset myVariable = 3>
<!--- Now, perform a loop that increments this value. --->
<cfloop from = "1" to = "4" index = "Counter">
<!--- On the second time through the loop, cfabort. --->
    <cfif Counter is 2>
    <!--- Take out the cferror line to see cfabort error processed by CF error page. --->
        <cferror type="request" template="request_err.cfm">
        <cfabort showerror="CFABORT has been called for no good reason">
<!--- Processing is stopped, --->
<!--- and subsequent operations are not carried out.--->
    <cfelse>
        <cfset myVariable = myVariable + 1>
    </cfif>
</cfloop>

<cfoutput>
<p> The value of myVariable after incrementing through the loop#counter# times is:
#myVariable#</p>
</cfoutput>
```

cfajaximport

Description

Controls the JavaScript files that are imported for use on pages that use ColdFusion AJAX tags and features.

Category

Internet protocol tags

Syntax

```
<cfajaximport
    cssSrc = "local URL path"
    scriptSrc = "local URL path"
    tags = "comma-delimited list">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *whose value is a structure. Specify the structure name in the* `attributeCollection` *and use the tag's attribute names as structure keys.*

See also

`cfform`, `cfgrid`, `cfinput`, `cflayout`, `cfmenu`, `cfpod`, `cfsprydataset`, `cftextarea`, `cftooltip`, `cftree`, `cfwindow`, "Specifying client-side support files" on page 667 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`cssSrc`	Optional	*scriptsrc*/ajax	Specifies the URL, relative to the web root, of the directory that contains the CSS files used by ColdFusion AJAX features, with the exception of the rich text editor. This directory must have the same directory structure, and contain the same CSS files, and image files required by the CSS files, as the *web_root*/CFIDE/scripts/ajax/resources directory. This attribute lets you create different custom styles for ColdFusion AJAX controls in different applications.
`scriptSrc`	Optional	`scriptsrc` setting in the Administrator; default path is /CFIDE/scripts/	Specifies the URL, relative to the web root, of the directory that contains the client-side script files used by ColdFusion. This directory includes the JavaScript files and the default location of the CSS files used for all AJAX features. If you use this attribute, the `cfajaximport` tag must precede all other ColdFusion AJAX tags on the page; that is, all tags that rely on the scripts. You can have only one `scriptsrc` attribute on a page, in a `cfajaximport` tag or a `cfform` tag. You can use a `scriptsrc` attribute in a `cfajaximport` tag to apply its value to all forms on a page.
`tags`	Optional		A comma-delimited list of tags or tag-attribute combinations for which to import the supporting JavaScript files on this page. If you use this attribute, it must specify all ColdFusion AJAX tags that you use on the page and on any pages specified in tag `source` attributes. For a list of valid attribute values and their purposes, see Usage.

Usage

Using the scriptsrc and cssSrc attributes

The `scriptsrc` attribute is useful if the JavaScript files are not in the default location. This attribute may be required in some hosting environments and configurations that block access to the /CFIDE directory.

The default `scriptsrc` value is determined by the Default CFFORM ScriptSrc Directory setting on the Server Settings > Settings page of the ColdFusion Administrator. For `cfform` tags, the tag's `scriptsrc` attribute takes precedence over this attribute.

You can use this attribute only if the `cfajaximport` tag is on a top-level page; that is, a page that is requested directly by the client. You cannot use it, for example, on a page that is specified in a `cfwindow` tag `source` attribute.

When you use the `cfajaximport` tag with a `scriptsrc` attribute, the specified directory must have the same structure as the /CFIDE/scripts directory. For example, if you specify `scriptsrc="/resources/myScripts"`, the JavaScript files used by AJAX must be in the /resources/myScripts/ajax directory.

This attribute specifies the folder that contains the ColdFusion client-side files for all subsequent tags on the current page, not just for AJAX-based tags. Therefore, the directory tree must include all ColdFusion client-side files used by those tags. For example, if a `cfform` tag on the page is in Flash or applet format, you must include the CF_RunActiveContent.js file in the directory specified by the `scriptsrc` attribute.

You use the `cssSrc` attribute to specify the location of the CSS files required by ColdFusion AJAX features. This attribute overrides the *scriptsrc*/ajax/resources directory for the current page. Therefore, if all pages that use a custom `scriptsrc` directory also use a custom `cssSrc` directory, you do not have to include the ColdFusion AJAX CSS files in the `scriptsrc` directory tree.

Using the tags attribute or no attribute

If you do not use the `cfajaximport` tag on a page that contains ColdFusion tags with AJAX UI features, ColdFusion correctly imports the required JavaScript files in most cases. You must use this tag to explicitly import JavaScript files in these cases:

- If you use a ColdFusion AJAX JavaScript function, such as `ColdFusion.navigate`, `ColdFusion.Ajax.submitForm`, or `ColdFusion.Log.info` on a page that does not otherwise import any AJAX JavaScript functions, use the `cfajaximport` tag with no attribute to import the base JavaScript functions only. For example, use this tag on a page that does not include any ColdFusion AJAX-based tags,
- If the following conditions are true:
  - You use any `source` attributes in `cflayoutarea`, `cfpod` or `cfwindow` tags, or bind attribute in `cfdiv` tag.
  - The file that the `source` or `bind` attribute specifies has any of the tags listed in the following table.
  - You do *not* use each of the listed tags on the top-level page.

  If these conditions are true, the top-level page must use the `cfajaximport` tag with a `tags` attribute that specifies the tags that only the other pages use. Otherwise, ColdFusion cannot identify that it will be using the tags and does not import the necessary JavaScript files.

You can specify any or all of the following tag attribute values:

Attribute value	Used for
`cfdiv`	`cfdiv` tags
`cfform`	Forms that are in `cfpod`, `cfwindow`, or `cflayoutarea` tag bodies
`cfgrid`	AJAX format `cfgrid` tags

Attribute value	Used for
cfinput-autosuggest	cfinput tags that use the autosuggest attribute
cfinput-datefield	HTML format cfinput tags that use the datefield attribute
cflayout-border	cflayout tags with a type attribute value of border
cflayout-tab	cflayout tags with a type attribute value of tab
cfmenu	cfmenu tags
cfpod	cfpod tags
cfsprydataset-JSON	cfsprydataset tags that generate Spry JSON data sets
cfsprydataset-XML	cfsprydataset tags that generate Spry XML data sets
cftextarea	HTML format cftextarea tags
cftooltip	cftooltip tags
cftree	HTML format cftree tags
cfwindow	cfwindow tags

Example

The following cfajaximport tag example specifies separate custom locations for the scripts used for AJAX features and for the AJAX CSS files. It also imports all JavaScript files used for cftree, and cftooltip.

```
<cfajaximport cssSrc="/collegeApp/application/cssFiles"
    scriptsrc="/collegeApp/ajaxScripts"
    tags="cftooltip, cfwindow">
```

cfajaxproxy

Description

Creates a JavaScript proxy for a ColdFusion component, for use in an AJAX client. Alternatively, creates a proxy for a single CFC method, JavaScript function, or URL that is bound to one or more control attribute values.

Category

Internet protocol tags

Syntax

```
<cfajaxproxy
    cfc = "CFC name"
    jsclassname = "JavaScript proxy class name">

OR

<cfajaxproxy
    bind = "bind expression"
    onError = "JavaScript function name"
    onSuccess = "JavaScript function name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *whose value is a structure. Specify the structure name in the* `attributeCollection` *and use the tag's attribute names as structure keys.*

See also

`DeserializeJSON`, `IsJSON`, `SerializeJSON`, `Using Ajax Data and Development Features` in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag

Attributes

Attribute	Req/Opt	Default	Description
`bind`	A `bind` or `cfc` attribute is required		A bind expression that specifies a CFC method, JavaScript function, or URL to call. For detailed information about specifying bind expressions, see "Binding data to form fields" on page 650 in the *ColdFusion Developer's Guide*. You cannot use this attribute with the `cfc` attribute.
`cfc`	A `bind` or `cfc` attribute is required		The CFC for which to create a proxy. You must specify a dot-delimited path to the CFC. The path can be an absolute filepath, or relative to location of the CFML page. For example, if the myCFC CFC is in the cfcs subdirectory of the ColdFusion page, specify cfcs.myCFC. On UNIX based systems, the tag searches first for a file whose name or path corresponds to the specified name or path, but is in all lowercase. If it does not find it, ColdFusion then searches for a file name or path that corresponds to the attribute value exactly, with identical character casing. This attribute cannot be used with the `bind` attribute.
`jsclassname`	Optional	The value of the `cfc` attribute	The name to use for the JavaScript proxy class that represents the CFC. This attribute cannot be used with a `bind` attribute.

Attribute	Req/Opt	Default	Description
onError	Optional		The name of a JavaScript function to invoke when a bind, specified by the bind attribute fails. The function must take two arguments: an error code and an error message. This attribute cannot be used with a cfc attribute.
onSuccess	Optional		The name of a JavaScript function to invoke when a bind, specified by the bind attribute succeeds. The function must take one argument, the bind function return value. If the bind function is a CFC function, the return value is automatically converted to a JavaScript variable before being passed to the onSuccess function. This attribute cannot be used with a cfc attribute.

Usage

Make sure that the Enable HTTP Status Codes option on the Server Settings > Settings page of the ColdFusion Administrator is selected. Otherwise, the proxy cannot determine if an error occurs in invoking the CFC function and cannot call the error handler.

A cfajaxproxy tag with a bind attribute does not refresh any control that is not specified in the bind expression.

If you specify a URL in the bind attribute, the HTTP response must consist of a single JSON representation of an object, array, or simple value, corresponding to the onSuccess argument.

Creating a CFC proxy

The cfajaxproxy tag with a cfc attribute generates a JavaScript proxy that represents a CFC on the web client. The tag and the proxy it generates have the following characteristics:

- The proxy provides one function that corresponds to each CFC remote function. Calling these functions in your client-side JavaScript code remotely calls the CFC functions on the server.
- The proxy provides several functions that you call to configure the interaction between the client and the server. These functions set the HTTP method and synchronization mode of the XMLHttpRequest call that the proxy uses to interact with the server. The functions also can specify a JavaScript callback handler and an error handler for asynchronous calls.
- Because JavaScript is case-sensitive, you must ensure that you match the case of the keys in any ColdFusion structure or scope that you send to the client. By default, ColdFusion sets variable names and structure element names to all-uppercase. (You can create structure element names with lowercase characters by specifying the names in associative array notation, for example, myStruct["myElement"]="value".) The keys for the two arrays in the JSON object that the ColdFusion serializeJSON tag generates to represent a query are COLUMNS and DATA, for example, not columns and data.

For detailed information on using AJAX CFC proxies, see "Using ColdFusion Ajax CFC proxies" on page 658 in "Using Ajax Data and Development Features" on page 648 in the *ColdFusion Developer's Guide*.

Note: *The proxy passes a _CF_NODEBUG Boolean argument to called CFC functions. ColdFusion checks this value, and when it is true, does not append to the response any debugging information that it normally would send. This behavior ensures that the JSON responses to AJAX requests do include any non-JSON (i.e., debug information) text.*

CFC proxy utility functions

When you use the cfc option, the JavaScript proxy object provides the following functions that you can use to control interaction with the server:

Function	Description
`setAsyncMode()`	Sets the call mode to asynchronous. The calling thread (the Java thread of the client system that is processing the page) is not blocked when you make a call to a proxy function, so page processing can continue while waiting for a response from the server. The proxy invokes the function specified by the `setCallbackHandler` function with the response from the server. If an error occurs, the proxy invokes the error handler specified by the `setErrorHandler` function.
`setCallbackHandler(`*`function`*`)`	Specifies the callback handler for an asynchronous call. The *`function`* parameter is the JavaScript function to invoke as an argument. The callback function must take one parameter, the return value from the CFC that the proxy has deserialized from JSON to a JavaScript representation. This method automatically sets the call mode to asynchronous.
`setErrorHandler(`*`function`*`)`	Sets the error handler that the proxy invokes if there is an error in an asynchronous call. The *`function`* parameter is the JavaScript function to invoke. The error handler function must take two parameters: • An HTTP error code • A status message This method automatically sets the call mode to asynchronous.
`setForm(`*`ID`*`)`	Adds names and values of the fields in the form specified by the `ID` attribute to the arguments passed by a proxy function that is called immediately after this function. For more information, see "Submitting data to a CFC" on page 659 in the *ColdFusion Developer's Guide*.
`setHTTPMethod("`*`method`*`")`	Sets the HTTP method to use for the call. The *`function`* parameter is a case-insensitive string, and must have one of the following values: • `GET` (the default method) • `POST`

Function	Description
`setQueryFormat(format)`	Specifies the JSON format in which to return ColdFusion query data. The parameter must have one of the following values: • `row`: (Default) Sends the data as a JSON object with two entries: the column names and an array of row arrays. • `column`: Sends the data as a JSON object that represents WDDX query format. This object has three entries: the number of rows, an array with the column names, and an object where the keys are the column names and the values are arrays containing the column data. For more information on query formats, see `SerializeJSON`.
`setReturnFormat(format)`	Specifies the format in which the CFC function returns the result. ColdFusion automatically converts the function return value into the specified format before returning it to the client. The parameter must have one of the following values: • `json` (the default format if you don't use this function) • `plain` • `wddx` If you specify `plain`, and set the "content-type" header on the response from the server to `text/xml`, the proxy returns an XML object to the caller or callback function. If the content type is not set to `text/xml`, the return value from the server is returned as-is. This function is useful if you return XML or a plain string to the browser.
`setSyncMode()`	Sets the call mode to synchronous (the default synchronization mode). The calling thread remains blocked until the call returns. If an error occurs, the proxy throws an exception. In synchronous mode, the methods in the CFC proxy return the CFC method results directly to the caller.

Example

The following example uses a remote CFC method to populate a drop-down list of employees. When you select a name from the list, it uses a call to the CFC method to get information about the employee, and displays the results.

The application page has the following lines:

```
<!--- The cfajaxproxy tag creates a client-side proxy for the emp CFC.
        View the generated page source to see the resulting JavaScript.
        The emp CFC is in the components subdirectory of the directory that
        contains this page. --->
<cfajaxproxy cfc="components.emp" jsclassname="emp">

<html>
    <head>
        <script type="text/javascript">

            // Function to find the index in an array of the first entry
            // with a specific value.
            // It is used to get the index of a column in the column list.
            Array.prototype.findIdx = function(value){
                for (var i=0; i < this.length; i++) {
                    if (this[i] == value) {
                        return i;
                    }
                }
            }

            // Use an asynchronous call to get the employees for the
            // drop-down employee list from the ColdFusion server.
```

```
var getEmployees = function(){
    // create an instance of the proxy.
    var e = new emp();
    // Setting a callback handler for the proxy automatically makes
    // the proxy's calls asynchronous.
    e.setCallbackHandler(populateEmployees);
    e.setErrorHandler(myErrorHandler);
// The proxy getEmployees function represents the CFC
// getEmployees function.
    e.getEmployees();
}

// Callback function to handle the results returned by the
// getEmployees function and populate the drop-down list.
var populateEmployees = function(res)
{
    with(document.simpleAJAX){
        var option = new Option();
        option.text='Select Employee';
        option.value='0';
        employee.options[0] = option;
        for(i=0;i<res.DATA.length;i++){
            var option = new Option();
            option.text=res.DATA[i][res.COLUMNS.findIdx('FIRSTNAME')]
                + ' ' + res.DATA[i][[res.COLUMNS.findIdx('LASTNAME')]];
            option.value=res.DATA[i][res.COLUMNS.findIdx('EMP_ID')];
            employee.options[i+1] = option;
        }
    }
}

// Use an asynchronous call to get the employee details.
// The function is called when the user selects an employee.
var getEmployeeDetails = function(id){
    var e = new emp();
    e.setCallbackHandler(populateEmployeeDetails);
    e.setErrorHandler(myErrorHandler);
// This time, pass the employee name to the getEmployees CFC
// function.
    e.getEmployees(id);
}
// Callback function to display the results of the getEmployeeDetails
// function.
var populateEmployeeDetails = function(employee)
{
    var eId = employee.DATA[0][0];
    var efname = employee.DATA[0][1];
    var elname = employee.DATA[0][2];
    var eemail = employee.DATA[0][3];
    var ephone = employee.DATA[0][4];
    var edepartment = employee.DATA[0][5];

    with(document.simpleAJAX){
        empData.innerHTML =
        '<span style="width:100px">Employee Id:</span>'
        + '<font color="green"><span align="left">'
        + eId + '</font></span><br>'
        + '<span style="width:100px">First Name:</span>'
        + '<font color="green"><span align="left">'
        + efname + '</font></span><br>'
        + '<span style="width:100px">Last Name:</span>'
```

```
                    + '<font color="green"><span align="left">'
                    + elname + '</font></span><br>'
                    + '<span style="width:100px">Email:</span>'
                    + '<font color="green"><span align="left">'
                    + eemail + '</span></font><br>'
                    + '<span style="width:100px">Phone:</span>'
                    + '<font color="green"><span align="left">'
                    + ephone + '</font></span><br>'
                    + '<span style="width:100px">Department:</span>'
                    + '<font color="green"><span align="left">'
                    + edepartment + '</font></span>';
                }
            }

            // Error handler for the asynchronous functions.
            var myErrorHandler = function(statusCode, statusMsg)
            {
                alert('Status: ' + statusCode + ', ' + statusMsg);
            }

        </script>
    </head>

    <body>
        <!--- The form to display the employee drop-down list and
        employee data. --->
        <form name="simpleAJAX" method="get">
        List of Employees:   
        <select name="employee" onChange="getEmployeeDetails(this.value)">
            <script language="javascript">
                getEmployees();
            </script>
        </select>
        <br><br>
        <span id="empData"></span>
        </form>
    </body>
</html>
```

The following component, which gets the data from the data source, must be in a file named emp.cfc in the components subdirectory of the application directory. The CFC uses the cfdocexamples data source that is installed with ColdFusion if you install the documentation.

```
<cfcomponent>
    <cfset this.dsn = "cfdocexamples">
    <cffunction name="getEmployees" access="remote" returnFormat="json"
            output="false">
        <cfargument name="empid" required="no" type="string" default="0">
        <cfquery name="qryEmp" datasource="#this.dsn#">
            select * from Employees
            <cfif empid neq 0>
            where Emp_ID = #empid#
            </cfif>
        </Cfquery>
        <cfreturn qryEmp>
    </cffunction>
</cfcomponent>
```

cfapplet

Description

This tag references a registered custom Java applet. To register a Java applet, in the ColdFusion Administrator, select Extensions > Java Applets.

Using this tag within a `cfform` tag is optional. If you use it within `cfform`, and the `method` attribute is defined in the Administrator, the return value is incorporated into the form.

Category

Forms tags

Syntax

```
<cfapplet
    appletSource = "applet name"
    name = "form variable name"
    align = "alignment option"
    height = "height in pixels"
    hSpace = "space on each side in pixels"
    notSupported = "message to display for non-Java browser"
    param_1 = "applet parameter name"
    param_2 = "applet parameter name"
    param_n = "applet parameter name"
    vSpace = "space above and below in pixels"
    width = "width in pixels">
```

Note: *You can specify this tag's attributes in an `attributeCollection` whose value is a structure. Specify the structure name in the `attributeCollection` and use the tag's attribute names as structure keys.*

See also

`cfform`, `cfformgroup`, `cfformitem`, `cfgrid`, `cfinput`, `cfobject`, `cfselect`, `cfservlet`, `cfslider`, `cftextarea`, `cftree`

History

ColdFusion MX:

- Removed the requirement that you use this tag within a `cfform` tag.
- Changed the behavior when this tag is used within a `cfform` tag; if the `method` attribute is defined in the Administrator, the return value of the applet's method is incorporated into the form.

Attributes

Attribute	Req/Opt	Default	Description
`appletSource`	Required		Name of registered applet.
`name`	Required		Form variable name for applet.
`align`	Optional		Alignment: • `Left` • `Right` • `Bottom` • `Top` • `TextTop` • `Middle` • `AbsMiddle` • `Baseline` • `AbsBottom`
`height`	Optional		Height of applet, in pixels.
`hSpace`	Optional		Space on left and right of applet, in pixels.
`notSupported`	Optional	See description	Text to display if a page that contains a Java applet-based `cfform` control is opened by a browser that does not support Java or has Java support disabled, for example: `notSupported = "<b>Browser must support Java to view ColdFusion Java Applets</b>"` Default value: `<b>Browser must support Java to  view ColdFusion Java Applets!</b>`
`param_n`	Optional		Registered parameter for applet. Specify only to override values for applet in ColdFusion Administrator.
`vSpace`	Optional		Space above and below applet, in pixels.
`width`	Optional		Width of applet, in pixels.

Usage

You can specify the applet `method` attribute only in the Administrator, Java Applets view. For other attributes, you can accept the default values in the Administrator view, or specify values in this tag and override the defaults.

If Java applet components are stored in a JAR file, enter the information in the J2EE Archives > ColdFusion Administrator. For more information, see “Embedding Java applets” on page 552 in the *ColdFusion Developer’s Guide*

Example

```
<p>cfapplet lets you reference custom Java applets that have been
    registered using the ColdFusion Administrator.
<p>To register a Java applet, open the ColdFusion Administrator and
    click "Applets" link under "extensions" section.
<p>This example applet copies text that you type into a form. Type
    some text, and then click "copy" to see the copied text.

<cfform action = "index.cfm">
```

```
    <cfapplet appletsource = "copytext" name = "copytext">
</cfform>
```

cfapplication

Description

Defines the scope of a ColdFusion application; enables and disables storage of Client variables; specifies the Client variable storage mechanism; enables Session variables; and sets Application variable time-outs.

Category

Application framework tags

Syntax

```
<cfapplication
    name = "application name"
    applicationTimeout = #CreateTimeSpan(days, hours, minutes, seconds)#
    clientManagement = "yes|no"
    clientStorage = "data source name|Registry|Cookie"
    loginStorage = "cookie|session"
    scriptProtect = "none|all|list"
    sessionManagement = "yes|no"
    sessionTimeout = #CreateTimeSpan(days, hours, minutes, seconds)#
    setClientCookies = "yes|no"
    setDomainCookies = "yes|no">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *whose value is a structure. Specify the structure name in the* `attributeCollection` *and use the tag's attribute names as structure keys.*

See also

`cfassociate`, `cferror`, `cflock`, `cfmodule`; "Application.CFC Reference" on page 1304; "Designing and Optimizing a ColdFusion Application" on page 219 and "Integrating J2EE and Java Elements in CFML Applications" on page 929 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added `secureJSON` and `SecureJSONPrefix` attributes

ColdFusion MX 7: Added `scriptProtect` attribute

ColdFusion MX 6.1: Added `loginStorage` attribute

ColdFusion MX:

- Changed how persistent scopes are available: Server, Session, and Application scope variables are stored in memory as structures. In earlier releases, only Session and Application scope variables were stored this way. You cannot access the UDF function scope as a structure.
- Changed the algorithm for setting the CFTOKEN variable value: if the registry key UUIDToken is a nonzero value, ColdFusion uses a number constructed from the UUID plus a random number. Otherwise, ColdFusion sets the CFTOKEN variable default value using a positive random integer. (In earlier releases, ColdFusion always used a number constructed from the UUID plus a random number.)

Attributes

Attribute	Req/Opt	Default	Description
`name`	See Description		Name of application. Up to 64 characters. For Application and Session variables: Required. For Client variables: Optional
`applicationTimeout`	Optional	Specified in Variables page of ColdFusion Administrator	Lifespan of application variables. `CreateTimeSpan` function and values in days, hours, minutes, and seconds, separated by commas.
`clientManagement`	Optional	`no`	• `yes`: enables client variables. • `no`
`clientStorage`	Optional	`registry`	How client variables are stored: • `datasource_name`: in ODBC or native data source. You must create storage repository in the Administrator. • `registry`: in the system registry. • `cookie`: on client computer in a cookie. Scalable. If client disables cookies in the browser, client variables do not work.
`loginStorage`	Optional	`cookie`	• `cookie`: store login information in the Cookie scope. • `session`: store login information in the Session scope.
`scriptProtect`	Optional	Determined by ColdFusion Administrator Enable Global Script Protection setting	Specifies whether to protect variables from cross-site scripting attacks • `none`: do not protect variables • `all`: protect Form, URL, CGI, and Cookie variables • comma-delimited list of ColdFusion scopes: protect variables in the specified scopes. For more information, see Usage.
`secureJSON`	Optional	Administrator value	A Boolean value that specifies whether to add a security prefix in front of any value that a ColdFusion function returns in JSON-format in response to a remote call. The default value is the value of the Prefix serialized JSON setting in the Administrator Server Settings > Settings page (which defaults to `false`). You can override this variable value in the `cffunction` tag. For more information see "Improving security" on page 674 in the *ColdFusion Developer's Guide*.
`secureJSONPrefix`	Optional	Administrator value	The security prefix to put in front of the value that a ColdFusion function returns in JSON-format in response to a remote call if the `secureJSON` setting is `true`. The default value is the value of the Prefix serialized JSON setting in the Administrator Server Settings > Settings page (which defaults to `//`, the JavaScript comment character). For more information see "Improving security" on page 674 in the *ColdFusion Developer's Guide*.
`sessionManagement`	Optional	`no`	• `yes`: enables session variables. • `no`
`sessionTimeout`	Optional	Specified in Variables page of ColdFusion Administrator	Life span of session variables. `CreateTimeSpan` function and values in days, hours, minutes, and seconds, separated by commas.

Attribute	Req/Opt	Default	Description
`setClientCookies`	Optional	`yes`	• `yes`: enables client cookies. • `no`: ColdFusion does not automatically send CFID and CFTOKEN cookies to client browser; you must manually code CFID and CFTOKEN on the URL for every page that uses Session or Client variables.
`setDomainCookies`	Optional	`no`	• `yes`: uses domain cookies for CFID and CFTOKEN cookies and for all Client variables when using cookies for client variable storage. Required for applications running on clusters. • `no`: uses host-specific cookies for CFID, CFTOKEN, and all client variable cookies.

Usage

This tag is typically used in the Application.cfm file, to set defaults for a ColdFusion application.

Note: *You can also set the application defaults in the Application.cfc file. For more information, see* *"Application variables" on page 1305*.

This tag enables application variables, unless they are disabled in the ColdFusion Administrator. The Administrator setting also overrides the `sessionManagement` attribute. For more information, see *Configuring and Administering ColdFusion*.

If ColdFusion is running on a cluster, you must specify `clientStorage = "cookie"` or a data source name; you cannot specify `"registry"`.

ColdFusion generates an error if the application name is longer than 64 characters.

The CFTOKEN variable is 8 bytes in length. Its range is 10000000 —99999999.

Note: *If you specify `ClientStorage=cookie`, any Client scope variables set following a `cfflush` tag are not saved in the Client browser.*

Protecting variables from cross-site scripting attacks

The `ScriptProtect` attribute lets you protect one or more variable scopes from cross-site scripting attacks, where a client attempts to get your application to send malicious code back to a user's browser. In these attacks, user input (for example, from form fields or from URL variables) sets a CF variable which is destined for user output. The submitted data includes malicious code, such as JavaScript or an applet or object reference, which then executes on the user's system.

Note: *The ColdFusion Administrator Settings page Enable Global Script Protection option determines the default script protection setting. You can use the `scriptProtect` attribute to override the Administrator setting. You can also use the Application.cfc initialization code to set the protection value.*

The ColdFusion cross-site scripting protection operation is done when ColdFusion processes the application settings at the beginning of a request. Thus, it can process the URL, and Cookie, CGI, and Form variables in a user's request. By default, it replaces occurrences of the following HTML tag names with the text *InvalidTag*: `object`, `embed`, `script`, `applet`, and `meta`. It allows these names in plain text, and replaces the words if they are used as tag names.

You can specify any or all ColdFusion scopes for protection, but only the Form, URL, CGI, and Cookie scopes have variables that are often provided by unknown sources. Also, protecting a scope requires additional processing. For these reasons, the `all` attribute value applies protection to only the four scopes.

The script protection mechanism applies a regular expression that is defined in the *cf_root*/lib/neo-security.xml file in the server configuration, or the *cf_root*/WEB-INF/cfusion/lib/neo-security.xml file in the J2EE configuration to the variable value. You can customize the patterns that ColdFusion replaces by modifying the regular expression in the *CrossSiteScriptPatterns* variable.

Locking server, application, and session variables

When you set or update variables in the server, application, and session scopes, use the `cflock` tag with the `scope` attribute set to the following value:

- For server variables, specify `server`
- For application variables, specify `application`
- For session variables, specify `session`

In some cases, you should also lock code that reads variables in these scopes. For information about locking scopes, see "cflock" on page 366.

Example

```
<!--- This example shows how to use cflock to prevent race conditions during data updates
to variables in Application, Server, and Session scopes. --->
<h3>cfapplication Example</h3>
<p>cfapplication defines scoping for a ColdFusion application and enables or disables
application and/or session variable storage. This tag is placed in a special file called
Application.cfm that automatically runs before any other CF page in a directory (or
subdirectory) where the Application.cfm file appears.</p>

<cfapplication name = "ETurtle"
sessionTimeout = #CreateTimeSpan(0, 0, 0, 60)#
sessionManagement = "Yes">

<!--- Initialize session and application variables used by E-Turtleneck. --->
<cfparam name="application.number" default="1">
<cfparam name="session.color" default= "">
<cfparam name="session.size" default="">

<cfif IsDefined("session.numPurchased") AND IsNumeric(trim(session.cartTotal))>
<!--- Use the application scope for the application variable to prevent race condition. This
variable keeps track of total number of turtlenecks sold. --->
    <cflock scope = "Application" timeout = "30" type = "Exclusive">
    <cfset application.number = application.number + session.numPurchased>
    </cflock>
</cfif>

<cfoutput>
E-Turtleneck is proud to say that we have sold #application.number# turtlenecks to date.
</cfoutput>
<!--- End of Application.cfm --->
```

cfargument

Description

Creates a parameter definition within a component definition. Defines a function argument. Used within a `cffunction` tag.

History

ColdFusion 8: Added `component` as a valid value for the `ReturnType` attribute.

ColdFusion MX 7: Added the `xml` value of `type` attribute.

ColdFusion MX: Added this tag.

Category

Extensibility tags

Syntax

```
<cfargument
    name="string"
    default="default value"
    displayname="descriptive name"
    hint="extended description"
    required="yes|no"
    type="data type">
```

See also

`cfcomponent`, `cffunction`, `cfinterface`, `cfinvoke`, `cfinvokeargument`, `cfobject`, `cfproperty`, `cfreturn`

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		String; an argument name.
`default`	Optional		If no argument is passed, specifies a default argument value.
`displayname`	Optional	`name` attribute value	Meaningful only for CFC method parameters. A value to display when using introspection to show information about the CFC.
`hint`	Optional		Meaningful only for CFC method parameters. Text to display when using introspection to show information about the CFC. The `hint` attribute value follows the `displayname` attribute value in the parameter description line. Use this attribute to describe the purpose of the parameter.

Attribute	Req/Opt	Default	Description
required	Optional	*no*	**Note:** All arguments are required when invoked as a web service, irrespective of how they are defined. Specifies whether the parameter is required to execute the component method. The parameter is *not* required if you specify a default attribute. • yes • no
type	Optional	any	String; a type name; data type of the argument. • any • array • binary • boolean • component: the argument must be a ColdFusion component. • date • guid: the argument must be a UUID or GUID of the form *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* where each *x* is a character representing a hexadecimal number (0-9A-F). • numeric • query • string • struct • uuid: the argument must be a ColdFusion UUID of the form *xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx* where each *x* is a character representing a hexadecimal number (0-9A-F). • variableName: a string formatted according to ColdFusion variable naming conventions. • xml: XML objects and XML strings • a component name: if the type attribute value is not one of the preceding items, ColdFusion treats it as the name of a ColdFusion component. When the function executes, it generates an error if the argument that is passed in is not a CFC with the specified name.

Usage

This tag must be in a cffunction tag, before any other tags in the cffunction tag body.

Arguments that are passed when a method is invoked can be accessed from the method body in the following ways:

- With shorthand syntax: #myargument#

  (This example accesses the argument myargument.)
- Using the arguments scope as an array: #arguments[1]#

  (This example accesses the first defined argument in the cffunction.)
- Using the arguments scope as a struct: #arguments.myargument#

  (This example accesses the argument myargument in the array.)

Example

```
<!--- This example defines a function that takes a course number parameter and returns
the course description. --->
<cffunction name="getDescription">
    <!--- Identify argument. --->
    <cfargument name="Course_Number" type="numeric" required="true">
    <!--- Use the argument to get a course description from the database. --->
    <cfquery name="Description" datasource="cfdocexamples">
        SELECT Descript
        FROM Courses
        WHERE Number = '#Course_Number#'
    </cfquery>
    <!--- Specify the variable that the function returns. --->
    <cfreturn Description.Descript>
</cffunction>
```

cfassociate

Description

Allows subtag data to be saved with a base tag. Applies only to custom tags.

Category

Application framework tags

Syntax

```
<cfassociate
    baseTag = "base tag name"
    dataCollection = "collection name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *whose value is a structure. Specify the structure name in the* `attributeCollection` *and use the tag's attribute names as structure keys.*

See also

`cfapplication`, `cferror`, `cflock`, `cfmodule`; "High-level data exchange" on page 202 in the *ColdFusion Developer's Guide*.

Attributes

Attribute	Req/Opt	Default	Description
`baseTag`	Required		Base tag name.
`dataCollection`	Optional	`AssocAttribs`	Structure in which base tag stores subtag data.

Usage

Call this tag within a subtag, to save subtag data in the base tag.

When ColdFusion passes subtag attributes back to the base tag, it saves them in a structure whose default name is `AssocAttribs`. To segregate subtag attributes (in a base tag that can have multiple subtags), specify a structure name in the `dataCollection` attribute. The structure is appended to an array whose name is *thistag*.collectionName.

In the custom tag code, the attributes passed to the tag by using the `cfmodule` tag `attributeCollection` attribute are saved as independent values, with no indication that they are grouped into a structure by the custom tag's caller. Therefore, in the called tag, if you assign a value to a specific attribute, it replaces the value passed in the `attributeCollection` attribute that you used when calling the subtag.

Example

```
<!--- Find the context. --->
<cfif thisTag.executionMode is "start">
 <!--- Associate attributes. --->
 <cfassociate baseTag = "CF_TAGBASE">

 <!--- Define defaults for attributes. --->
 <cfparam name = "attributes.happy" default = "yes">
 <cfparam name = "attributes.sad" default = "no">
...
```

cfauthenticate

Description

This tag is obsolete. Use the newer security tools; see "Conversion functions" on page 641 and "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*.

History

ColdFusion MX: This tag is obsolete. It does not work in ColdFusion MX and later releases.

cfbreak

Description

Used within a `cfloop` tag. Breaks out of a loop.

Category

Flow-control tags

Syntax

```
<cfbreak>
```

See also

cfabort, cfexecute, cfif, cflocation, cfloop, cfthrow, cftry; "cfloop and cfbreak" on page 19 in the *ColdFusion Developer's Guide*

Example

```
<!--- This shows the use of cfbreak to exit a loop when a condition is met.--->
<!--- Select courses; use cfloop to find a condition; then break the loop. --->
<!--- Check that number is numeric. --->
<cfif IsDefined("form.course_number")>
    <cfif Not IsNumeric(form.course_number)>
        <cfabort>
    </cfif>
</cfif>
<cfquery name="GetCourses" datasource="cfdocexamples">
    SELECT *
    FROM Courses
    ORDER by course_number
</cfquery>

<p> This example uses CFLOOP to cycle through a query to find a value.
(In our example, a list of values corresponding to courses in the Snippets
datasource). When the conditions of the query are met, CFBREAK stops the loop. </p>
<p> Please enter a Course Number, and hit the "submit" button: </p>
<form action="cfbreak.cfm" method="POST">
    <select name="courseNum">
        <cfoutput query="GetCourses">
            <option value="#course_number#">#course_number#
        </cfoutput>
    </select>
    <input type="Submit" name="" value="Search on my Number">
</form>
<!--- If the courseNum variable is not defined, don't loop through the query.--->
<cfif IsDefined ("form.courseNum") IS "True">
<!--- Loop through query until value found, then use CFBREAK to exit query.--->
    <cfloop query="GetCourses">
        <cfif GetCourses.course_number IS form.courseNum>
            <cfoutput>
                <h4>Your Desired Course was found:</h4>
                <pre>#course_number# #descript#</pre>
            </cfoutput>
            <cfbreak>
        <cfelse>
            <br> Searching...
        </cfif>
    </cfloop>
</cfif>
```

cfcache

Description

Stores a copy of a page on the server and/or client computer, to improve page rendering performance. To do this, the tag creates temporary files that contain the static HTML returned from a ColdFusion page.

Use this tag if it is not necessary to get dynamic content each time a user accesses a page.

You can use this tag for simple URLs and for URLs that contain URL parameters.

Category

Page processing tags

Syntax

```
<cfcache
    action = "cache|flush|clientcache|servercache|optimal"
    directory = "directory name"
    expireURL = "wildcarded URL reference"
    password = "password"
    port = "port number"
    protocol = "http://|https://"
    timespan = "value">
    username = "username">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfflush`, `cfheader`, `cfhtmlhead`, `cfsetting`, `cfsilent`

History

ColdFusion MX:

- Deprecated the `cachedirectory` and `timeout` attributes. They might not work, and might cause an error, in later releases.
- Added the `timespan` attribute.
- Changed how pages are cached: the default `action` attribute value, `cache`, caches a page on the server and the client. (In earlier releases, this option cached a page only on the server.)
- Changed the source of the `protocol` and `port` values: the default `protocol` and `port` values are now taken from the current page URL. (In earlier releases, they were `"http"` and `"80"`, respectively.)
- Changed how session state is handled when caching a page: this tag can cache pages that depend on session state, including pages that are secured with a ColdFusion login. (In earlier releases, the session state was cleared when caching the page, causing authentication to be lost.)
- Changed how files are cached: this tag uses a hash() of the URL for the filename to cache a file. (In earlier releases, ColdFusion used the cfcache.map file.)

Attributes

Attribute	Req/Opt	Default	Description
`action`	Optional	`cache`	• `cache`: server-side and client-side caching. • `flush`: refresh cached page(s). • `clientcache`: browser-side caching only. To cache a personalized page, use this option. • `servercache`: server-side caching only. Not recommended. • `optimal`: same as `cache`.
`directory`	Optional	cf_root/cache	Absolute path of cache directory.
`expireURL`	Optional	Flush all cached pages	Used with `action = "flush"`. A URL reference. ColdFusion matches it against the mappings in the specified cache directory. Can include wildcards, for example: `"*/view.cfm?id=*"`.
`password`	Optional		A password. Provide this if the page requires authentication at the web-server level.
`port`	Optional	The port for the current page	Port number of the web server from which the URL is requested. In the internal call from `cfcache` to `cfhttp`, ColdFusion resolves each URL variable in the page; this ensures that links in the page remain functional.
`protocol`	Optional	The current page protocol	Protocol that is used to create URL from cache. • `http://` • `https://`
`timespan`	Optional	Page is flushed only when `cfcache action = "flush"` is executed	The interval until the page is flushed from the cache. • A decimal number of days, for example: ".25", for one-fourth day (6 hours); "1", for one day; "1.5", for one and one half days • A return value from the `CreateTimeSpan` function, for example, `"#CreateTimeSpan(0, 6, 0, 0)#"`.
`username`	Optional		A username. Provide this if the page requires authentication at the web server level.

Usage

Use this tag in pages whose content is not updated frequently. Taking this action can greatly improve the performance of your application.

The output of a cached page is stored in a file on the client browser and/or the ColdFusion server. Instead of regenerating and downloading the output of the page each time it is requested, ColdFusion uses the cached output. ColdFusion regenerates and downloads the page only when the cache is flushed, as specified by the `timespan` attribute, or by invoking `cfcache action=flush`.

To enable a simple form of caching, put a `cfcache` tag, specifying the `timespan` attribute, at the top of a page. Each time the specified time span passes, ColdFusion flushes (deletes) the copy of the page from the cache and caches a new copy for users to access.

You can specify client-side caching or a combination of client-side and server-side caching (the default), using the action attribute. The advantage of client-side caching is that it requires no ColdFusion resources; the browser stores pages in its own cache, improving performance. The advantage of combination caching is that it optimizes server performance; if the browser does not have a cache of the page, the server can get data from its own cache. (Adobe recommends that you use combination caching, and do not use server-side only caching.)

If a page contains personalized content, use the `action = "clientcache"` option to avoid the possibility of caching a personalized copy of a page for other users.

Debug settings have no effect on `cfcache` unless the application page enables debugging. When generating a cached file, `cfcache` uses `cfsetting showDebugOutput = "no"`.

The `cfcache` tag evaluates each unique URL, including URL parameters, as a distinct page, for caching purposes. For example, the output of http://server/view.cfm?id=1 and the output of http://server/view.cfm?id=2 are cached separately.

The `cfcache` tag uses the `cfhttp` tag to get the contents of a page to cache. If there is an HTTP error accessing the page, the contents are not cached. If a ColdFusion error occurs, the error is cached.

For more information, see "Caching ColdFusion pages that change infrequently" on page 240 in the *ColdFusion Developer's Guide*.

Example

```
<!--- This example produces as many cached files as there are URL parameter permutations.
You can see that the page is cached when the timestamp doesn't change.--->

<cfcache
    timespan="#createTimeSpan(0,0,10,0)#">
<body>

<h3>This is a test of some simple output</h3>

<cfoutput>
    This page was generated at #now()#<br>
</cfoutput>

<cfparam name = "URL.x" default = "no URL parm passed">
<cfoutput>The value of URL.x = # URL.x #</cfoutput>
```

cfcalendar

Description

Puts an interactive Flash format calendar in an HTML or Flash form. Not supported in XML format forms. The calendar lets a user select a date for submission as a form variable.

Category

Forms tags

Syntax

```
<cfcalendar
    name = "name of calendar"
    dayNames = "days of the week labels"
    disabled = "yes|no|no attribute value"
    enabled = "yes|no"
    endRange = "last disabled date"
    height = "height"
    mask = "character pattern"
    monthNames = "month labels"
    onBlur = "ActionScript to invoke"
    onChange = "ActionScript to invoke"
    onFocus = "ActionScript to invoke"
    selectedDate = "date"
    startRange = "first disabled date"
    style="Flash ActionScript style"
    tooltip = "text"
    visible = "yes|no"
    width = "width">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfform`, `cfgrid`, `cfinput`, `cfselect`, `cfslider`, `cftextarea`, `cftree`; "About Flash form styles" on page 588 in the *ColdFusion Developer's Guide*.

History

ColdFusion MX 7.01: Added support for `onBlur` and `onFocus` events.

ColdFusion MX 7: Added tag.

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		The name of the calendar.
`dayNames`	`Optional`	`S, M, T, W, Th, F, S`	A comma-delimited list that sets the names of the weekdays displayed in the calendar. Sunday is the first day and the rest of the weekday names follow in the normal order.
`disabled`	Optional	Not disabled	Disables all user input, making the control read-only. To disable input, specify `disabled` without an attribute or `disabled="Yes"` (or any ColdFusion positive boolean value, such as `true`). To enable input, omit the attribute or specify `disabled="No"` (or any ColdFusion negative Boolean value, such as `false`).
`enabled`	Optional	`yes`	Flash only: Boolean value that specifies whether the control is enabled. A disabled control appears in light gray. This is the inverse of the `disabled` attribute.

Attribute	Req/Opt	Default	Description
`endRange`	Optional		The end of a range of dates that are disabled. Users cannot select dates from the date specified by the `startRange` attribute through this date.
`firstDayOfWeek`	Optional	0	Integer in the range 0-6 specifying the first day of the week in the calendar: 0 indicates Sunday; 6 indicates Saturday.
`height`	Optional	Determined by Flash	The vertical dimension of the calendar specified in pixels.
`mask`	Optional	MM/DD/YYYY	A pattern that specifies the format of the submitted date. Mask characters are: • D = day; can use 0–2 mask characters • M = month; can use 0–4 mask characters • Y = year; can use 0, 2, or 4 characters • E = day in week; can use 0–4 characters • Any other character = put the character in the specified location For more information on masking, see Masking input data in the `cfinput` reference page.
`monthNames`	Optional	`January, February, March, April, May, June, July, August, September, October, November, December`	A comma-delimited list of the month names that are displayed at the top of the calendar.
`onBlur`	Optional		ActionScript that runs when the calendar loses focus.
`onChange`	Optional		ActionScript that runs when the user selects a date.
`onFocus`	Optional		ActionScript that runs when the calendar gets focus.
`selectedDate`	Optional	None (Flash shows the current month)	The date that is initially selected. It is highlighted in a color determined by the form skin. Must be in mm/dd/yyyy or dd/mm/yyyy format, depending on the current locale. (Use the `setlocale` tag to set the locale, if necessary.)
`startRange`	Optional		The start of a range of dates that are disabled. Users cannot select dates from this date through the date specified by the `endRange` attribute.
`style`	Optional		Flash ActionScript style or styles to apply to the calendar. For more information, see "Setting styles and skins in Flash forms" on page 588 in the *ColdFusion Developer's Guide*.
`tooltip`	Optional		Flash only: Text to display when the mouse pointer hovers over the control.
`visible`	Optional	`yes`	Flash only: Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank.
`width`	Optional	Determined by Flash	The horizontal dimension of the calendar specified in pixels.

Usage

The `cfcalendar` tag displays a calendar month, showing the month, the year, a grid of the days of the month, and headings for the days of the week. The calendar contains forward and back arrow buttons to let you change the month and year that are displayed.

If you include a value for the selectedDate attribute, that date is highlighted in green and determines the month and year that display initially. Changing the month and year display does not change the selected date. A user can change the selected date by clicking a different date on the calendar. The onChange attribute can specify an ActionScript event handler function that runs when the user selects a date.

The current date is highlighted in reverse (that is, a white number on a black background). If the selected date is in a different month or year, however, the current date does not appear unless you move to it by clicking the forward or back arrows.

The mask attribute lets you specify the format of the selected date that is returned to the application.

You can use the keyboard to access and select dates from a cfcalendar control:

- Use the Up, Down, Left, and Right Arrow keys to change the selected date.
- Use the Home and End keys to reach the first and last enabled date in a month, respectively.
- Use the Page Up and Page Down keys to reach the previous and next month, respectively.

Note: *The cfcalendar tag is not supported in XML format forms.*

Example

This example produces a 200-pixel by 150-pixel calendar with a Flash haloBlue skin. It displays abbreviated month names and two-character days of the week. It initially displays today's date as determined by the selectedDate attribute. When you click the Save button, the form submits back to the current page, which displays the submitted information.

The example also has three dateField controls that let the user change the initial selected date that displays on the calendar and a blocked-out date range. The initial blocked-out date is a four-day period immediately preceding today's date.

Note: *This example must be modified to work in locales that do not use mm/dd/yyyy date formats. To do so, use the LSDateFormat function in place of the DateFormat function and a mask that is appropriate for your locale, such as dd/mm/yyyy.*

```
<!--- Set initial selected and blocked-out dates.--->
<cfparam name="Form.startdate" default="#dateformat(now()-5, 'mm/dd/yyyy')#">
<cfparam name="Form.enddate" default="#dateformat(now()-1, 'mm/dd/yyyy')#">
<cfparam name="Form.selectdate" default="#dateformat(now(), 'mm/dd/yyyy')#">

<!--- If the form has been submitted, display the selected date. --->
<cfif isDefined("Form.submitit")>
    <cfoutput><b>You selected #Form.selectedDate#</b><br><br></cfoutput>
</cfif>

<b>Please select a date on the calendar and click Save.</b><br>
<br>
<cfform name="form1" format="Flash" skin="haloBlue" width="375" height="350" >
    <cfcalendar name="selectedDate"
        selectedDate="#Form.selectdate#"
        startRange="#Form.startdate#"
        endRange="#Form.enddate#"
        mask="mmm dd, yyyy"
        dayNames="SU,MO,TU,WE,TH,FR,SA"
        monthNames="JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC"
        style="rollOverColor:##FF0000"
        width="200" height="150">
    <cfinput type="dateField" name="startdate" label="Block out starts"
        width="100" value="#Form.startdate#">
```

```
    <cfinput type="dateField" name="enddate" label="Block out ends" width="100"
        value="#Form.enddate#">
    <cfinput type="dateField" name="selectdate" label="Initial date" width="100"
        value="#Form.selectdate#" >
    <cfinput type="Submit" name="submitit" value="Save" width="100">
</cfform>
```

cfcase

Description

Used only inside the cfswitch tag body. Contains code to execute when the expression specified in the cfswitch tag has one or more specific values.

Category

Flow-control tags

Syntax

```
<cfcase
    value = "value|delimited set of values"
    delimiters = "delimiter characters">
```

See also

cfdefaultcase, cfswitch; "cfswitch, cfcase, and cfdefaultcase" on page 18 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Changed the way ColdFusion parses cfcase values. Previously, cfcase tags with numeric value dates did not return expected results . For example, <cfcase value="00"> and <cfcase value="0A> were both evaluated to 0. The value "0A" was treated as a date and converted to 0 number of days from 12/30/1899. The value "00" was also evaluated to the value 0. This caused the exception "Context validation error for tag CFCASE. The CFSWITCH has a duplicate CFCASE for value "0.0"." The cfswitch tag now returns the expected result.

Attributes

Attribute	Req/Opt	Default	Description
value	Required		The value or values that the expression attribute of the cfswitch tag must match. To specify multiple matching values, separate the values with the delimiter character. The value or values must be simple constants or constant expressions, not variables.
delimiters	Optional	, (comma)	Specifies the delimiter character or characters that separate multiple values to match. If you specify multiple delimiter characters, you can use any of them to separate the values to be matched.

Usage

The contents of the cfcase tag body executes only if the expression attribute of the cfswitch tag evaluates to a value specified by the value attribute. The contents of the cfcase tag body can include HTML and text, and CFML tags, functions, variables, and expressions. You do not have to explicitly break out of the cfcase tag, as you do in some languages.

One cfcase tag can match multiple expression values. To do this, separate the matching values with the default value of the delimiter character. For example the following line matches "red", "blue", or "green":

```
<cfcase value="red,blue,green">
```

You can use the delimiter attribute to specify one or more delimiters to use in place of the comma. For example, the following line matches "cargo, live", "cargo, liquid", and "cargo, solid":

```
<cfcase value="cargo, live;cargo, liquid-cargo, solid" delimiters=";-">
```

Example

The following example displays a grade based on a 1-10 score. Several of the `cfcase` tags match more than one score. For simplicity, the example sets the score to 7.

```
<cfset score="7">
<cfswitch expression="#score#">
    <cfcase value="10">
        <cfset grade="A">
    </cfcase>
    <cfcase value="9;8" delimiters=";">
        <cfset grade="B">
    </cfcase>
    <cfcase value="7;6" delimiters=";">
        <cfset grade="C">
    </cfcase>
    <cfcase value="5;4;" delimiters=";">
        <cfset grade="D">
    </cfcase>
    <cfdefaultcase>
        <cfset grade="F">
    </cfdefaultcase>
</cfswitch>
<cfoutput>
    Your grade is #grade#
</cfoutput>
```

cfcatch

Description

Used inside a cftry tag. Together, they catch and process exceptions in ColdFusion pages. *Exceptions* are events that disrupt the normal flow of instructions in a ColdFusion page, such as failed database operations, missing include files, and developer-specified events.

Category

Exception handling tags

Syntax

```
<cfcatch type = "exception type">
    Exception processing code here
</cfcatch>
```

See also

cftry, cferror, cfrethrow, cfthrow, onError; "Handling Errors" on page 247 in the *ColdFusion Developer's Guide*

History

ColdFusion MX:

- Changed SQLSTATE value behavior: the SQLSTATE return value in a cfcatch tag depends on the database driver type:
  - Type 1 (JDBC-ODBC bridge): the value is the same as in ColdFusion 5.
  - Type 4 (100% Java, no native methods): the value might be different.

  If your application depends on SQLSTATE values for flow control, the application might produce unexpected behavior with ColdFusion MX.
- Changed the behavior of this tag when type="any": it is not necessary, when you include a cfcatch tag with type="any", to do so in the last cfcatch tag in the block, to ensure that all other tests are executed before it. ColdFusion finds the best-match cfcatch block.
- Changed the behavior of the cfscript tag: it includes try and catch statements that are equivalent to the cftry and cfcatch tags.
- Changed object modification: you cannot modify the object returned by cfcatch.
- Changed thrown exceptions: the cfcollection, cfindex, and cfsearch tags can throw the SEARCHENGINE exception. In earlier releases, an error in processing these tags threw only an UNKNOWN exception.

Attributes

Attribute	Req/Opt	Default	Description
type	Optional	Any	• application: catches application exceptions • database: catches database exceptions • template: catches ColdFusion page exceptions • security: catches security exceptions • object: catches object exceptions • missingInclude: catches missing include file exceptions • expression: catches expression exceptions • lock: catches lock exceptions • *custom_type*: catches the specified custom exception type that is defined in a cfthrow tag • searchengine: catches Verity search engine exceptions • any: catches all exception types

Usage

You must code at least one cfcatch tag within a cftry block. Put cfcatch tags at the end of a cftry block. ColdFusion tests cfcatch tags in the order in which they appear. This tag requires an end tag.

If type="any", ColdFusion catches exceptions from any CFML tag, data source, or external object. To get the exception type use code such as the following:

```
#cfcatch.type#
```

Applications can use the cfthrow tag to throw developer-defined exceptions. Catch these exceptions with any of these type options:

- "*custom_type*"
- "Application"
- "Any"

The *custom_type* type is a developer-defined type specified in a cfthrow tag. If you define a custom type as a series of strings concatenated by periods (for example, "MyApp.BusinessRuleException.InvalidAccount"), ColdFusion can catch the custom type by its character pattern. ColdFusion searches for a cfcatch tag in the cftry block with a matching exception type, starting with the most specific (the entire string), and ending with the least specific.

For example, you could define a type as follows:

```
<cfthrow type = "MyApp.BusinessRuleException.InvalidAccount">
```

If you have the following cfcatch tag, it handles the exception:

```
<cfcatch type = "MyApp.BusinessRuleException.InvalidAccount">
```

Otherwise, if you have the following cfcatch tag, it handles the exception:

```
<cfcatch type = "MyApp.BusinessRuleException">
```

Finally, if you have the following cfcatch tag, it handles the exception:

```
<cfcatch type = "MyApp">
```

You can code cfcatch tags in any order to catch a custom exception type.

If you specify `type = "Application"`, the `cfcatch` tag catches only custom exceptions that have the `Application` type in the `cfthrow` tag that defines them.

The `cfinclude`, `cfmodule`, and `cferror` tags throw an exception of `type = "template"`.

An exception that is thrown within a `cfcatch` block cannot be handled by the `cftry` block that immediately encloses the `cfcatch` tag. However, you can rethrow the currently active exception with the `cfrethrow` tag.

The `cfcatch` variables provide the following exception information:

cfcatch variable	Content
`cfcatch.type`	Type: Exception type, as specified in `cfcatch`.
`cfcatch.message`	Message: Exception's diagnostic message, if provided; otherwise, an empty string; in the `cfcatch.message` variable.
`cfcatch.detail`	Detailed message from the CFML interpreter or specified in a `cfthrow` tag. When the exception is generated by ColdFusion (and not `cfthrow`), the message can contain HTML formatting and can help determine which tag threw the exception.
`cfcatch.tagcontext`	An array of tag context structures, each representing one level of the active tag context at the time of the exception.
`cfcatch.NativeErrorCode`	Applies to `type = "database"`. Native error code associated with exception. Database drivers typically provide error codes to diagnose failing database operations. Default value is -1.
`cfcatch.SQLState`	Applies to `type = "database"`. SQLState associated with exception. Database drivers typically provide error codes to help diagnose failing database operations. Defaultvalue is -1.
`cfcatch.Sql`	Applies to `type = "database"`. The SQL statement sent to the data source.
`cfcatch.queryError`	Applies to `type ="database"`. The error message as reported by the database driver.
`cfcatch.where`	Applies to `type = "database"`. If the query uses the `cfqueryparam` tag, query parameter name-value pairs.
`cfcatch.ErrNumber`	Applies to `type = "expression"`. Internal expression error number.
`cfcatch.MissingFileName`	Applies to `type = "missingInclude"`. Name of file that could not be included.
`cfcatch.LockName`	Applies to `type = "lock"`. Name of affected lock (if the lock is unnamed, the value is `"anonymous"`).
`cfcatch.LockOperation`	Applies to `type = "lock"`. Operation that failed (Timeout, Create Mutex, or Unknown).
`cfcatch.ErrorCode`	Applies to `type = "custom"`. String error code.
`cfcatch.ExtendedInfo`	Applies to `type = "application"` and `"custom"`. Custom error message; information that the default exception handler does not display.

Advanced exception types

You can specify the following advanced exception types in the `type` attribute:

ColdFusion advanced exception type
COM.Allaire.ColdFusion.CFEXECUTE.OutputError
COM.Allaire.ColdFusion.CFEXECUTE.Timeout
COM.Allaire.ColdFusion.FileException
COM.Allaire.ColdFusion.HTTPAccepted
COM.Allaire.ColdFusion.HTTPAuthFailure

ColdFusion advanced exception type
COM.Allaire.ColdFusion.HTTPBadGateway
COM.Allaire.ColdFusion.HTTPBadRequest
COM.Allaire.ColdFusion.HTTPCFHTTPRequestEntityTooLarge
COM.Allaire.ColdFusion.HTTPCGIValueNotPassed
COM.Allaire.ColdFusion.HTTPConflict
COM.Allaire.ColdFusion.HTTPContentLengthRequired
COM.Allaire.ColdFusion.HTTPContinue
COM.Allaire.ColdFusion.HTTPCookieValueNotPassed
COM.Allaire.ColdFusion.HTTPCreated
COM.Allaire.ColdFusion.HTTPFailure
COM.Allaire.ColdFusion.HTTPFileInvalidPath
COM.Allaire.ColdFusion.HTTPFileNotFound
COM.Allaire.ColdFusion.HTTPFileNotPassed
COM.Allaire.ColdFusion.HTTPFileNotRenderable
COM.Allaire.ColdFusion.HTTPForbidden
COM.Allaire.ColdFusion.HTTPGatewayTimeout
COM.Allaire.ColdFusion.HTTPGone
COM.Allaire.ColdFusion.HTTPMethodNotAllowed
COM.Allaire.ColdFusion.HTTPMovedPermanently
COM.Allaire.ColdFusion.HTTPMovedTemporarily
COM.Allaire.ColdFusion.HTTPMultipleChoices
COM.Allaire.ColdFusion.HTTPNoContent
COM.Allaire.ColdFusion.HTTPNonAuthoritativeInfo
COM.Allaire.ColdFusion.HTTPNotAcceptable
COM.Allaire.ColdFusion.HTTPNotFound
COM.Allaire.ColdFusion.HTTPNotImplemented
COM.Allaire.ColdFusion.HTTPNotModified
COM.Allaire.ColdFusion.HTTPPartialContent
COM.Allaire.ColdFusion.HTTPPaymentRequired
COM.Allaire.ColdFusion.HTTPPreconditionFailed
COM.Allaire.ColdFusion.HTTPProxyAuthenticationRequired
COM.Allaire.ColdFusion.HTTPRequestURITooLarge
COM.Allaire.ColdFusion.HTTPResetContent

ColdFusion advanced exception type
COM.Allaire.ColdFusion.HTTPSeeOther
COM.Allaire.ColdFusion.HTTPServerError
COM.Allaire.ColdFusion.HTTPServiceUnavailable
COM.Allaire.ColdFusion.HTTPSwitchingProtocols
COM.Allaire.ColdFusion.HTTPUnsupportedMediaType
COM.Allaire.ColdFusion.HTTPUrlValueNotPassed
COM.Allaire.ColdFusion.HTTPUseProxy
COM.Allaire.ColdFusion.HTTPVersionNotSupported
COM.Allaire.ColdFusion.POPAuthFailure
COM.Allaire.ColdFusion.POPConnectionFailure
COM.Allaire.ColdFusion.POPDeleteError
COM.Allaire.ColdFusion.Request.Timeout
COM.Allaire.ColdFusion.SERVLETJRunError
COMCOM.Allaire.ColdFusion.HTTPConnectionTimeout

Example

```
<!--- The cfcatch example that uses TagContext to display the tag stack. --->
<h3>cftry Example</h3>
<!--- Open a cftry block. --->
<cftry>
    <!--- Notice misspelled tablename "employees" as "employeeas". --->
    <cfquery name = "TestQuery" dataSource = "cfdocexamples">
        SELECT *
        FROM employees
    </cfquery>
    <!--- Other processing goes here. --->
    <!--- Specify the type of error for which we search. --->
    <cfcatch type = "Database">
        <!--- The message to display. --->
        <h3>You've Thrown a Database <b>Error</b></h3>
        <cfoutput>
            <!--- The diagnostic message from ColdFusion. --->
            <p>#cfcatch.message#</p>
            <p>Caught an exception, type = #CFCATCH.TYPE#</p>
            <p>The contents of the tag stack are:</p>
            <cfdump var="#cfcatch.tagcontext#">
        </cfoutput>
    </cfcatch>
</cftry>
```

cfchart

Description

Generates and displays a chart.

Category

Data output tags, Extensibility tags

Syntax

```
<!--- This syntax uses an XML file or string to specify the chart style. --->
<cfchart
    style = "XML string|XML filename">
</cfchart>

OR

<!--- This syntax uses the attributes of the cfchart tag to specify the chart style. --->
<cfchart
    backgroundColor = "hexadecimal value|web color"
    chartHeight = "integer number of pixels"
    chartWidth = "integer number of pixels"
    dataBackgroundColor = "hexadecimal value|web color"
    font = "font name"
    fontBold = "yes|no"
    fontItalic = "yes|no"
    fontSize = "font size"
    foregroundColor = "hexadecimal value|web color"
    format = "flash|jpg|png"
    gridlines = "integer number of lines"
    labelFormat = "number|currency|percent|date"
    markerSize = "integer number of pixels"
    name = "string"
    pieSliceStyle = "solid|sliced"
    scaleFrom = "integer minimum value"
    scaleTo = "integer maximum value"
    seriesPlacement = "default|cluster| stacked|percent"
    show3D = "yes|no"
    showBorder = "yes|no"
    showLegend = "yes|no"
    showMarkers = "yes|no"
    showXGridlines = "yes|no"
    showYGridlines = "yes|no"
    sortXAxis = "yes|no"
    tipBGColor = "hexadecimal value|web color"
    tipStyle = "MouseDown|MouseOver|none"
    title = "title of chart"
    url = "onClick destination page"
    xAxisTitle = "title text"
    xAxisType = "scale|category"
    xOffset = "number between -1 and 1"
    yAxisTitle = "title text"
    yAxisType = "scale|category"
    yOffset = "number between -1 and 1">
</cfchart>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfchartdata, cfchartseries, cfdocument, "Controlling chart appearance" on page 796 in the *ColdFusion Developer's Guide*

History

ColdFusion 8:

- Added support for embedding Flash charts within the cfdocument tag.
- Added the new attribute showColumnLegend to the chart style files, which are the XML files located in the charting\styles directory- This attribute displays an entry for each point and is applicable only to charts that contain a single series. By default, the value of showColumnLegend is set to true. To turn off this feature, you can either modify the setting in all the chart style files, or use a custom style file.

ColdFusion MX 7.01: Changed documentation to state that the fontSize attribute can accept a number that is not an integer.

ColdFusion MX 7:

- Added style and title attributes.
- Added support for eight-digit hexadecimal values to specify RGB color and transparency.
- Removed the rotated attribute.

ColdFusion MX 6.1:

- Added the xAxisType and yAxisType attributes.
- Changed interpolation behavior: the tag now interpolates data points on line charts with multiple series.

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
backgroundColor	Optional		Color of the area between the data background and the chart border, around labels and around the legend. Hexadecimal value or supported named color; see the name list in Usage. For a hexadecimal value, use the form "##xxxxxx" or "##xxxxxxxx", where x = 0-9 or A-F; use two number signs or none.
chartHeight	Optional	240	Chart height; integer number of pixels.
chartWidth	Optional	320	Chart width; integer number of pixels.
dataBackgroundColor	Optional	white	Color of area around chart data. Hexadecimal value or supported named color; see the name list in the Usage section. For a hexadecimal value, use the form "##xxxxxx" or "##xxxxxxxx", where x = 0-9 or A-F; use two number signs or none.
font	Optional	arial	Name of text font: • arial • times • courier • arialunicodeMS. This option is required, if you are using a double-byte character set on UNIX, or using a double-byte character set in Windows with a file type of Flash.

Attribute	Req/Opt	Default	Description
fontBold	Optional	no	Whether to make the text bold: • yes • no
fontItalic	Optional	no	Whether to make the text italicized: • yes • no
fontSize	Optional	11	Font size. If the number is not an integer, ColdFusion rounds the number up to the next integer.
foregroundColor	Optional	black	Color of text, grid lines, and labels. Hexadecimal value or supported named color; see name list in the Usage section. For a hexadecimal value, use the form "##xxxxxx" or "##xxxxxxxx", where x = 0-9 or A-F; use two number signs or none.
format	Optional	flash	File format in which to save the graph: • flash • jpg • png
gridlines	Optional	10, including top and bottom	Number of grid lines to display on the value axis, including axis; positive integer.
labelFormat	Optional	number	Format for y-axis labels: • number • currency • percent • date
markerSize	Optional	(Automatic)	Size of data point marker in pixels; integer.
name	Optional		Page variable name; string. Generates the graph as binary data and assigns it to the specified variable. Suppresses chart display. You can use the name value in the cffile tag to write the chart to a file.
pieSliceStyle	Optional	sliced	Applies to the cfchartseries type attribute value pie. • solid: displays pie as if unsliced. • sliced: displays pie as if sliced.
scaleFrom	Optional	Determined by data	Y-axis minimum value; integer.
scaleTo	Optional	Determined by data	Y-axis maximum value; integer.
seriesPlacement	Optional	default	Relative positions of series in charts that have more than one data series. • default: ColdFusion determines relative positions, based on graph types • cluster • stacked • percent

Attribute	Req/Opt	Default	Description
show3D	Optional	yes	Whether to display the chart with three-dimensional appearance: • yes • no
showBorder	Optional	no	Whether to display a border around the chart: • yes • no
showLegend	Optional	yes	Whether to display the legend if the chart contains more than one data series: • yes • no
showMarkers	Optional	yes	Whether to display markers at data points in line, curve, and scatter graphs: • yes • no
showXGridlines	Optional	no	Whether to display x-axis gridlines: • yes • no
showYGridlines	Optional	yes	Whether to display y-axis gridlines: • yes • no
sortXAxis	Optional	no	Whether to display column labels in alphabetic order along the x axis: • yes • no Ignored if the xAxisType attribute is scale.
style	Optional		XML file or string to use to specify the style of the chart.
title	Optional		Title of the chart.
tipbgcolor	Optional	white	Background color of tips. Applies only to Flash format graph files. Hexadecimal value or supported named color; see the name list in the Usage section. For a hexadecimal value, use the form "##xxxxxx", where x = 0-9 or A-F; use two number signs or none.
tipStyle	Optional	mouseOver	Determines the action that opens a pop-up window to display information about the current chart element. • mouseDown: display if the user positions the cursor at the element and clicks the mouse. Applies only to Flash format graph files. (For other formats, this option functions the same as mouseOver.) • mouseOver: displays if the user positions the cursor at the element. • none: suppresses display.

Attribute	Req/Opt	Default	Description
url	Optional		URL to open if the user clicks item in a data series; the onClick destination page. You can specify variables within the URL string; ColdFusion passes current values of the variables. • $VALUE$: the value of the selected row. If none, the value is an empty string. • $ITEMLABEL$: the label of the selected item. If none, the value is an empty string. • $SERIESLABEL$: the label of the selected series. If none, the value is an empty string, for example: "somepage.cfm?item=$ITEMLABEL$&series=$SERIESLABEL$&value=$VALUE$ • "javascript:...": executes a client-side script.
xAxisTitle	Optional		Title that appears on the x axis; text.
xAxisType	Optional	category	Whether the x axis indicates data or is numeric: • category: The axis indicates the data category. Data is sorted according to the sortXAxis attribute. • scale: The axis is numeric. All cfchartdata item attribute values must be numeric. The x axis is automatically sorted numerically.
xOffset	Optional	0.1	Number of units by which to display the chart as angled, horizontally. Applies if show3D="yes". The number can be between -1 and 1, where "-1" specifies 90 degrees left and "1" specifies 90 degrees right.
yAxisTitle	Optional		Title of the y axis; text.
yAxisType	Optional	category	Currently has no effect, as the y axis is always used for data values.
yOffset	Optional	0.1	Number of units by which to display the chart as angled, vertically. Applies if show3D="yes". The number can be between -1 and 1, where "-1" specifies 90 degrees left and "1" specifies 90 degrees right.

Usage

The cfchart tag defines a *container* in which a graph displays: its height, width, background color, labels, and so on. The cfchartseries tag defines the chart style in which data displays: bar, line, pie, and so on. The cfchartdata tag defines a data point.

Data is passed to the cfchartseries tag in the following ways:

- As a query
- As data points, using the cfchartdata tag

For the font attribute value ArialUnicodeMS, the following rules apply:

- In Windows, to permit Flash charts (type = "flash") to render a double-byte character set, you must select this value.
- In UNIX, for all type values, to render a double-byte character set, you must select this value.
- If this value is selected, the fontBold and fontItalic attributes have no effect.

The following table lists W3C HTML 4 named color value or hexadecimal values that the color attribute accepts:

Color name	RGB value
Aqua	##00FFFF
Black	#000000
Blue	##0000FF
Fuchsia	##FF00FF
Gray	##808080
Green	##008000
Lime	##00FF00
Maroon	##800000
Navy	##000080
Olive	##808000
Purple	##800080
Red	##FF0000
Silver	##C0C0C0
Teal	##008080
White	##FFFFFF
Yellow	##FFFF00

For all other color values, you must enter the hexadecimal value. You can enter a six-digit value, which specifies the RGB value, or an eight-digit value, which specifies the RGB value and the transparency. The first two digits of an eight-digit hexadecimal value specify the degree of transparency, with FF indicating opaque and 00 indicating transparent. Values between 00 and FF are allowed.

For more color names that are supported by popular browsers, go to www.w3.org/TR/css3-color

You can specify whether charts are cached in memory, the number of charts to cache, and the number of chart requests that ColdFusion can process concurrently. To set these options in the ColdFusion Administrator, select Server Settings > Charting.

ColdFusion 8 lets you embed Flash charts in a PDF document using the `cfdocument` tag, for example:

```
<cfdocument format="pdf">
...
<!--- Flash chart embedded here. --->
    <cfchart format="flash" xaxistitle="Department" yaxistitie="Salary Average">
        <cfchartseries type="bar" query="DataTable" itemcolumn="Dept_Name"
            valuecolumn="avgSal">
            <cfchardata item="Facilities" value="35000">
        </cfchartseries>
    </cfchart>
</cfdocument>
```

Example

```
<!---The following example analyzes the salary data in the cfdocexamples database and
    generates a bar chart showing average salary by department. The body of the
    cfchartseries tag includes one cfchartdata tag to include data that is not available
    from the query. --->
```

```
<!--- Get the raw data from the database. --->
<cfquery name="GetSalaries" datasource="cfdocexamples">
    SELECT Departmt.Dept_Name,
        Employee.Dept_ID,
        Employee.Salary
    FROM Departmt, Employee
    WHERE Departmt.Dept_ID = Employee.Dept_ID
</cfquery>

<!--- Use a query of queries to generate a new query with --->
<!--- statistical data for each department. --->
<!--- AVG and SUM calculate statistics. --->
<!--- GROUP BY generates results for each department. --->
<cfquery dbtype = "query" name = "DataTable">
    SELECT Dept_Name,
    AVG(Salary) AS avgSal,
    SUM(Salary) AS sumSal
    FROM GetSalaries
    GROUP BY Dept_Name
</cfquery>

<!--- Reformat the generated numbers to show only thousands. --->
<cfloop index = "i" from = "1" to = "#DataTable.RecordCount#">
    <cfset DataTable.sumSal[i] = Round(DataTable.sumSal[i]/1000)*1000>
    <cfset DataTable.avgSal[i] = Round(DataTable.avgSal[i]/1000)*1000>
</cfloop>

<h1>Employee Salary Analysis</h1>
<!--- Bar graph, from Query of Queries --->
<cfchart format="flash"
    xaxistitle="Department"
    yaxistitle="Salary Average">

<cfchartseries type="bar"
    query="DataTable"
    itemcolumn="Dept_Name"
    valuecolumn="avgSal">

<cfchartdata item="Facilities" value="35000">

</cfchartseries>
</cfchart>
```

cfchartdata

Description

Used with the cfchart and cfchartseries tags. This tag defines chart data points. Its data is submitted to the cfchartseries tag.

Category

Data output tags, Extensibility tags

Syntax

```
<cfchartdata
    item = "text"
    value = "number">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfchart, cfchartseries; "Creating Charts and Graphs" on page 787 in the *ColdFusion Developer's Guide*

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
item	Required		Data point name; string.
value	Required		Data point value; number or expression.

Example

```
<!--- The following example analyzes the salary data in the cfdocexamples
database and generates a bar chart showing average salary by department. The
body of the cfchartseries tag loops over a cfchartdata tag to include data
available from the query. --->

<!--- Get the raw data from the database. --->
<cfquery name="GetSalaries" datasource="cfdocexamples">
    SELECT Departmt.Dept_Name,
        Employee.Dept_ID,
        Employee.Salary
    FROM Departmt, Employee
    WHERE Departmt.Dept_ID = Employee.Dept_ID
</cfquery>

<!--- Use a query of queries to generate a new query with --->
<!--- statistical data for each department. --->
<!--- AVG and SUM calculate statistics. --->
<!--- GROUP BY generates results for each department. --->
<cfquery dbtype = "query" name = "DataTable">
    SELECT Dept_Name,
    AVG(Salary) AS avgSal,
    SUM(Salary) AS sumSal
    FROM GetSalaries
    GROUP BY Dept_Name
</cfquery>

<!--- Reformat the generated numbers to show only thousands. --->
```

```
<cfloop index = "i" from = "1" to = "#DataTable.RecordCount#">
<cfset DataTable.sumSal[i] = Round(DataTable.sumSal[i]/1000)*1000>
<cfset DataTable.avgSal[i] = Round(DataTable.avgSal[i]/1000)*1000>
</cfloop>

<h1>Employee Salary Analysis</h1>
<!--- Bar graph, from Query of Queries. --->
<cfchart format="flash"
xaxistitle="Department"
yaxistitle="Salary Average">

<cfchartseries type="bar"
itemcolumn="Dept_Name"
valuecolumn="avgSal">

<cfloop query="DataTable">
<cfchartdata item="#DataTable.Dept_Name#" value="#DataTable.avgSal#">
</cfloop>

</cfchartseries>
</cfchart>
```

cfchartseries

Description

Used with the `cfchart` tag. This tag defines the chart style in which the data displays: bar, line, pie, and so on.

Category

Data output tags, Extensibility tags

Syntax

```
<cfchartseries
    type="type"
    itemColumn="query column"
    valueColumn="query column"
    colorlist = "list"
    dataLabelStyle="style"
    markerStyle="style"
    paintStyle="plain|raise|shade|light"
    query="query name"
    seriesColor="hexadecimal value|web color"
    seriesLabel="label text">
</cfchartseries>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfchart`, `cfchartdata`; "Creating Charts and Graphs" on page 787 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7:

- Added the `dataLabelStyle` attribute.
- Added the `horizontalbar` value of the `type` attribute.

ColdFusion MX 6.1: Changed interpolation behavior: the tag now interpolates data points on line charts with multiple series.

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
type	Required		Sets the chart display style: • bar • line • pyramid • area • horizontalbar • cone • curve • cylinder • step • scatter • pie
itemColumn	Required if query attribute is specified		Name of a column in the query specified in the query attribute; contains the item label for a data point to graph.
valueColumn	Required if query attribute is specified		Name of a column in the query specified in the query attribute; contains data values to graph.
colorlist	Optional		Sets colors for each data point. Applies if the cfchartseries type attribute is pie, pyramid, area, horizontalbar, cone, cylinder, or step. Comma-delimited list of hexadecimal values or supported, named web colors; see the name list and information about six- and eight-digit hexadecimal values in the cfchart Usage section. For a hexadecimal value, use the form "##xxxxxx" or "##xxxxxxxx", where x = 0-9 or A-F; use two number signs or none.
dataLabelStyle	Optional	none	Specifies the way in which the color is applied to the item in the series: • none: nothing is printed. • value: the value of the datapoint. • rowLabel: the row's label. • columnLabel: the column's label. • pattern: combination of column label, value, and aggregate information, such as the columnLabel value for the percentage of total graph, for example, Sales 55,000 20% of 277,000.

Attribute	Req/Opt	Default	Description
markerStyle	Optional	rectangle	Sets the icon that marks a data point for two-dimensional line, curve, and scatter graphs: • rectangle • triangle • diamond • circle • letter • mcross • snow • rcross
paintStyle	Optional	plain	Sets the paint display style of the data series: • plain: solid color. • raise: the appearance of a button. • shade: gradient fill, darker at the edges. • light: a lighter shade of color; gradient fill.
query	Optional		Name of the ColdFusion query from which to get data to graph.
seriesColor	Optional		Color of the main element (such as the bars) of a chart. For a pie chart, the color of the first slice. Hexadecimal value or supported named color; see the name list and information about six- and eight-digit hexadecimal values in the Usage section for the cfchart tag. For a hexadecimal value, use the form "##xxxxxx" or "##xxxxxxxx", where x = 0-9 or A-F; use two number signs or none.
seriesLabel	Optional		Text of the data series label

Usage

For a pie chart, ColdFusion sets pie slice colors as follows:

- If the seriesColor attribute is omitted, ColdFusion automatically determines the colors of the slices.
- If the seriesColor attribute is specified, ColdFusion automatically determines the colors of the slices after the first one, starting with the specified color for the first slice.

Example

```
<!--- The following example analyzes the salary data in the cfdocexamples
database and generates a bar chart showing average salary by department. --->

<!--- Get the raw data from the database. --->
<cfquery name="GetSalaries" datasource="cfdocexamples">
    SELECT Departmt.Dept_Name,
        Employee.Dept_ID,
        Employee.Salary
    FROM Departmt, Employee
    WHERE Departmt.Dept_ID = Employee.Dept_ID
</cfquery>

<!--- Use a query of queries to generate a new query with --->
<!--- statistical data for each department. --->
```

```
<!--- AVG and SUM calculate statistics. --->
<!--- GROUP BY generates results for each department. --->
<cfquery dbtype = "query" name = "DataTable">
SELECT
Dept_Name,
AVG(Salary) AS avgSal,
SUM(Salary) AS sumSal
FROM GetSalaries
GROUP BY Dept_Name
</cfquery>

<!--- Reformat the generated numbers to show only thousands. --->
<cfloop index = "i" from = "1" to = "#DataTable.RecordCount#">
<cfset DataTable.sumSal[i] = Round(DataTable.sumSal[i]/1000)*1000>
<cfset DataTable.avgSal[i] = Round(DataTable.avgSal[i]/1000)*1000>
</cfloop>

<h1>Employee Salary Analysis</h1>
<!--- Bar graph, from Query of Queries --->
<cfchart format="flash"
xaxistitle="Department"
yaxistitle="Salary Average">

<cfchartseries type="bar"
query="DataTable"
itemcolumn="Dept_Name"
valuecolumn="avgSal" />
</cfchart>
```

cfcol

Description

Defines table column header, width, alignment, and text. Used within a `cftable` tag.

Category

Data output tags

Syntax

```
<cfcol
    header = "column header text"
    text = "column text"
    align = "left|right|center"
    width = "number that indicates width of column">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfcontent`, `cfoutput`, `cftable`; "Performing file operations with cfftp" on page 1044 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Added the ability to construct dynamic `cfcol` statements.

Attribute	Req/Opt	Default	Description
`header`	Required		Column header text. To use this attribute, you must also use the `cftable colHeaders` attribute.
`text`	Required		Double-quotation mark-delimited text; determines what to display. Rules: same as for `cfoutput` sections. You can embed hyperlinks, image references, and input controls.
`align`	Optional	`left`	Column alignment: • `left` • `right` • `center`
`width`	Optional	20	Column width. If the length of data displayed exceeds this value, data is truncated to fit. To avoid this, use an HTML `table` tag. If the surrounding `cftable` tag includes the `htmltable` attribute, `width` specifies the percent of the table width and it does not truncate text; otherwise, `width` specifies the number of characters.

Usage

At least one `cfcol` tag is required within the `cftable` tag. You must put `cfcol` and `cftable` tags adjacent in a page. The only tag that you can nest within the `cftable` tag is the `cfcol` tag. You cannot nest `cftable` tags.

To display the `cfcol header` text, you must specify the `cfcol header` and the `cftable colHeader` attribute. If you specify either attribute without the other, the header does not display. No error is thrown.

Example

```
<!--- This example shows the use of cfcol and cftable to align
    information returned from a query. --->
<!--- Query selects information from cfdocexamples data source. --->
```

```
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
    SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
    FROM Employees
</cfquery>
<html>
<body>
<h3>cfcol Example</h3>
<!--- Uses the HTMLTable attribute to display cftable as an HTML
    table, rather than PRE formatted information --->
<cftable
    query = "GetEmployees"
    startRow = "1" colSpacing = "3"
    HTMLTable colheaders>
<!--- Each cfcol tag sets the width of a column in the table,
    the header information, and the text/CFML for the cell. --->
    <cfcol header = "<b>ID</b>"
        align = "Left"
        width = 2
        text= "#Emp_ID#">
    <cfcol header = "<b>Name/Email</b>"
        align = "Left"
        width = 15
        text= "<a href = 'mailto:#Email#'>#FirstName# #LastName#</A>">
    <cfcol header = "<b>Phone Number</b>"
        align = "Center"
        width = 15
        text= "#Phone#">
</cftable>
```

cfcollection

Description

Creates and administers Verity search engine collections.

Category

Extensibility tags

Syntax

```
<cfcollection
    action = "action"
    categories = "yes|no"
    collection = "collection name"
    language = "language"
    name = "query name"
    path = "c">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfexecute`, `cfindex`, `cfobject`, `cfreport`, `cfsearch`, `cfwddx`

History

ColdFusion MX 7:

- Starting with ColdFusion MX 7, you cannot use the cfcollection tag to create alias names for already existing collections. Because Verity maintains all the collection information, you cannot have two names point to the same collection.
- Removed reference to external collections.
- Deprecated the `map` and `repair` options of the `action` attribute. They might not work, and might cause an error, in later releases.
- Added `categories` attribute and `categorylist` action.
- Added CATEGORIES, SIZE, DOCCOUNT, and LASTMODIFIED to list of variables returned by the `list` action.
- Marked as obsolete the MAPPED, ONLINE, and REGISTERED variables returned by the `list` action.

ColdFusion MX:

- Changed the requirements for the `action` attribute: it is now required.
- Added the `action` attribute `list` value. It is the default.
- Changed the requirements for the `action` attribute value `map`: it is not necessary to specify the `action` attribute value `map`. (ColdFusion detects collections and creates maps collections as required.)
- Changed acceptable collection naming: this tag accepts collection names that include spaces.
- Changed Verity operations behavior: ColdFusion supports Verity operations on Acrobat PDF files.
- Changed thrown exceptions: this tag can throw the SEARCHENGINE exception.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required; see Usage	`list`	• `categorylist`: retrieves categories from the collection and indicates how many documents are in each one. Returns a structure of structures in which the category representing each substructure is associated with a number of documents. For a category in a category tree, the number of documents is the number at *or below* that level in the tree. • `create`: registers the collection with ColdFusion. If the collection is present, the tag creates a map to it. If the collection is not present, the tag creates it. • `delete`: unregisters a collection and deletes its directories. • `list`: returns a query result set, named from the `name` attribute value, of the attributes of the collections that are registered by ColdFusion. • `map`: creates a map to a collection. If the action is `create` and the collection already exists, ColdFusion also creates a map to the collection. • `optimize`: optimizes the structure and contents of the collection for searching; recovers space. Causes collection to be taken offline, preventing searches and indexing. • `repair`: deprecated. Does nothing.
`categories`	See Usage	`no`	Used only for creating a collection: • `yes`: this collection includes support for categories. • `no`: this collection does not support categories.
`collection`	See Usage		• A collection name. The name can include spaces.
`language`	See Usage	`English`	Although English is the default language, Englishx, a more advanced English locale, is also provided. For a list of options, see Usage. Requires the appropriate (European or Asian) Verity Locales language pack.
`name`	See Usage		Name for the query results returned by the `list` and `categorylist` actions.
`path`	See Usage		Absolute path to a Verity collection. To map an existing collection, specify a fully qualified path to the collection (not including the collection name); for example, `"C:\MyCollections\"`.

Usage

With this tag you can create, register, and administer a Verity collection that was created by ColdFusion or by a Verity application.

The following table shows the dependence relationships among this tag's attribute values:

This attribute is required, optional, or unnecessary (blank):	**For this action attribute value:**						
	`list`	**`create`**	**`map`**	**`optimize`**	**`repair`**	**`delete`**	**`categorylist`**
`collection`		Required	Required	Required	Required	Required	Required
`path`		Required	Required				
`language`		Optional	Optional				
`name`	Required						Required
`categories`							

The following examples show the structures returned by the `categorylist` action:

CATEGORIES	
blue	10
green	3
magenta	3
purple	2
CATEGORYTREES	
a/	10
a/b	10
a/b/c	10
a/b/c/subdir	3

The `list` action returns the following information in a result set that contains one row per collection:

Column	Contents
CATEGORIES	• `yes`: the collection has category support enabled. • `no`: the collection does not have category support enabled.
CHARSET	The character set of the collection.
CREATED	The date and time that the collection was created.
DOCCOUNT	The number of documents in this collection.
EXTERNAL	• `yes`: the collection is external. • `no`: the collection is not external. • `not found`: the collection is registered but is not available in the defined path .
LANGUAGE	The locale setting of the collection. This information is not available for K2Server collections.
LASTMODIFIED	The date and time that the collection was last changed.
MAPPED	Obsolete.
NAME	The name of the collection.
ONLINE	Obsolete.
PATH	Absolute path to the collection.
REGISTERED	Obsolete.
SIZE	The size of the collection, expressed in kilobytes.

You can also specify `uni` to enable support for multiple languages.

The ColdFusion Administrator Verity > Collections page displays the information that is returned when you use the `list` attribute.

If the Verity Server is not running when the `list` action is executed, the tag throws an error.

To determine whether a collection exists, use code, such as the following, to execute a query of queries:

```
<cfcollection action="list" name="myCollections" >
<cfquery name="qoq" dbtype="query">
```

```
    SELECT * from myCollections
    WHERE myCollections.name = 'myCollectionName'
    </cfquery>
<cfif qoq.recordcount GT 0>
    <!--- Collection exists --->
    <cfdump var = #qoq#>
</cfif>
```

To get a result set with values for all the collections that are registered with the Verity server, use code such as the following:

```
<cfcollection action="list" name="myCollections">
<cfoutput query="myCollections">
    #name#<br>
</cfoutput>
```

To add content to a collection, use `cfindex`. To search a collection, use `cfsearch`.

You cannot delete Verity collections on Windows if they are created outside of the ColdFusion collections directory or on a drive other than C:, D: or E:. To use a different drive letter, edit the cf_dir/verity/common/verity.cfg file and replace an entry with the directory you wish to use as follows:

```
alias11=path6
mapping11=F:\
dirmode11=rw
```

Restart the ColdFusion Search Service for this change to take affect.

The `language` attribute of this tag supports the following options:

Asian Language Pack			
Japanese	Korean	Chinese	Traditional Chinese
Multilanguage Language Pack			
Unicode			
Western European Language Pack			
Bokmal	Finnish	Italian	Spanish
Danish	French	Nynorsk	Swedish
Dutch	German	Portuguese	
Eastern European/Middle Eastern Language Pack			
Arabic	Greek	Polish	Turkish
Bulgarian	Hebrew	Russian	
Czech	Hungarian	Russian2	

The default location of Verity collections is as follows:

- Server configuration:
  - Windows: C:\CFusionMX7\verity\collections
  - UNIX system: /opt/coldfusionmx7/verity/collections
- J2EE configuration: *webapp_root*/WEB-INF/cfusion/verity/collections

Example

```
<!---------------------------------------------------------------------------
(coll_actn.cfm)
Check for server platform and use its default Verity Collection directory.
If you did not install ColdFusion in the default directory, or if you use
the J2EE configuration, or if your webroot is not C:\CFusionMX7\wwwroot, you
might need to change the path in this example. For example, for JRun4 the path
might be C:\JRun4\Verity\Collections\
---------------------------------------------------------------------------->
<cfif Find("Windows", Server.OS.Name)>
    <cfset collPath = "C:\JRun4\Verity\Collections\">
<cfelse>
    <cfset collpath = "/opt/coldfusionmx7/verity/collections/">
</cfif>

<!---------------------------------------------------------------------------
Process form input and do the requested cfcollection operation.
---------------------------------------------------------------------------->

<cfif IsDefined("form.CollectionName") AND IsDefined("form.CollectionAction")>
    <cfif form.CollectionName is not "">
        <cfswitch expression="#FORM.CollectionAction#">
        <cfcase value="Create">
        <cfcollection action="CREATE" collection="#FORM.CollectionName#"
                    path="#collPath#" categories="yes">
        <h3>Collection created.<br>
            Use CFINDEX to populate it.</h3>
        </cfcase>
        <cfcase value="Repair">
        <cfcollection action="REPAIR" collection="#FORM.CollectionName#">
        <h3>Collection repaired.</h3>
        </cfcase>
        <cfcase value="Optimize">
        <cfcollection action="OPTIMIZE" collection="#FORM.CollectionName#">
        <h3>Collection optimized.</h3>
        </cfcase>
        <cfcase value="Delete">
        <cfcollection action="DELETE" collection="#FORM.CollectionName#">
         <h3>Collection deleted.</h3>
        </cfcase>
        </cfswitch>
    <cfelse>
    <h3>Please enter a name for your collection</h3>
    </cfif>
</cfif>

<!---------------------------------------------------------------------
(coll_form.cfm)
Form to specify the collection name and action
    coll_form.cfm
---------------------------------------------------------------------->

<form action="coll_actn.cfm" method="POST" >
<select name="CollectionAction">
    <option value="Create">Create this collection
    <option value="Optimize">Optimize this collection
    <option value="Repair">Repair this collection
    <option value="Delete">Delete this collection
</select>

<p><strong>Collection on which to act</strong><br>
```

```
Use the default value or enter your own Collection name<br>
<input type="Text" name="CollectionName" value="My_coll"></p>

<input type="Submit" name="" value="alter or create my collection">
</form>
```

cfcomponent

Description

Creates and defines a component object; encloses functionality that you build in CFML and enclose in cffunction tags. This tag contains one or more cffunction tags that define methods. Code within the body of this tag, other than cffunction tags, is executed when the component is instantiated.

A component file has the extension CFC and is stored in any directory of an application.

A component method is invoked in the following ways:

- In the cfinvoke tag in a ColdFusion page
- In a URL that calls a CFC file and passes a method name as a URL parameter
- In the cfscript tag
- As a web service
- From Flash code

Category

Extensibility tags

Syntax

```
<cfcomponent
    bindingname = "binding element name"
    displayname = "string"
    extends = "component name"
    hint = "string"
    implements = "ColdFusion interface"
    namespace = "default service namespace"
    output = "no value|no|yes"
    porttypename = "port type element name"
    serviceaddress = "service URL"
    serviceportname = "port element name"
    style = "rpc|document"
    wsdlfile = "path">
    variable declarations
    <cffunction ...>
        ...
    </cffunction>

    <cffunction ...>
        ...
    </cffunction>
</cfcomponent>
```

See also

cfargument, cffunction, cfinterface, cfinvoke, cfinvokeargument, cfobject, cfproperty, cfreturn, IsInstanceOf, "Building and Using ColdFusion Components" on page 158 in the *ColdFusion Developer's Guide*

History

ColdFusion 8:

- Added the implements and serviceaddress attributes.

- Added support for the `onMissingMethod` function.

ColdFusion MX 7:

- Added support for publishing document-literal style web services.
- Added the `style`, `namespace`, `serviceportname`, `porttypename`, `wsdlfile`, `bindingname`, and `output` attributes.
- Extended functionality for the `hint` and `displayname` attributes when publishing document-literal style web services.

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`bindingname`	Optional		Specifies the `binding` attribute of the `port` element in the WSDL. If you don't specify this attribute, ColdFusion derives the value from the CFC class name.
`displayname`	Optional		A string that displays when you use introspection to show information about the CFC. The information appears on the heading, following the component name.
`extends`	Optional	WEB-INF.cftags.component	Name of parent component from which to inherit methods and properties. You can use the keyword `component` to specify the default value.
`hint`	Optional		Text that displays when you use introspection to show information about the CFC. The `hint` attribute value appears below the component name heading. Use this attribute to describe the purpose of the parameter.
`implements`	Optional		Name of the ColdFusion interface or interfaces that this component implements. If the component implements an interface, it must define all the functions in the interface, and the function definitions must conform to the definitions specified in the interface. For more information, see `cfinterface`. A component can implement any number of interfaces. To specify multiple interfaces, use a comma-delimited list with the format `interface1,interface2`.
`namespace`	Optional	class name	Specifies the namespace used in the WSDL for a CFC that is invoked as a web service. If you don't specify this attribute, ColdFusion MX derives the value from the CFC class name.
`output`	Optional	Component body displayable text that is processed as standard CFML	Specifies whether constructor code in the component can generate HTML output; does not affect output in the body of `cffunction` tags in the component . • `yes`: Constructor code is processed as if it were within a `cfoutput` tag. Variable names surrounded by number signs (#) are automatically replaced with their values. • `no`: Constructor code is processed as if it were within a `cfsilent` tag. • If you do not specify this attribute, constructor code is processed as standard CFML. Any variables must be in `cfoutput` tags.
`porttypename`	Optional		Specifies the `name` attribute of the `porttype` element in the WSDL. If you don't specify this attribute, ColdFusion MX derives the value from the CFC class name.

Attribute	Req/Opt	Default	Description
`serviceaddress`	Optional	URL of the CFC	Specifies the SOAP URL of the web service. If you don't specify this attribute, ColdFusion MX uses the URL of the CFC in the WSDL service description. Use this attribute to specify the protocol, for example, by specifying a URL that starts with https://. This attribute applies only for web services.
`serviceportname`	Optional		Specifies the `name` attribute of the `port` element in the WSDL. If you don't specify this attribute, ColdFusion MX derives the value from the CFC class name.
`style`	Optional	`rpc`	Specifies whether a CFC used for web services uses RPC-encoded style or document-literal style: • `rpc`: RPC-encoded style • `document`: Document-literal style
`wsdlfile`	Optional		A properly formatted WSDL file to be used instead of WSDL generated by ColdFusion.

Usage

If you specify the `extends` attribute, the data and methods of the parent component are available to CFC methods as if they were parts of the current component. If the managerCFC component extends the employeeCFC component, and the employeeCFC component has a `getEmployeeName` method, you can call this method by using the managerCFC, as follows:

```
<cfinvoke component="managerCFC" method="getEmployeeName" returnVariable="managerName"
    EmployeeID=#EmpID#>
```

This tag requires an end tag.

If you specify `style="document"`, ColdFusion publishes the CFC as a document-literal style web service. For more information, see "Publishing document-literal style web services" on page 918 in the *ColdFusion Developer's Guide*.

CFCs support an `onMissingMethod` function. By defining an `onMissingMethod` function in the `cfcomponent` tag body in the CFC, you can handle calls to methods that are not implemented in the CFC. If an application calls a function that is not defined in the CFC, ColdFusion calls the `onMissingMethod` function and passes it the requested method's name and arguments. If you do not define an `onMissingMethod` function, a call to a method that is not defined in the CFC causes ColdFusion to throw an error that must be handled in the calling code.

The `onMissingMethod` function is useful for several purposes:

- To handle errors directly in the component, instead of requiring that each instance of code that calls the component handles them.
- To create a dynamic proxy, an object that can take arbitrary calls and dynamically determines the correct action.

The `onMissingMethod` function must have the following format:

```
<cffunction name="onMissingMethod">
    <cfargument name="missingMethodName" type="string">
    <cfargument name="missingMethodNameArguments" type="struct">
        code to handle call to nonexistent method
</cffunction>
```

Example

```
<cfcomponent>
    <cffunction name="getEmp">
         <cfquery name="empQuery" datasource="cfdocexamples" >
            SELECT FIRSTNAME, LASTNAME, EMAIL
```

```
            FROM tblEmployees
         </cfquery>
         <cfreturn empQuery>
    </cffunction>

    <cffunction name="getDept">
        <cfquery name="deptQuery" datasource="cfdocexamples" >
            SELECT *
            FROM tblDepartments
         </cfquery>
         <cfreturn deptQuery>
    </cffunction>
</cfcomponent>
```

cfcontent

Description

Does either or both of the following:

- Sets the MIME content encoding header for the current page; if the encoding information includes a character encoding, sets the character encoding of generated output.
- Sends the contents of a file, or of a variable that contains binary data, as the page output.

To restrict this tag, use the settings in the ColdFusion Administrator > Security > Sandbox Security. For more information, see the Administrator online Help.

Category

Data output tags

Syntax

```
<cfcontent
    deleteFile = "yes|no"
    file = "filename"
    reset = "yes|no"
    type = "file type"
    variable = "variable name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcol`, `cfheader`, `cfhttp`, `cfoutput`, `cftable`

History

ColdFusion 8: Changed the behavior of the tag if the `type` attribute is not specified and the `file` attribute is specified. Previously, ColdFusion assumed a default file type of text/html. Now, ColdFusion attempts to get the content type from the file.

ColdFusion MX 7: Added the `variable` attribute.

Attributes

Attribute	Req/Opt	Default	Description
`deleteFile`	Optional	`no`	Applies only if you specify a file with the `file` attribute. • `yes`: deletes the file on the server after sending its contents to the client. • `no`: leaves the file on the server.
`file`	Optional		Name of a file whose contents provide the page output. The filename must start with a drive letter and a colon, or a forward or backward slash. When using ColdFusion in a distributed configuration, the `file` attribute must refer to a path on the system on which the web server runs. When you use this attribute, any other output on the current CFML page is ignored; only the contents of the file are sent to the client.
`reset`	Optional	`yes`	If you specify a `file` or `variable` attribute, this attribute has no effect; otherwise, it does the following: • `yes`: discards output that precedes call to `cfcontent` • `no`: preserves output that precedes call to `cfcontent`. In this case, all output is sent with the specified type.

Attribute	Req/Opt	Default	Description
type	Optional		The MIME content type of the page, optionally followed by a semicolon and the character encoding. By default, ColdFusion sends pages as text/html content type in the UTF-8 character encoding. However, if the file attribute is specified, ColdFusion attempts to get the content type from the file. The content type determines how the browser or client interprets the page contents. The following are some of the content type values that you can use: • text/html • text/plain • application/x-shockwave-flash • application/msword • image/jpeg The following list includes commonly used character encoding values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For example: type = "text/html" type = "text/html; charset=ISO-8859-1"
variable	Optional		Name of a ColdFusion binary variable whose contents can be displayed by the browser, such as the contents of a chart generated by the cfchart tag or a PDF or Excel file retrieved by a cffile action="readBinary" tag. When you use this attribute, any other output on the current CFML page is ignored; only the contents of the file are sent to the client.

Usage

To set the character encoding (character set) of generated output, including the page HTML, use code such as the following:

```
<cfcontent type="text/html; charset=ISO-8859-1">
```

When ColdFusion processes an HTTP request, it determines the character encoding to use for the data it returns in the HTTP response. By default, ColdFusion returns character data using the Unicode UTF-8 format, regardless of the value of an HTML meta tag in the page. You can use the cfcontent tag to override the default character encoding of the response. For example, to tell ColdFusion to return the page using Japanese EUC character encoding, use the type attribute, as follows:

```
<cfcontent type="text/html; charset=EUC-JP">
```

If you call the cfcontent tag from a custom tag, and you do not want the tag to discard the current page when it is called from another application or custom tag, set reset = "no".

If a file delete operation is unsuccessful, ColdFusion throws an error.

Do not use this tag after the cfflush tag on a page, it has no effect or ColdFusion throws an error.

The following tag can force most browsers to display a dialog box that asks users whether they want to save the contents of the file specified by the cfcontent tag using the filename specified by the filename value. If the user selects to open the file, most browsers open the file in the related application, not the browser window.

```
<cfheader name="Content-Disposition" value="attachment; filename=filename.ext">
```

Some file types, such as PDF documents, do not use executable code and can display directly in most browsers. To request the browser to display the file directly, use a cfheader tag similar to the following:

```
<cfheader name="Content-Disposition" value="inline; filename=name.ext">
```

You can use any value for the *filename* part of the filename attribute, but the *ext* part must be the standard Windows extension for the file type.

For file types that might contain executable code, such as Microsoft Excel documents, most browsers always ask before opening the document. For these file types, the inline content disposition specification requests the browser to display the file directly if the user selects to open the file.

For more information on character encodings, see the following web pages:

- The page at www.w3.org/International/O-charset.html provides general information on character encodings and the web, and has several useful links.
- The page at www.iana.org/assignments/character-sets is a complete list of character sets names used on the Internet, maintained by the Internet Assigned Numbers Authority.
- The page at java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html lists the character encodings that Java, and therefore ColdFusion, can interpret. This list uses Java internal names, not the IANA character encoding names that you use in the SetEncoding charset parameter and other ColdFusion attributes and parameters. ColdFusion MX 6.0 Updater 3 uses JDK 1.3. ColdFusion MX 6.1 uses JDK 1.4.2; for encoding support, see http://java.sun.com/j2se/1.4.2/docs/guide/intl/encoding.doc.html.

For a complete list of media types used on the Internet, see www.iana.org/assignments/media-types/.

Example

```
<!--- CFCONTENT Example 1
This example shows the use of cfcontent to return the contents of the CF
Documentation page dynamically to the browser. You might need to change the
path and/or drive letter depending on how ColdFusion is installed on your
system. Notice that the graphics do not display and the hyperlinks do not work,
because the html page uses relative filename references.
The root of the reference is the ColdFusion page, not the location of the
html page. --->

<cfcontent type = "text/html"
    file = "C:\CFusionMX7\wwwroot\cfdocs\dochome.htm"
    deleteFile = "no">

<!--- CFCONTENT Example 2
This example shows how the Reset attribute changes text output. Notice how the
first text section ("This example shows how the Reset attribute changes output
for text reset = "Yes":123) does NOT print out to the screen. --->
```

```
<p>This example shows how the Reset attribute changes output for text.</p>
<p>reset = "Yes": 123 <BR> <cfcontent type = "text/html" reset = "Yes">456</p>
<p>This example shows how the Reset attribute changes output for text.</p>
<p>reset = "No": 123 <BR> <cfcontent type = "text/html" reset = "No">456</p>
<!--- CFCONTENT Example 3
This example triggers a download of an Excel file. The user is prompted with an option to
save the file or open it in the browser. --->

<cfheader name="Content-Disposition" value="inline; filename=acmesales03.xls">
    <cfcontent type="application/vnd.ms-excel" file="c:\temp\acmesales03.xls">

<!--- CFCONTENT Example 4
This example triggers a download of a Word document then deletes the original from the "temp"
directory. The user is prompted with an option to save the file or open it in the browser.
--->

<cfheader name="Content-Disposition" value="inline; filename=temp.doc">
<cfcontent type="application/msword" file="c:\temp\Cable.doc" deletefile="yes">

<!--- CFCONTENT Example 5
This example causes the browser to treat the HTML table as Excel data.
Excel interprets the table format.
Because Excel can include executable code, the browser prompts the user whether
to save the file or open it in a browser. --->

<cfheader name="Content-Disposition" value="inline; filename=acmesalesQ1.xls">
<cfcontent type="application/vnd.msexcel">

<table border="2">
<tr><td>Month</td><td>Quantity</td><td>$ Sales</td></tr>
<tr><td>January</td><td>80</td><td >$245</td></tr>
<tr><td>February</td><td>100</td><td>$699</td></tr>
<tr><td>March</td><td>230</td><td >$2036</td></tr>
<tr><td>Total</td><td>=Sum(B2..B4)</td><td>=Sum(C2..C4)</td></tr>
</table>
```

cfcookie

Description

Defines web browser cookie variables, including expiration and security options.

Category

Forms tags, Variable manipulation tags

Syntax

```
<cfcookie
    name = "cookie name"
    domain = ".domain"
    expires = "period"
    path = "URL"
    secure = "yes|no"
    value = "text">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdump`, `cfparam`, `cfregistry`, `cfsavecontent`, `cfschedule`, `cfset`

History

ColdFusion MX 6.1:

- Changed the `expires` attribute: it now accepts a date time object.
- Cookie names can include all ASCII characters except commas, semicolons, or whitespace characters.

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		Name of cookie variable. ColdFusion converts cookie names to all-uppercase. Cookie names set using this tag can include any printable ASCII characters except commas, semicolons or white space characters.
`domain`	Required if `path` attribute is specified. Optional otherwise		Domain in which cookie is valid and to which cookie content can be sent from the user's system. By default, the cookie is only available to the server that set it. Use this attribute to make the cookie available to other servers. Must start with a period. If the value is a subdomain, the valid domain is all domain names that end with this string. This attribute sets the available subdomains on the site on which the cookie can be used. For a `domain` value that ends in a country code, the specification must contain at least three periods; for example, `".mongo.state.us"`. For top-level domains, two periods are required; for example, `".mgm.com"`. You cannot use an IP address as a domain.

Attribute	Req/Opt	Default	Description
expires	Optional	session only	Expiration of cookie variable. • The cookie expires when the user closes the browser, that is, the cookie is "session only". • A date or date/time object (for example, 10/09/97). • A number of days (for example, 10, or 100). • now: deletes cookie from client cookie.txt file (but does not delete the corresponding variable the Cookie scope of the active page). • never: The cookie expires in 30 years from the time it was created (effectively never in web years).
path	Optional		URL, within a domain, to which the cookie applies; typically a directory. Only pages in this path can use the cookie. By default, all pages on the server that set the cookie can access the cookie. path = "/services/login" To specify multiple URLs, use multiple cfcookie tags. If you specify path, you must also specify domain.
secure	Optional		If browser does not support Secure Sockets Layer (SSL) security, the cookie is not sent. To use the cookie, the page must be accessed using the https protocol. • yes: Variable must be transmitted securely. • no
value	Optional		Value to assign to cookie variable. Must be a string or variable that can be stored as a string.

Usage

If this tag specifies that a cookie is saved beyond the current browser session, the client browser writes or updates the cookie in its local cookies file. Until the browser is closed, the cookie resides in browser memory. If the expires attribute is not specified, the cookie is not written to the browser cookies file.

If you use this tag after the cfflush tag on a page, ColdFusion does not send the cookie to the browser; however, the value you set is available to ColdFusion in the Cookie scope during the browser session.

Note: *You can also create a cookie that expires when the current browser session expires by using the cfset tag or a CFScript assignment statement to set a variable in the Cookie scope, as in <cfset Cookie.mycookie="sugar">. To get a cookie's value, refer to the cookie name in the Cookie scope, as in <cfif Cookie.mycookie is "oatmeal">.*

You can use dots in cookie names, as the following examples show:

```
<cfcookie name="person.name" value="wilson, john">
<cfset cookie.person.lastname="Santiago">
```

To access cookies, including cookies that you set and all cookies that are sent by the client, use the Cookie scope. For example, to display the value of the person.name cookie set in the preceding code, use the following line:

```
<cfoutput>#cookie.person.name#</cfoutput>
```

Example

```
<!--- This example shows how to set/delete a cfcookie variable. --->
<!--- Select users who have entered comments into a sample database. --->
<cfquery name = "GetAolUser" dataSource = "cfdocexamples">
    SELECT EMail, FromUser, Subject, Posted
    FROM Comments
</cfquery>
<html>
<body>
<h3>cfcookie Example</h3>
```

```
<!--- If the URL variable delcookie exists, set cookie expiration date
    to NOW --->
<cfif IsDefined("url.delcookie") is True>
    <cfcookie name = "TimeVisited"
    value = "#Now()#"
    expires = "NOW">
<cfelse>
<!--- Otherwise, loop through list of visitors; stop when you match
    the string aol.com in a visitor's e-mail address. --->
<cfloop query = "GetAolUser">
    <cfif FindNoCase("aol.com", Email, 1) is not 0>
        <cfcookie name = "LastAOLVisitor"
        value = "#Email#"
        expires = "NOW" >
    </cfif>
</cfloop>
<!--- If the timeVisited cookie is not set, set a value. --->
    <cfif IsDefined("Cookie.TimeVisited") is False>
        <cfcookie name = "TimeVisited"
        value = "#Now()#"
        expires = "10">
    </cfif>
</cfif>
<!--- Show the most recent cookie set. --->
<cfif IsDefined("Cookie.LastAOLVisitor") is "True">
    <p>The last AOL visitor to view this site was
    <cfoutput>#Cookie.LastAOLVisitor#</cfoutput>, on
    <cfoutput>#DateFormat(COOKIE.TimeVisited)#</cfoutput>
<!--- Use this link to reset the cookies. --->
<p><a href = "cfcookie.cfm?delcookie = yes">Hide my tracks</A>
<cfelse>
    <p>No AOL Visitors have viewed the site lately.
</cfif>
```

cfdbinfo

Description

Lets you retrieve information about a data source, including details about the database, tables, queries, procedures, foreign keys, indexes, and version information about the database, driver, and JDBC.

Category

Database manipulation tags

Syntax

```
<cfdbinfo
    datasource="data source name"
    name="result name"
    type="dbnames|tables|columns|version|procedures|foreignkeys|index"
    dbname="database name"
    password="password"
    pattern="filter pattern"
    table="table name"
    username="username">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfinsert`, `cfprocparam`, `cfprocresult`, `cfqueryparam`, `cfstoredproc`, `cftransaction`, `cfupdate`;
"Optimizing database use" on page 243 in the *ColdFusion Developer's Guide*.

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`datasource`	Required		Datasource to use to connect to the database.
`name`	Required		Name to use to refer to the result.
`type`	Required		Type of information to get: • `dbnames`: database name and type • `tables`: name, type, and remarks • `columns`: name, SQL data type, size, decimal precision, default value, maximum length in bytes of a character or integer data type column, whether nulls are allowed, ordinal position, remarks, whether the column is a primary key, whether the column is a foreign key, the table that the foreign key refers to, the key name the foreign key refers to • `version`: database product name and version, driver name and version, JDBC major and minor version • `procedures`: name, type, and remarks • `foreignkeys`: foreign key name and table, primary key name, delete and update rules • `index`: name, column on which the index is applied, ordinal position, cardinality, whether the row represents a table statistic or an index, number of pages used by the table or index, whether the index values are unique

Attribute	Req/Opt	Default	Description
dbname	Optional		Name of the database. Used only if the action = "This overrides the one mentioned as a part of datasource definition."
password	Optional		Password to connect to the database.
pattern	Optional		Used only if type = "tables", type = "columns", or type = "procedures". Specifies a filter to retrieve information about specific tables, columns, or stored procedures. Use an underline (_) to represent a single wildcard character and a percent sign (%) to represent a wildcard of zero or more characters.
table	Required if type = "columns" or type = "foreignkeys" or type = "index"		Name of the table from which you retrieve information.
username	Optional	no	User name to connect to the database.

Usage

Use the cfdbinfo tag to return a query object that contains information about a database. The query object varies, depending on the value that you specify in the type attribute. The following table lists the query object contents for each type:

Type	Column name	Description
dbnames	DATABASE_NAME	Name of the database.
	TYPE	Type of the database, whether schema or catalog.
tables	TABLE_NAME	Name of the table.
	TABLE_TYPE	Type of the table, including view, table, synonym, and system table.
	REMARKS	Remarks of the table.
columns	COLUMN_NAME	Name of the column.
	TYPE_NAME	SQL data type of the column.
	IS_NULLABLE	Whether the column allows nulls.
	IS_PRIMARYKEY	Whether the column is a primary key.
	IS_FOREIGNKEY	Whether the column is a foreign key.
	REFERENCED_PRIMARYKEY	If the column is a foreign key, the name of the table it refers to.
	REFERENCED_PRIMARYKEY_TABLE	If the column is a foreign key, the key name it refers to.
	COLUMN_SIZE	Size of the column
	DECIMAL_DIGITS	Number of digits to the right of the decimal point.
	COLUMN_DEFAULT_VALUE	Default value of column.
	CHAR_OCTET_LENGTH	Maximum length in bytes of a character or integer data type column.
	ORDINAL_POSITION	Ordinal position of the column.
	REMARKS	Remarks of the column.

Type	Column name	Description
version	DATABASE_VERSION	Version of the database management system.
	DATABASE_PRODUCTNAME	Name of the database management system.
	DRIVER_VERSION	Version of the database driver.
	DRIVER_NAME	Name of the database driver.
	JDBC_MAJOR_VERSION	Major version number of the driver.
	JDBC_MINOR_VERSION	Minor version number of the driver.
procedures	PROCEDURE_NAME	Name of the stored procedure.
	REMARKS	Remarks for the stored procedure.
	PROCEDURE_TYPE	Procedure type, which indicates whether the procedure returns a result.
foreignkeys	FKCOLUMN_NAME	Foreign key name.
	FKTABLE_NAME	Foreign key table name.
	PKCOLUMN_NAME	Primary key name.
	DELETE_RULE	Specifies what action to take when you delete a record that has dependent records.
	UPDATE_RULE	Specifies what action to take when you update a record that has dependent records.
index	INDEX_NAME	Name of the index, empty if type is table statistic.
	COLUMN_NAME	Name of the column on which the index is applied, empty if the type is table statistic.
	ORDINAL_POSITION	Ordinal position.
	CARDINALITY	Number of unique values if the type is index, or number of rows if the type is statistic
	TYPE	Whether the row represents a table statistic or an index. Index types are clustered, hashed, or other.
	PAGES	Number of pages used by the table if the type is table statistic, or the number of pages used by the index.
	NON_UNIQUE	Whether the index values are unique.

Example

```
<cfset datasrc = "oratest">

<cfdbinfo
    type="dbnames"
    datasource="#datasrc#"
    name="dbdata">

<cfoutput>
The #datasrc# data source has the following databases:<br />
</cfoutput>
<table border="1">
<tr>
    <th valign="top" align="left">Database name</th><th>Type</th>
</tr>
    <cfoutput query="dbdata">
    <tr>
        <td>#dbdata.DATABASE_NAME#</td><td>#dbdata.TYPE#</td>
    </tr>
    </cfoutput>
```

```
</table>
```

cfdefaultcase

Description

Used only inside the cfswitch tag body. Contains code to execute when the expression specified in the cfswitch tag does not match the value specified by a cfcase tag.

Category

Flow-control tags

Syntax

```
<cfdefaultcase>
```

See also

cfcase, cfswitch; "cfswitch, cfcase, and cfdefaultcase" on page 18 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Changed placement requirements: this tag does not have to follow all cfcase tags in the cfswitch tag body.

Usage

The contents of the cfdefaultcase tag body executes if the expression attribute of the cfswitch tag does not match any of the values specified by the cfcase tags in the cfswitch tag body. The contents of the cfdefaultcase tag body can include HTML and text, and CFML tags, functions, variables, and expressions.

You can specify only one cfdefaultcase tag within a cfswitch tag. You can put the cfdefaultcase tag at any position within a cfswitch statement; it is not required to be the last item, but it is good programming practice to put it last.

Example

```
<!--- The following example displays a grade based on a 1-10 score.
        Several of the cfcase tags match more than one score.
        For simplicity, the example sets the score to 7. --->
<cfset score="7">
<cfswitch expression="#score#">
    <cfcase value="10">
        <cfset grade="A">
    </cfcase>
    <cfcase value="9;8" delimiters=";">
        <cfset grade="B">
    </cfcase>
    <cfcase value="7;6" delimiters=";">
        <cfset grade="C">
    </cfcase>
    <cfcase value="5;4;" delimiters=";">
        <cfset grade="D">
    </cfcase>
    <cfdefaultcase>
        <cfset grade="F">
    </cfdefaultcase>
</cfswitch>
<cfoutput>
    Your grade is #grade#
</cfoutput>
```

cfdirectory

Description

Manages interactions with directories.

Category

File management tags

Syntax

```
<cfdirectory
    directory = "directory name"
    action = "list|create|delete|rename"
    filter = "list filter"
    listInfo = "name|all"
    mode = "permission"
    name = "query name"
    newDirectory = "new directory name"
    recurse = "yes|no"
    sort = "sort specification"
    type = "file|dir|all">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cffile`

History

ColdFusion 8: Added the `listinfo` and `type` attributes.

ColdFusion MX 7: Added the `recurse` attribute (named `recursive` in Alpha 1) and `directory` result-set column.

ColdFusion MX:

- Changed behavior for `action = "list"`:
  - On Windows, `cfdirectory` `action = "list"` no longer returns the directory entries `"."` (dot) or `".."` (dot dot), which represent "the current directory" and "the parent directory."
  - On Windows, `cfdirectory` `action = "list"` no longer returns the values of the Archive and System attributes.
  - On UNIX and Linux, `cfdirectory` `action = "list"` does not return any information in the `mode` column.

Attributes

Attribute	Req/Opt	Default	Description
`directory`	Required		Absolute pathname of directory against which to perform action. You can use an IP address, as in the following example: `<cfdirectory directory="//12.3.123.123/c_drive/" name="dirQuery" action="LIST">`
`action`	Optional	`list`	• `list`: returns a query record set of the files in the specified directory. The directory entries "." (dot) and ".." (dot dot), which represent the current directory and the parent directory, are not returned. • `create` • `delete` • `rename`
`filter`	Optional if `action = "list"`		File extension filter applied to returned names, for example, *.cfm. One filter can be applied.
`listinfo`	Optional	all	• `all`: includes all information in the result set. • `name`: includes only filenames in the result set.
`mode`	Optional		Used with `action = "create"`. Permissions. Applies only to UNIX and Linux. Octal values of `chmod` command. Assigned to owner, group, and other, respectively, for example: • 644: assigns read/write permission to owner; read permission to group and other. • 777: assigns read/write/execute permission to all.
`name`	Required if `action = "list"`		Name for output record set.
`newDirectory`	Required if `action = "rename"`		New name for directory.
`recurse`	Optional	`no`	Whether ColdFusion performs the action on subdirectories: • `yes` • `no` Valid for `action="list"` and `action="delete"`.
`sort`	Optional; used if `action = "list"`	ASC	Query columns by which to sort a directory listing. Delimited list of columns from query output. To qualify a column, use one of the following values: • `asc`: ascending (a to z) sort order. • `desc`: descending (z to a) sort order. For example: `sort = "directory ASC, size DESC, datelastmodified"`
`type`	Optional	all	• `file`: includes only filenames. • `dir`: includes only directory names. • `all`: includes both filenames and directory names.

Usage

If you put ColdFusion applications on a server that is used by multiple customers, you must consider the security of files and directories that could be uploaded or otherwise manipulated with this tag by unauthorized users. For more information about securing ColdFusion tags, see *Configuring and Administering ColdFusion*.

If `action = "list"`, `cfdirectory` returns the following result columns, which you can reference in a `cfoutput` tag:

- `name`: Directory entry name. The entries `"."` and `".."` are not returned.
- `directory`: Directory that contains the entry.
- `size`: Directory entry size.
- `type`: File type: `file`, for a file; `dir`, for a directory.
- `dateLastModified`: The date that an entry was last modified.
- `attributes`: File attributes, if applicable.
- `mode`: Empty column; retained for backward compatibility with ColdFusion 5 applications on UNIX.

Use the following result columns in standard CFML expressions, preceding the result column name with the query name:

```
#mydirectory.name#
#mydirectory.directory#
#mydirectory.size#
#mydirectory.type#
#mydirectory.dateLastModified#
#mydirectory.attributes#
#mydirectory.mode#
```

Note: *If the `cfdirectory` tag does not appear to work, for example, if a list operation returns an empty result set, make sure that you have correct permissions to access the directory. For example, if you run ColdFusion as a service on Windows, it operates by default as System, and cannot access directories on a remote system or mapped drive; to resolve this issue, do not run ColdFusion using the local system account.*

The `filter` attribute specifies a pattern of one or more characters. All names that match that pattern are included in the list. On Windows systems, pattern matching ignores text case, on UNIX and Linux, pattern matches are case-sensitive.

The following two characters have special meaning in the pattern and are called metacharacters:

- The asterisk (*) matches any zero or more characters.
- The question mark (?) matches any single character.

The following table shows examples of patterns and filenames that they match:

Pattern	Matches
foo.*	Any file called foo with any extension; for example, foo.html, foo.cfm, and foo.xml.
*.html	All files with the suffix .html, but not files with the suffix .htm.
??	All files with two-character names.

Example

```
<!--- EXAMPLE 1: Creating and Renaming
Check that the directory exists to avoid getting a ColdFusion error message. --->
<cfset newDirectory = "otherNewDir">
```

```
<cfset currentDirectory = GetDirectoryFromPath(GetTemplatePath()) & "newDir">
<!--- Check whether the directory exists. --->
<cfif DirectoryExists(currentDirectory)>
<!--- If yes, rename the directory. --->
    <cfdirectory action = "rename" directory = "#currentDirectory#"
        newDirectory = "#newDirectory#" >
    <cfoutput>
    <p>The directory existed and the name has been changed to: #newDirectory#</p>
    </cfoutput>
<cfelse>
    <!--- If no, create the directory. --->
    <cfdirectory action = "create" directory = "#currentDirectory#" >
    <cfoutput><p>Your directory has been created.</p></cfoutput>
</cfif>

<!--- EXAMPLE 2: Deleting a directory
Check that the directory exists and that files are not in the directory to avoid getting
ColdFusion error messages. --->

<cfset currentDirectory = GetDirectoryFromPath(GetTemplatePath()) & "otherNewDir">
<!--- Check whether the directory exists. --->
<cfif DirectoryExists(currentDirectory)>
    <!--- If yes, check whether there are files in the directory before deleting. --->
    <cfdirectory action="list" directory="#currentDirectory#"
    name="myDirectory">
    <cfif myDirectory.recordcount gt 0>
    <!--- If yes, delete the files from the directory. --->
        <cfoutput>
        <p>Files exist in this directory. Either delete the files or code
            something to do so.</P>
        </cfoutput>
    <cfelse>
    <!--- Directory is empty - just delete the directory. --->
        <cfdirectory action = "delete" directory = "#currentDirectory#">
        <cfoutput>
        <p>The directory existed and has been deleted.</P>
        </cfoutput>
    </cfif>
<cfelse>
    <!--- If no, post message or do some other function. --->
    <cfoutput><p>The directory did NOT exist.</p></cfoutput>
</cfif>
<!---EXAMPLE 3: List directories
The following example creates both an array of directory names and a query that contains
entries for the directories only. --->

<cfdirectory directory="C:/temp" name="dirQuery" action="LIST">

<!--- Get an array of directory names. --->
<cfset dirsArray=arraynew(1)>
<cfset i=1>
<cfloop query="dirQuery">
<cfif dirQuery.type IS "dir">
    <cfset dirsArray[i]=dirQuery.name>
    <cfset i = i + 1>
</cfif>
</cfloop>
<cfdump var="#dirsArray#">
<br>
<!--- Get all directory information in a query of queries.--->
<cfquery dbtype="query" name="dirsOnly">
```

```
SELECT * FROM dirQuery
WHERE TYPE='Dir'
</cfquery>
<cfdump var="#dirsOnly#">
```

cfdiv

Description

Creates an HTML `div` tag or other HTML container tag and lets you use asynchronous form submission or a bind expression to dynamically control the tag contents.

Category

Display management tags

Syntax

```
<cfdiv
    bind = "bind expression"
    bindOnLoad = "true|false"
    ID = "HTML tag ID"
    onBindError = "JavaScript function name"
    tagName = "HTML tag name"
/>

OR

<cfdiv
    ID = "HTML tag ID"
    tagName = "HTML tag name">
        tag body contents
</cfdiv>
```

If the tag does not have a body and end tag, you must close it with /> character combination.

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.*

See also

`cfajaximport`, `cflayout`, `cfpod`, `cfwindow`

History

ColdFusion 8: Added this tag

Attributes

The following table lists attributes that ColdFusion uses directly. The tag passes any other attributes that you specify directly as tag attributes to the generated HTML tag.

Attribute	Req/Opt	Default	Description
`bind`	Optional		A bind expression that returns the container contents. If you specify this attribute the `cfdiv` tag cannot have a body. **Note:** If a CFML page specified in this attribute contains tags that use AJAX features, such as `cfform`, `cfgrid`, and `cfwindow`, you must use a `cfajaximport` tag on the page with the `cfdiv` tag. For more information, see `cfajaximport`.
`bindOnLoad`	Optional	`true`	• *`true`*: executes the `bind` attribute expression when first loading the tag. • `false`: does not execute the `bind` attribute expression until the first bound event . To use this attribute, you must also specify a `bind` attribute. For more information, see "Using the bindOnLoad attribute" on page 634 in "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*.

Attribute	Req/Opt	Default	Description
ID	Optional		The HTML ID attribute value to assign to the generated container tag.
onBindError	Optional	See Description	The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and have specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up window appears. To use this attribute, you must also specify a bind attribute.
tagName	Optional	DIV	The HTML container tag to create.

Usage

By default, the cfdiv tag creates a div HTML element. You can use standard HTML and CSS techniques to control the position and appearance of the element and its contents.

Use the tagName attribute to create and populate an HTML content element, such as span or b. Use the cfdiv tag to create tags that can take HTML markup content directly in the body, such as span, i, b, or p, and not for tags that cannot, such as input, option, and frameset.

If you submit a form that is inside a cfdiv tag (including in HTML returned by a bind expression), the form submits asynchronously, and the response from the form submission populates the cfdiv region.

If you specify a bind attribute, the tag dynamically populates the element using a bind expression. The bind expression can specify a CFC function, a JavaScript function, a URL, or a string that contains bind parameters. An animated icon and the text "Loading..." appears while the contents are being fetched. For detailed information on using the bind attribute and bind expressions, see “Using Ajax Data and Development Features” on page 648 in the *ColdFusion Developer’s Guide*.

Example

The following simple example shows how you can use the cfdiv tag. It uses binding to display the contents of a text input field in an HTML DIV region.

The cfdivtag.cfm file, the main application file, has the following contents.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>cfdiv Example</title>
</head>

<body>
<cfform>
    <cfinput name="tinput1" type="text">
</cfform>

<h3> using a div</h3>
<cfdiv bind="url:divsource.cfm?InputText={tinput1}" ID="theDiv"
        style="background-color:##CCffFF; color:red; height:350"/>
</body>
</html>
```

The divsource.cfm file that defines the contents of the div region has the following code:

```
<h3>Echoing main page input:</h3>
<cfoutput>
    <cfif isdefined("url.InputText") AND url.InputText NEQ "">
        #url.InputText#
```

```
    <cfelse>
        No input
    </cfif>
</cfoutput>
```

To test the code, run the cfdivtag.cfm page, enter some text, and tab out of the text box or click outside the text box. The div region appears with a light blue background and red text, and when you exit the text box, it shows the text you entered.

cfdocument

Description

Creates PDF or FlashPaper output from a text block containing CFML and HTML.

Category

Data output tags

Syntax

```
<cfdocument
    format = "PDF|FlashPaper"
    authPassword = "authentication password"
    authUser = "authentication user name"
    backgroundVisible = "yes|no"
    bookmark = "yes|no"
    encryption = "128-bit|40-bit|none"
    filename = "filename"
    fontEmbed = "yes|no"
    localUrl = "yes|no"
    marginBottom = "number"
    marginLeft = "number"
    marginRight = "number"
    marginTop = "number"
    mimeType = "text/plain|application/xml|image/jpeg|image/png|image/bmp|image/gif"
    name = "output variable name"
    orientation = "portrait|landscape"
    overwrite = "yes|no"
    ownerPassword = "password"
    pageHeight = "page height in inches"
    pageType = "page type"
    pageWidth = "page width in inches"
    permissions = "permission list"
    proxyHost = "IP address or server name for proxy host"
    proxyPassword = "password for the proxy host"
    proxyPort = "port of the proxy host"
    proxyUser = "user name for the proxy host"
    saveAsName = "PDF filename"
    scale = "percentage less than 100"
    src = "URL|pathname relative to web root"
    srcfile = "absolute pathname to a file"
    unit = "in|cm"
    userAgent = "HTTP user agent identifier"
    userPassword = "password">
        HTML and CFML code
</cfdocument>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdocumentitem`, `cfdocumentsection`, `cfformUsage`, `cfpdf`, `cfpdfform`, `cfpresentation`, `cfprint`, `cfreport`

History

ColdFusion 8: Added the following attributes and variables:

- `bookmark` attribute

- `localUrl` attribute
- Ability to embed existing PDF forms by using the `cfpdfform` tag in the `cfdocument` tag.
- ColdFusion determines the MIME type of a source file based on the source filename, if the `mimeType` attribute is not specified.
- Ability to pass a PDF variable created with the `cfdocument` tag as the source for the `cfpdf` tag.
- `authPassword`, `authUser`, `proxyHost`, `proxyPassword`, `proxyPort`, `proxyUser`, and `userAgent` attributes
- `saveAsName` attribute
- `totalsectionpagecount` and `currentsectionpagenumber` scope variables.

ColdFusion MX 7.01: Added the `src`, `srcfile`, and `mimetype` attributes.

ColdFusion MX 7: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`authPassword`	Optional		Password sent to the target URL for Basic Authentication. Combined with `username` to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
`authUser`	Optional		User name sent to the target URL for Basic Authentication. Combined with `password` to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
`backgroundVisible`	Optional	`no`	Specifies whether the background prints when the user prints the document: • `yes`: includes the background when printing. • `no`: does not includes the background when printing.
`bookmark`	Optional	`no`	Specifies whether bookmarks are created in the document: • `yes`: creates bookmarks. • `no`: does not create bookmarks.
`encryption`	Optional	`none`	(`format="PDF"` only) Specifies whether the output is encrypted: • `128-bit` • `40-bit` • `none`
`filename`	Optional		Name of a file to contain the PDF or FlashPaper output. If you omit the `filename` attribute, ColdFusion displays the output in the browser.
fontEmbed	Optional	`yes`	Specifies whether ColdFusion embeds fonts in the output: • `yes`: embeds fonts. • `no`: does not embed fonts. • `selective`: embed sall fonts except Java fonts and core fonts.
format	Required		Report format: • `PDF` • `FlashPaper`

Attribute	Req/Opt	Default	Description
localUrl	Optional	no	Specifies whether to retrieve image files directly from the local drive: • yes: ColdFusion retrieves image files directly from the local drive rather than by using HTTP, HTTPS, or proxy. • no: ColdFusion uses HTTP, HTTPS, or proxy to retrieve image files even if the files are stored locally. For more information, see "Using an image file URL" on page 125.
marginBottom	Optional		Bottom margin in inches (default) or centimeters. To specify the bottom margin in centimeters, include the unit=cm attribute.
marginLeft	Optional		Left margin in inches (default) or centimeters. To specify the left margin in centimeters, include the unit=cm attribute.
marginRight	Optional		Right margin in inches (default) or centimeters. To specify the right margin in centimeters, include the unit=cm attribute.
marginTop	Optional		Top margin in inches (default) or centimeters. To specify the top margin in centimeters, include the unit=cm attribute.
mimeType	Optional	text/html	MIME type of the source document. Supported MIME types are: • text/html • text/plain • application/xml • image/bmp • image/jpeg • image/png • image/gif If you do not specify this attribute explicitly, ColdFusion uses the filename to determine the MIME type.
name	Optional		Name of an existing variable into which the tag stores the PDF or FlashPaper output.
orientation	Optional	portrait	Page orientation: • portrait • landscape
overwrite	Optional	no	Specifies whether ColdFusion overwrites an existing file. Used in conjunction with the filename attribute.
ownerPassword	Optional		(format="PDF" only) Specifies the owner password.
pageHeight	Optional		Page height in inches (default) or centimeters. This attribute is only valid if pagetype=custom. To specify page height in centimeters, include the unit=cm attribute.

Attribute	Req/Opt	Default	Description
pageType	Optional	letter	Page type into which ColdFusion generates the report: • legal: 8.5 inches x 14 inches. • letter: 8.5 inches x 11 inches. • A4: 8.27 inches x 11.69 inches. • A5: 5.81 inches x 8.25 inches. • B4: 9.88 inches x 13.88 inches. • B5: 7 inches x 9.88 inches. • B4-JIS: 10.13 inches x 14.31 inches. • B5-JIS: 7.19 inches x 10.13 inches. • custom: custom height and width. If you specify custom, you must also specify the pageHeight and pageWidth attributes, can optionally specify margin attributes and whether the units are inches or centimeters.
pageWidth	Optional		Page width in inches (default) or centimeters. This attribute is only valid if pageType=custom. To specify page width in centimeters, include the unit=cm attribute.
permissions	Optional		(format="PDF" only) Sets one or more of the following permissions: • AllowPrinting • AllowModifyContents • AllowCopy • AllowModifyAnnotations • AllowFillIn • AllowScreenReaders • AllowAssembly • AllowDegradedPrinting Separate multiple permissions with commas.
proxyHost	Optional		Host name or IP address of a proxy server to which to send the request.
proxyPassword	Optional		Password required by the proxy server.
proxyPort	Optional	80	The port to connect to on the proxy server.
proxyUser	Optional		User name to provide to the proxy server.
scale	Optional	Calculated by ColdFusion	Scale factor as a percentage. Use this option to reduce the size of the HTML output so that it fits on that paper. Specify a number less than 100.
saveAsName	Optional		(format="PDF" only) The filename that appears in the SaveAs dialog when a user saves a PDF file written to the browser.
src	Optional		URL or the relative path to the web root. You cannot specify both the src and srcfile attributes. The file must be in a browser-writable format such as, HTML, HTM, BMP, PNG, and so on.
srcfile	Optional		Absolute path of a file that is on the server. You cannot specify both the src and srcfile attributes. The file must be in a browser-writable format such as, HTML, HTM, BMP, PNG, and so on.
unit	Optional	in	Default unit for the pageHeight, pageWidth, and margin attributes: • in: inches. • cm: centimeters.

Attribute	Req/Opt	Default	Description
userAgent	Optional	ColdFusion	Text to put in the HTTP User-Agent request header field. Used to identify the request client software.
userPassword	Optional		(format="PDF" only) Specifies a user password.

Usage

Use the cfdocument tag to render HTML and CFML output into PDF or FlashPaper format. ColdFusion does not return HTML and CFML outside of the
<cfdocument> </cfdocument> pair.

The cfdocument tag can render HTML that supports the following standards:

- HTML 4.01
- XML 1.0
- DOM Level 1 and 2
- CSS1 and CSS2 (For more information, see "Supported CSS styles" on page 124).

The cfdocument tag does not support the Internet Explorer-specific HTML generated by Microsoft Word.

You can use the src, srcfile, and mimeType attributes to create PDF or FlashPaper output from a specified file or URL. Use the src and srcfile attributes instead of using the cfhttp tag to display the result in the cfdocument tag. When you specify the src or srcfile attributes, do not include any other content inside the cfdocument tag: ColdFusion ignores the additional content.

The PDF or FlashPaper document returned by the cfdocument tag overwrites any previous HTML in the input stream and ignores any HTML after the </cfdocument> tag.

You cannot embed a cfreport tag in a cfdocument tag.

Note: *If you notice that the header text is cropped in the* cfdocument *tag output, increase the value of the* marginTop *attribute.*

Supported CSS styles

The cfdocument tag supports the following CSS styles:

background	background-attachment	background-color	background-image
background-position	background-repeat	border	border-bottom
border-bottom-color	border-bottom-style (solid border only)	border-bottom-width	border-color
border-left	border-left-color	border-left-style (solid border only)	border-left-width
border-right	border-right-color	border-right-style (solid border only)	border-right-width
border-spacing	border-style (solid border only)	border-top	border-top-color
border-top-style (solid border only)	border-top-width	border-width	bottom
clear	clip	color	content (strings, counters only)
counter-increment	counter-reset	cursor	display
float	font	font-family	font-size
font-style	font-weight	height	left

letter-spacing	line-height	list-style-type	margin
margin-bottom	margin-left	margin-right	margin-top
outline	outline-color	outline-style (solid, dotted, dashed only)	outline-width
padding	padding-bottom	padding-left	padding-right
padding-top	page-break-after	page-break-before	page-break-inside
position	right	text-align	text-decoration
text-indent	top	unicode-bidi	vertical-align
visibility	white-space (normal, nowrap only)	width	z-index

Using an image file URL

For optimal performance and reliability, Adobe recommends that you specify a local file URL for images stored on the server. In the following example, the `cfdocument` tag requests the server for images over HTTP even though the image files are stored locally:

```
<cfdocument format="PDF">
    <table>
        <tr>
            <td>bird</td>
            <td><image src="images/bird.jpg"></td>
        </tr>
        <tr>
            <td>fruit</td>
            <td><image src="images/fruit.jpg"></td>
        </tr>
        <tr>
            <td>rose</td>
            <td><image src="images/rose.jpg"></td>
        </tr>
    </table>
</cfdocument>
```

Also, in some applications, the browser displays a Red X image error instead of the image in the browser. For better performance, and to avoid Red X image errors, set the `localUrl` attribute to `yes`:

```
<cfdocument localUrl="yes" format="PDF">
    <table>
        <tr>
            <td>bird</td>
            <td><image src="images/bird.jpg"></td>
        </tr>
        <tr>
            <td>fruit</td>
            <td><image src="images/fruit.jpg"></td>
        </tr>
        <tr>
            <td>rose</td>
            <td><image src="images/rose.jpg"></td>
        </tr>
    </table>
</cfdocument>
```

Scope variables

When you use the `cfdocument` tag, ColdFusion creates a scope named `cfdocument`. This scope contains the following variables:

- currentpagenumber
- totalpagecount
- totalsectionnumber
- currentsectionpagenumber

ColdFusion 8 lets you use the scope variables inside any expression within a cfdocumentitem tag. For example, you can use the currentpagenumber variable to place the section name on even pages and the chapter name on odd pages in the header, as follows:

```
<cfdocument format="flashpaper">
    <cfdocumentitem type="header">
        <cfif (cfdocument.currentpagenumber mod 2) is 0>
            <cfoutput>#sectionTitle#</cfoutput>
        <cfelse>
            <cfoutput>#chapterTitle#</cfoutput>
        </cfif>
    </cfdocumentitem>
        ...
</cfdocument>
```

Bookmarks

ColdFusion 8 supports bookmarks. In the cfdocument tag, set the bookmark attribute to yes. Then do one of the following:

- Specify the bookmark name for each cfdocumentsection tag.
- Use the cfdocumentitem type = "bookmark".

The following example shows how to specify bookmarks for document sections:

```
<!--- This example creates two bookmarks named "Section 1" and "Section 2" in a PDF file. --->
<cfdocument format="pdf" bookmark="yes">
    <cfdocumentsection name="Section 1">
<!--- Insert HTML content here.--->
    </cfdocumentsection>
    <cfdocumentsection name="Section 2">
<!--- Insert HTML content here. --->
    </cfdocumentsection>
</cfdocument>
```

Example

Example 1

```
<!--- This example creates generates a FlashPaper document. --->
<cfdocument format="flashpaper">
<p>This is a document rendered by the cfdocument tag.</p>

<table width="50%" border="2" cellspacing="2" cellpadding="2">
<tr>
<td><strong>Name</strong></td>
<td><strong>Role</strong></td>
</tr>
<tr>
<td>Bill</td>
<td>Lead</td>
</tr>
<tr>
<td>Susan</td>
<td>Principal Writer</td>
</tr>
<tr>
```

```
<td>Adelaide</td>
<td>Part Time Senior Writer</td>
</tr>
<tr>
<td>Thomas</td>
<td>Full Time for 6 months</td>
</tr>
<tr>
<td>Michael</td>
<td>Full Time for 4 months</td>
</tr>
</table>
</cfdocument>
```

Example 2

```
<!--- The following example shows how to use the cfdocument scope variables to generate
section numbers and page numbers. --->

<cfdocument format="pdf">
<cfdocumentitem type="header">
    <table width="100%" border="0" cellpadding="0" cellspacing="0">
        <tr><td align="right"><cfoutput>#cfdocument.currentsectionpagenumber# of
            #cfdocument.totalsectionpagecount#</cfoutput></td></tr>
    </table>
</cfdocumentitem>

<cfdocumentitem type="footer">
    <table width="100%" border="0" cellpadding="0" cellspacing="0">
        <tr><td align="center"><cfoutput>#cfdocument.currentpagenumber# of
            #cfdocument.totalpagecount#</cfoutput></td></tr>
    </table>
</cfdocumentitem>

<cfdocumentsection>
    <h1>Section 1</h1>
    <cfloop from=1 to=50 index="i">
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est
laborum.<p>
    </cfloop>
</cfdocumentsection>

<cfdocumentsection>
    <h1>Section 2</h1>
    <cfloop from=1 to=50 index="i">
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est
laborum.<p>
    </cfloop>
</cfdocumentsection>

<cfdocumentsection>
<h1>Section 3</h1>
    <cfloop from=1 to=50 index="i">
```

```
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est
laborum.<p>
    </cfloop>
</cfdocumentsection>
</cfdocument>
```

cfdocumentitem

Description

Specifies action items for a PDF or FlashPaper document created by the `cfdocument` tag. Action items include the following:

- `header`
- `footer`
- `pagebreak`

Category

Data output tags

Syntax

```
<cfdocument ...>
    <cfdocumentitem
    type = "pagebreak|header|footer"
    header/footer text
    </cfdocumentitem>
</cfdocument>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfreport`, `cfdocument`, `cfdocumentsection`

History

ColdFusion 8: Added support for `cfdocument.currentpagenumber`, `cfdocument.totalpagecount` `cfdocument.totalsectionpagecount`, and `cfdocument.currentsectionpagenumber` scope variables.

ColdFusion MX 7.01: Added the `src`, `srcfile`, and `mimetype` attributes.

ColdFusion MX 7: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`type`	Required		Specifies the action: • `pagebreak`: starts a new page at the location of the tag. • `header`: uses the text between the `<cfdocumentitem>` and `</cfdocumentitem>` tags as the running header. • `footer`: uses the text between the `<cfdocumentitem>` and `</cfdocumentitem>` tags as the running footer.

Usage

Use the `cfdocumentitem` tag to control the formatting of a PDF or FlashPaper report. This tag must be wrapped inside a `<cfdocument>` `</cfdocument>` pair.

Write code for one `cfdocumentitem` tag for each page break, running header, or running footer.

ColdFusion 8 added support for `cfdocument` scope variables within the `cfdocumentitem` tag. You can use the `cfdocument` scope variable, `cfdocument.currentpagenumber`, to display the current page number in a header or footer. You can also use `cfdocument.totalpagecount` to display the total number of pages, for example:

```
...
<cfdocumentitem type= "footer>
   #cfdocument.currentpagenumber# of #cfdocument.totalpagecount#
</cfdocumentitem>
```

For an example that uses the `cfdocument.totalsectionpagecount` and `cfdocument.currentsectionpagenumber` scope variables, see `cfdocument`.

You can use `cfdocumentitem` tags with or without the `cfdocumentsection` tag, as follows:

Without cfdocumentsection he `cfdocumentitem` attribute applies to the entire document, as follows:

- If the tag is at the top of the document, it applies to the entire document.
- If the tag is in the middle of the document, it applies to the rest of the document.
- If the tag is at the end of the document, it has no affect.

With cfdocumentsection tags he `cfdocumentitem` attribute applies only to the section and overrides previously specified header and footer specifications.

Example

```
<cfquery datasource="cfdocexamples" name="parksQuery">
    SELECT parkname, suptmgr from parks
</cfquery>

<cfdocument format="PDF">
<cfdocumentitem type="header">National Parks Report</cfdocumentitem>
<!--- Use a footer with current page of totalpages format. --->
<cfdocumentitem type="footer">
<cfoutput>Page #cfdocument.currentpagenumber# of #cfdocument.totalpagecount#</cfoutput>
 </cfdocumentitem>

<h1>Park list</h1>
<table width="95%" border="2" cellspacing="2" cellpadding="2" >
<tr>
<th>Park</th>
<th>Manager</th>
</tr>
 <cfoutput query="parksQuery">
 <tr>
<td><font size="-1">#parkname#</font></td>
<td><font size="-1">#suptmgr#</font></td>
 </tr>
 </cfoutput>
</table>
</cfdocument>
```

cfdocumentsection

Description

Divides a PDF or FlashPaper document into sections. By using this tag in conjunction with a cfdocumentitem tag, each section can have unique headers, footers, and page numbers.

Category

Data output tags

Syntax

```
<cfdocument ...>
    <cfdocumentsection
        authPassword = "authentication password"
        authUser = "authentication user name"
        marginBottom = "number"
        marginLeft = "number"
        marginRight = "number"
        marginTop = "number"
        mimeType = "text/plain|application/xmlimage/jpeg|image/png|image/bmp|image/gif"
        name = "bookmark for the section"
        src = "URL|path relative to web root"
        srcfile = "absolute path of file"
        userAgent = "HTTP user agent identifier">
        HTML, CFML, and cfdocumentitem tags
    </cfdocumentsection>
</cfdocument>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfreport, cfdocument, cfdocumentitem

History

ColdFusion 8: Added the `name`, `authPassword`, `authUser`, and `userAgent` attributes.

ColdFusion MX 7.01: Added the `src`, `srcfile`, and `mimetype` attributes.

ColdFusion MX 7: Added this tag and the `margintop`, `marginbottom`, `marginleft`, `marginright` attributes.

Attributes

Attribute	Req/Opt	Default	Description
`authPassword`	Optional		Password sent to the target URL for Basic Authentication. Combined with `username` to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
`authUser`	Optional		User name sent to the target URL for Basic Authentication. Combined with `password` to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
`marginBottom`	Optional		Bottom margin in inches (default) or centimeters. To specify the bottom margin in centimeters, include the `unit="cm"` attribute in the parent `cfdocument` tag.
`marginLeft`	Optional		Left margin in inches (default) or centimeters. To specify the left margin in centimeters, include the `unit="cm"` attribute in the parent `cfdocument` tag.

Attribute	Req/Opt	Default	Description
marginRight	Optional		Right margin in inches (default) or centimeters. To specify the right margin in centimeters, include the unit="cm" attribute in the parent cfdocument tag.
marginTop	Optional		Top margin in inches (default) or centimeters. To specify the top margin in centimeters, include the unit="cm" attribute in the parent cfdocument tag.
mimeType	Optional	text/html	MIME type of the source document. Supported MIME types are: • text/html • text/plain • application/xml • image/jpeg • image/png • image/gif If you do not specify this attribute explicitly, ColdFusion uses the filename to determine the MIME type.
name	Optional		Bookmark name for the section.
src	Optional		URL or the relative path to the web root. You cannot specify both the src and srcfile attributes.
srcfile	Optional		Absolute path of a file that is on the server. You cannot specify both the src and srcfile attributes.
userAgent	Optional	ColdFusion	Text to put in the HTTP User-Agent request header field. Used to identify the request client software.

Usage

Use the cfdocumentsection tag to divide a report into sections. Within each cfdocumentsection tag, you can use one or more cfdocumentitem tags to specify unique headers and footers for each section.

When using cfdocumentsection, ColdFusion ignores HTML and CFML not enclosed within cfdocumentsection tags.

The margin attributes override margins specified in previous sections or in the parent cfdocument tag. If you specify margin attributes, the units are controlled by the unit attribute of the parent cfdocument tag; the unit attribute has a default value of inches. The cfdocumentsection tag forces a page break so that each section starts on a new page.

ColdFusion 8 added the name attribute to support bookmarks. Bookmarks defined at the documentsection tag level are children of the cfdocument root.

Example

Example 1

```
<cfquery datasource="cfdocexamples" name="empSalary">
SELECT Emp_ID, firstname, lastname, e.dept_id, salary, d.dept_name
FROM employee e, departmt d
WHERE e.dept_id = d.dept_id
ORDER BY d.dept_name
</cfquery>

<cfdocument format="PDF">
<cfoutput query="empSalary" group="dept_id">
    <cfdocumentsection>
    <cfdocumentitem type="header">
        <font size="-3"><i>Salary Report</i></font>
    </cfdocumentitem>
```

```
    <cfdocumentitem type="footer">
        <font size="-3">Page #cfdocument.currentpagenumber#</font>
    </cfdocumentitem>
    <h2>#dept_name#</h2>
     <table width="95%" border="2" cellspacing="2" cellpadding="2" >
    <tr>
         <th>Employee</th>
         <th>Salary</th>
    </tr>
    <cfset deptTotal = 0 >
    <!--- inner cfoutput --->
    <cfoutput>
    <tr>
<td><font size="-1">
        #empSalary.lastname#, #empSalary.firstname#</font>
        </td>
        <td align="right"><font size="-1">
        #DollarFormat(empSalary.salary)#</font>
        </td>
 </tr>
         <cfset deptTotal = deptTotal + empSalary.salary>
    </cfoutput>
        <tr>
<td align="right"><font size="-1">Total</font></td>
        <td align="right"><font size="-1">#DollarFormat(deptTotal)#</font></td>
</tr>
        <cfset deptTotal = 0>
     </table>
    </cfdocumentsection>
    </cfoutput>
</cfdocument>
```

Example 2: Bookmarks

```
<!--- This example uses the name attribute to define bookmarks in a PDF document at the
    section level. --->
<cfdocument format="pdf" bookmark="yes">
    <cfdocumentsection name="section 1">
        <!--- Insert some HTML content here. --->
    </cfdocumentsection>
    <cfdocumentsection name="section 2">
        <!--- Insert some HTML content here. --->
    </cfdocumentsection>
</cfdocument>
```

cfdump

Description

Use the cfdump tag to get the elements, variables, and values of most kinds of ColdFusion objects. Useful for debugging. You can display the contents of simple and complex variables, objects, components, user-defined functions, and other elements.

Category

Debugging tags, Variable manipulation tags

Syntax

```
<cfdump
    var = "#variable#"
    expand = "yes|no"
    format = "text|html"
    hide = "columns|keys"
    keys = "number of keys to display for structures"
    label = "text"
    metainfo = yes|no"
    output = "browser|console|file"
    show = "columns|keys"
    showUDFs = "yes|no"
    top = "number of rows|number of levels">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfcookie, cfparam, cfsavecontent, cfschedule, cfset, cftimer, cfwddx

History

- ColdFusion 8: Added the show, format, hide, keys, metainfo, output, and showUDFs attributes.
- ColdFusion MX 7: Added the top attribute.
- ColdFusion MX 6.1: Added the ability to dump COM objects; it displays the methods and Get and Put properties typeinfo information for the object.

Attributes

Attribute	Req/Opt	Default	Description
var	Required		Variable to display. Enclose a variable name in number signs. These kinds of variables yield meaningful cfdump output: • array • CFC • COM object • file object • Java object • simple • query • structure • UDF • wddx • xml
expand	Optional	yes	• yes: in Internet Explorer and Mozilla, expands views. • no: contracts expanded views.
format	Optional	text	Use with the output attribute to specify whether to save the results of a cfdump to a file in text or HTML format.
hide	Optional	all	For a query, this is a column name or a comma-delimited list of column names. For a structure, this is a key or a comma-delimited list of keys. If you specify a structure element that doesn't exist, ColdFusion ignores it and does not generate an error.
keys	Optional	9999	For a structure, the number of keys to display.
label	Optional		A string; header for the dump output. Ignored if the value of the var attribute is a simple types.
metainfo	Optional	yes	For use with queries only. Includes information about the query in the cfdump results, including whether the query was cached, the execution time, and the SQL. You must specify metainfo="no" to exclude this information from the query result.
output	Optional	browser	Where to send the results of cfdump. The following values are valid: • browser • console • filename The filename should include the full pathname of the file. You can specify an absolute path, or a path that is relative to the ColdFusion temporary directory. You can use the GetTempDirectory() function to determine the ColdFusion temporary directory.
show	Optional	all	For a query, this is a column name or a comma-delimited list of column names. For a structure, this is a key or a comma-delimited list of keys.
showUDFs	Optional	yes	• yes: includes UDFs, with the methods collapsed. • no: excludes UDFs.
top	Optional	9999	The number of rows to display. For a structure, this is the number of nested levels to display.

Usage

The expand/contract display capability is useful when working with large structures, such as XML document objects, structures, and arrays.

To display a construct, use code such as the following, in which *myDoc* is a variable of type XmlDocument:

```
<cfif IsXmlDoc(mydoc) is "yes">
    <cfdump var="#mydoc#">
</cfif>
```

The tag output is color-coded according to data type.

If a table cell is empty, this tag displays "[empty string]".

Example

```
<!--- This example shows how to use this tag to display the CGI scope as a structure: --->

<cfdump var="#cgi#">


<!--- This displays information about file objects. --->
<cfscript>
myfile = FileOpen("c:\temp\test1.txt", "read");
</cfscript>
myfile refers to:
<cfdump var="#myfile.filepath#">
```

cfelse

Description

Used as the last control block in a `cfif` tag block to handle any case not identified by the `cfif` tag or a `cfelseif` tag.

Category

Flow-control tags

Syntax

```
<cfif expression>
        HTML and CFML tags
    <cfelseif expression>
        HTML and CFML tags
    <cfelse>
        HTML and CFML tags
</cfif>
```

See also

`cfif`, `cfelseif`, `cfabort`, `cfbreak`, `cfexecute`, `cfexit`, `cflocation`, `cfloop`, `cfswitch`, `cfthrow`, `cftry`

Usage

If the values of the *expressions* in the containing `cfif` tag and all `cfelseif` tags are no, ColdFusion processes the code between this tag and the `cfif` end tag. This tag must be inside a `cfif` tag block. It does not require an end tag.

For more information and an example, see cfif.

cfelseif

Description

Used as a control block in a `cfif` tag block to handle any case not identified by the `cfif` tag or a `cfelseif` tag.

Category

Flow-control tags

Syntax

```
<cfif expression>
        HTML and CFML tags
    <cfelseif expression>
        HTML and CFML tags
    <cfelse>
        HTML and CFML tags
</cfif>
```

See also

`cfif`, `cfelse`, `cfabort`, `cfbreak`, `cfexecute`, `cfexit`, `cflocation`, `cfloop`, `cfswitch`, `cfthrow`, `cftry`

Usage

If the value of the *expression* in this tag is `yes`, and the values of the *expressions* in the containing `cfif` tag and preceding `cfelseif` tags are `no`, ColdFusion processes the code between this tag and a following `cfelseif` or `cfelse` tag, or the `cfif` end tag and then skips to the code following the `cfif` end tag. Otherwise, ColdFusion skips the code.

This tag must be inside a `cfif` tag block. It does not require an end tag.

For more information and an example, see "cfif" on page 298.

cferror

Description

Displays a custom HTML page when an error occurs. This lets you maintain a consistent look and feel among an application's functional and error pages.

Category

Exception handling tags, Extensibility tags, Application framework tags

Syntax

```
<cferror
    template = "template path"
    type = "exception|validation|request"
    exception = "exception type"
    mailTo = "e-mail address">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfrethrow`, `cfthrow`, `cftry`, "Handling Errors" on page 247 in the *ColdFusion Developer's Guide*.

History

ColdFusion MX: Deprecated the `monitor` option of the `exception` attribute. It might not work, and might cause an error, in later releases.

Attributes

Attribute	Req/Opt	Default	Description
`template`	Required		Relative path to the custom error page. (A ColdFusion page was formerly called a template.)
`type`	Required		Type of error that the custom error page handles. The type also determines how ColdFusion handles the error page. For more information, see "Specifying a custom error page" on page 255 in the *ColdFusion Developer's Guide*. • `exception`: an exception of the type specified by the `exception` attribute. • `validation`: errors recognized by server-side type validation. • `request`: any encountered error.

Attribute	Req/Opt	Default	Description
exception	Optional	any	Type of exception that the tag handles: • application: application exceptions. • database: database exceptions. • template: ColdFusion page exceptions. • security: security exceptions. • object: object exceptions. • missingInclude: missing include file exceptions. • expression: expression exceptions. • lock: lock exceptions. • *custom_type*: developer-defined exceptions, defined in the cfthrow tag. • any: all exception types. For more information on exception types, see cftry.
mailTo	Optional		An e-mail address. This attribute is available on the error page as the variable error.mailto. ColdFusion does not automatically send anything to this address.

Usage

Use this tag to provide custom error messages for pages in an application. This lets you maintain a consistent look and feel within the application, even when errors occur.

You generally embed this tag in your Application CFC or Application.cfm file to specify error-handling responsibilities for an entire application. You **must** put it in one of these files if you specify type="validation"; ColdFusion ignores it on any other page.

The cftry and cfcatch tags provide a more interactive way to handle ColdFusion errors within a ColdFusion page than the cferror tag, but the cferror tag is a good safeguard against general errors.

To ensure that error pages display successfully, avoid using the cfencode utility to encode pages that include the cferror tag.

Page types

The following table describes the types of errors you can specify and code you can use on the pages that handle these error type:

Page type	Description	Use
Exception	Dynamically invoked by the CFML language processor when it detects an unhandled exception condition. Uses the full range of CFML tags. Error variables must be in `cfoutput` tags.	Can handle specific exception types or display general information for exceptions.
Request	Includes the error variables described in the Error variables section. Cannot include CFML tags, but you can display values of the error variables by enclosing them in number signs (#), as in `#error.MailTo#`.	Use as a backup error handler to other error handling methods, including exception type.
Validation	Handles data input validation errors that occur when submitting a form that uses hidden form-field validation or onSubmit validation. Cannot include CFML tags, but you can display values of the error variables by enclosing them in number signs (#), as in `#Error.InvalidFields#`. You must specify the validation error handler in the Application.cfc or Application.cfm file.	Handles hidden form-field or onSubmit format validation errors only.

Error variables

The exception-handling page specified in the `cferror` tag `template` attribute contains one or more error variables. ColdFusion substitutes the value of the error variable when an error displays.

The following table lists error variables:

Page type	Error variable	Description
Validation only	error.validationHeader	Validation message header text.
	error.invalidFields	Unordered list of validation errors.
	error.validationFooter	Validation message footer text.
Request and Exception	error.diagnostics	Detailed error diagnostics from ColdFusion MX.
	error.mailTo	E-mail address (same as value in `cferror.MailTo`).
	error.dateTime	Date and time when error occurred.
	error.browser	Browser that was running when error occurred.
	error.remoteAddress	IP address of remote client.
	error.HTTPReferer	Page from which client accessed link to page where error occurred.
	error.template	Page executing when error occurred.
	error.generatedContent	The content generated by the page up to the point where the error occurred.
	error.queryString	URL query string of client's request.

Page type	Error variable	Description
Exception only	error.message	Error message associated with the exception.
	error.rootCause	The root cause of the exception. This structure contains the information that is returned by a cfcatch tag. For example, for a database exception, the SQL statement that caused the error is in the error.RootCause.Sql variable. For Java exceptions, this variable contains the Java servlet exception reported by the JVM as the cause of the "root cause" of the exception.
	error.tagContext	Array of structures containing information for each tag in the tag stack. The tag stack consists of each tag that is currently open.
	error.type	Exception type.

Note: *If* `type = "exception"`*, you can substitute the prefix* `cferror` *for* `Error`*; for example,* `cferror.diagnostics`*,* `cferror.mailTo`*, or* `cferror.dateTime`**.**

Example

```
<h3>cferror Example</h3>

<!--- Example of cferror call within a page.
         NOTE: If you use cferror type="VALIDATION" you MUST put it in
         Application.cfc or Application.cfm --->
<cferror type = "REQUEST"
template = "request_err.cfm"
mailTo = "admin@mywebsite.com">
<!--- This query calls a non-existent datasource, triggering an error to be handled. --->
<cfquery name="testQuery" datasource="doesNotExist">
select * from nothing
</cfquery>

<!--- Example of the page (request_err.cfm) to handle this error. --->
<html>
<head>
<title>We're sorry -- An Error Occurred</title>
</head>
<body>
<h2>We're sorry -- An Error Occurred</h2>
<p>
If you continue to have this problem, please contact #error.mailTo#
with the following information:</p>
<p>
<ul>
<li><b>Your Location:</b> #error.remoteAddress#
<li><b>Your Browser:</b> #error.browser#
<li><b>Date and Time the Error Occurred:</b> #error.dateTime#
<li><b>Page You Came From:</b> #error.HTTPReferer#
<li><b>Message Content</b>:
<p>#error.diagnostics#</p>
</ul>
```

cfexchangecalendar

Description

Creates, deletes, modifies, gets, and responds to Microsoft Exchange calendar events, and gets calendar event attachments.

History

ColdFusion 8: Added this tag.

Category

Communications tags

Syntax

```
create
<cfexchangecalendar
    required
    action = "create"
    event = "#event information structure#"
    optional
    connection = "connection ID"
    result = "variable for event UID">

delete
<cfexchangecalendar
    required
    action = "delete"
    uid = "event UID,event UID, ..."
    optional
    connection = "connection ID"
    message = "string"
    notify = "yes|no">

deleteAttachments
<cfexchangecalendar
    required
    action = "deleteAttachments"
    uid = "event UID"
    optional
    connection = "connection ID">

get
<cfexchangecalendar
    required
    action = "get"
    name = "query identifier"
    optional
    connection = "connection ID">

getAttachments
<cfexchangecalendar
    required
    action = "getAttachments"
    name = "query identifier"
    uid = "event UID"
    optional
    attachmentPath = "directory path"
    connection = "connection ID">
    generateUniqueFilenames = "no|yes"
```

```
modify
<cfexchangecalendar
    required
    action = "modify"
    event = "#event information structure#"
    uid = "event UID"
    optional
    connection = "connection ID">

respond
<cfexchangecalendar
    required
    action = "respond"
    responseType = "accept|decline|tentative"
    uid = "event UID"
    optional
    connection = "connection ID"
    message = "string">
    notify = "yes|no">
```

Note: *For all actions, see* `cfexchangeconnection` *for additional attributes that you use if you do not specify the* `connection` *attribute. If you omit the* `connection` *attribute, you must create a temporary connection by specifying* `cfexchangeconnection` *tag attributes in the* `cfexchangecalendar` *tag. In this case, ColdFusion closes the connection when the tag completes. For details, see the* `cfexchangeconnection` *tag* `open` *action.*

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfexchangeconnection`, `cfexchangecontact`, `cfexchangefilter`, `cfexchangemail`, `cfexchangetask`, "Working with meetings and appointments" on page 1030 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Action	Req/Opt	Default	Description
`action`	N/A	Required		The action to take. Must be one of the following values: • `create` • `delete` • `deleteAttachments` • `get` • `getAttachments` • `modify` • `respond`
`attachmentPath`	`getAttachments`	Optional		The filepath of the directory in which to put the attachments. If the directory does not exist, ColdFusion creates it. **Note:** If you omit this attribute, ColdFusion does not save any attachments. If you specify a relative path, the path root is the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.

Attribute	Action	Req/Opt	Default	Description
connection	all	Optional		The name of the connection to the Exchange server, as specified in the cfexchangeconnection tag. If you omit this attribute, you must create a temporary connection by specifying cfexchangeconnection tag connection attributes in the cfexchangecalendar tag.
event	create modify	Required		A reference to the structure that contains the event properties to be set or changed, and their values. You must specify this attribute in number signs (#). For more information on the event structure, see Usage.
generateUniqueFilenames	getAttachments	Optional	no	A Boolean value that specifies whether to generate unique file-names if multiple attachments have the same filenames. If two or more attachments have the same filename and this option is yes, ColdFusion appends a number to the filename body (before the extension) of any conflicting filenames. Thus, if three attachments have the name myfile.txt, ColdFusion saves the attachments as myfile.txt, myfile1.txt, and myfile2.txt.
message	delete respond	Optional		The text of an optional message to send in the response or deletion notification.
name	get getAttachments	Required		The name of the ColdFusion query variable that contains the retrieved events or information about the attachments that were retrieved. For more information on the returned data, see Usage.
notify	delete respond	Optional	true	Boolean value that specifies whether to notify others of the changes made to the event.
responseType	respond	Required		Must be one of the following values: • accept • decline • tentative
result	create	Optional		The name of a variable that contains the UID of the event that is created. You use the UID value in the uid attribute of actions other than create to identify the event to be acted on.
uid	delete getAttachments modify respond	Required		Case-sensitive Exchange UID value or values that uniquely identify the event or events on which to perform the action. For the delete action, this attribute can be a comma-delimited list of UID values. The deleteAttachments, getAttachments, modify, and respond actions allow only a single UID value.

Usage

The cfexchangecalendar tag manages calendar events on the Exchange server. Use the cfexchangecalendar to do the following actions:

- Create an appointment or meeting event. You can create all-day events.
- Delete one or more events.
- Get one or more events that conform to an optional set of filter specifications, such as the subject, sender or recipient ID, time received, and so on.
- Get the attachments for a specific event.
- Modify an existing event.

- Respond to an event.

To use this tag, you must have a connection to an Exchange server. If you are using multiple tags that interact with the Exchange server, such as if you are creating several contact records, use the `cfexchangeconnection` tag to create a persistent connection. Then specify the connection identifier in each `cfexchangecalendar` tag, or in any other ColdFusion Exchange tag, if you are also accessing tasks, contacts, or mail. Doing this eliminates the overhead of creating and closing the connection for each tag.

Alternatively, you can create a temporary connection that lasts only for the time that ColdFusion processes the single `cfexchangecalendar` tag. To do this, specify the connection attributes directly in the `cfexchangecontact` tag. For details on the connection attributes, see the `cfexchangeconnection` tag.

Note: *To create an Exchange calendar appointment, create a calendar event and do not specify any required or optional attendees.*

The create action

When you specify the `create` action, the `event` attribute must specify a structure that contains the information that defines the events. The structure can have the following entries:

Element	Default	Description
`AllDayEvent`	`no`	A Boolean value that indicates whether this is an all-day event.
`Attachments`		One or more paths to the files to send as attachments. Separate filepaths with semicolons (;) for Windows, and colons (:) for UNIX and Linux. Paths to the attachments must be absolute. If you specify one or more attachments for a `modify` action, the specified attachments are added to any existing attachments; the pre-existing attachments are not deleted.
`Duration`		The duration of the event in minutes.
`EndTime`		The end time of the event, in any valid ColdFusion date-time format.
`Importance`	`normal`	One of the following values: • `high` • `normal` • `low`.
`IsRecurring`		A Boolean value that indicates whether this event repeats. If `yes`, you must specify a `RecurrenceType` element and elements to specify the recurrence details. For information on the recurrence fields, see the next table.
`Location`		A string that specifies the location of the event.
`Message`		A string that contains a message about the event. The string can include HTML formatting.
`OptionalAttendees`		A comma-delimited list of mail IDs.
`Organizer`		A string that specifies the name of the meeting organizer.
`Reminder`		The time, in minutes before the event, at which to display a reminder message.
`RequiredAttendees`		A comma-delimited list of mail IDs.
`Resources`		A comma-delimited list of mail IDs for Exchange scheduling resources, such as conference rooms and display equipment.
`Sensitivity`		The valid values are `normal`, `company-confidential`, `personal`, and `private`.

Element	Default	Description
StartTime		The start time of the event, in any valid ColdFusion date-time format. If you specify a date and time in this attribute and specify a YEARLY RecurrenceType with no other recurrence attributes, the event recurs yearly at the day and time specified in this attribute.
Subject		A string that describes the event subject.

The following table lists the elements that you use to specify the event recurrence if you set the IsRecurring field to a yes value. For a detailed description of how to specify event recurrence, see "Specifying Calendar recurrence" on page 1032 in the *ColdFusion Developer's Guide*.

Element	Type	Default	Description
RecurrenceType	all	DAILY	Used only if the structure has a yes IsRecurring element. Must be one of the following values: • DAILY • WEEKLY • MONTHLY • YEARLY
RecurrenceNoEndDate	all	yes	Boolean value; if yes, the event recurs until you change or delete the event. Cannot be used with RecurrenceCount or RecurrenceEndDate.
RecurrenceCount	all		The number of times the event recurs. Cannot be used with RecurrenceEndDate or RecurrenceNoEndDate.
RecurrenceEndDate	all		The date of the last recurrence. Cannot be used with RecurrenceCount or RecurrenceEndDate.
RecurrenceFrequency	DAILY, WEEKLY, MONTHLY	1	The frequency of the recurrence in days, weeks, or months, depending on the type. For example, for DAILY recurrence, a RecurrenceFrequency of 3 schedules the event every three days.
RecurEveryWeekDay	DAILY		The recurrence of the event on every week day, but not on Saturday or Sunday. Cannot be used with RecurrenceFrequency.
RecurrenceDays	WEEKLY		The day or days of the week on which the event occurs. Must be one or more of the following values in a comma-delimited list: MON, TUE, WED, THU, FRI, SAT, SUN If you omit this field for a weekly recurrence, the event recurs on the day of the week that corresponds to the specified start date.
RecurrenceDay	MONTHLY, YEARLY		The day of the week on which the event occurs. Must be one of the following values: • MON • TUE • WED • THU • FRI • SAT • SUN

Element	Type	Default	Description
RecurrenceWeek	MONTHLY, YEARLY		The week of the month or year on which the event recurs. The valid values are: • first • second • third • fourth • last
RecurrenceMonth	YEARLY		The month of the year on which the event recurs. The valid values are JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, and DEC.

The delete action

When you specify the delete action, you must specify a uid attribute with a comma-delimited list of one or more Exchange UIDs that identify the events to delete. Use the get action, with an appropriate filter expression, to determine the UID values to specify.

If all UIDs that you specify are invalid, the cfexchangecalendar tag generates an error. If at least one UID is valid, the tag ignores any invalid UIDs and deletes the items specified by the valid UID.

The get action

When you specify the get action, use child cfexchangefilter tags to specify the messages to get. For detailed information on filters, see cfexchangefilter.

When the tag completes processing, the query object specified by the name attribute contains one record for each retrieved message. Each record has the following columns:

AllDayEvent	Duration	EndTime	From
HasAttachment	HtmlMessage	Importance	IsRecurring
Location	Message	OptionalAttendees	Organizer
Reminder	RequiredAttendees	Resources	Sensitivity
StartTime	Subject	UID	

The following table describes the From, HtmlMessage, Message, and UID fields. For detailed information on the other fields, see the table in the create action description.

Column	Description
From	The Exchange ID of the person who created the event.
HtmlMessage	An HTML-formatted version of the message about the event.
Message	A plain-text version of the message about the event.
UID	The Exchange unique identifier for the mail event. Use this value to identify the event in the delete, getAttachments, and modify actions.

The getAttachments action

When you use the getAttachments action, you must specify a single UID and a name attribute. The cfexchangecalendar tag populates a query object with the specified name. Each record has the following information about an attachment to the event specified by the UID:

Column	Description
attachmentFileName	The filename of the attachment.
attachmentFilePath	The absolute path of the attachment file on the server. If you omit the `attachmentPath` attribute, this column contains the empty string.
CID	The content-ID of the attachment. Typically used in HTML img tags to embed images in a message.
mimeType	The MIME type of the attachment, such as text/html.
isMessage	A Boolean value that specifies whether the attachment is a message.
size	The attachment size in bytes.

The tag places the attachments in the directory specified by the `attachmentPath` attribute. If you omit the `attachmentPath` attribute, ColdFusion does not get any attachments, it gets the information about the attachments. This lets you determine the event's attachments without incurring the overhead of getting the attachment files.

The modify action

When you specify the `modify` action, you select the event to modify by specifying a `uid` attribute with single event UID; multiple UIDs are not allowed. You populate the `event` structure with only the fields that you are changing. For a detailed description of the fields and their valid values, see the table in the `create` action.

If an event has attachments and you specify attachments when you modify the event, the new attachments are added to the previous attachments; they do not replace them. You must use the `deleteAttachments` action to remove any attachments.

The respond action

You use the `respond` action to respond to a meeting notification that you received by using the `cfexchangemail` tag. A meeting does not appear in your calendar, and cannot be accessed by using the `cfexchangecalendar` tag, until you respond to the mail message and accept or tentatively accept the request.

When you specify the `respond` action, you must specify the UID, from the notification mail message, of the event to which you are responding. You must also specify the response type; that is, whether you are accepting, rejecting, or tentatively accepting the event. You can optionally specify a message to include in the response and set a flag whether to notify the creator of the event of your response.

For detailed information on using the `respond` action, see "Working with meeting notices and requests" on page 1030 in the *ColdFusion Developer's Guide*.

Example

The following example lets you create, and then modify a calendar event. When you first submit the form, ColdFusion creates the calendar event and redisplays the form with the data you entered. You should accept the event before you modify the form and resubmit it. When you submit the form a second time, ColdFusion sends the modification information. For more information, see "Working with meetings and appointments" on page 1030 in the *ColdFusion Developer's Guide*.

This example resends all the event data (to limit the example length), but you could change the example so that it only sends modified data.

```
<!--- Create a structure to hold the event information. --->
<!--- A self-submitting form for the event information --->
<!--- This example omits recurrence to keep the code relatively simple --->
<cfparam name="form.eventID" default="0">
```

```
<!--- If the form was submitted, populate the event structure from it. --->
<cfif isDefined("Form.Submit")>
        <cfscript>
        sEvent.AllDayEvent="no";
        sEvent=StructNew();
        sEvent.Subject=Form.subject;
        if (IsDefined("Form.allDay")) {
            sEvent.AllDayEvent="yes";
            sEvent.StartTime=createDateTime(Year(Form.date), Month(Form.date),
                Day(Form.date), 8, 0, 0);
        }
        else {
            sEvent.StartTime=createDateTime(Year(Form.date), Month(Form.date),
                Day(Form.date), Hour(Form.startTime), Minute(Form.startTime), 0);
            sEvent.EndTime=createDateTime(Year(Form.date), Month(Form.date),
                Day(Form.date), Hour(Form.endTime), Minute(Form.endTime), 0);
        }
        sEvent.Location=Form.location;
        sEvent.RequiredAttendees=Form.requiredAttendees;
        sEvent.OptionalAttendees=Form.optionalAttendees;
        //sEvent.Resources=Form.resources;
        if (Form.reminder NEQ "") {
            sEvent.Reminder=Form.reminder;
        }
        else {
            sEvent.Reminder=0;
        }
        sEvent.Importance=Form.importance;
        sEvent.Sensitivity=Form.sensitivity;
        sEvent.message=Form.Message;
    </cfscript>

    <!--- If this is the first time the form is being submitted
            Create a new event. --->
    <cfif form.eventID EQ 0>
    <!--- Create the event in Exchange --->
        <cfexchangecalendar action="create"
            username ="#user1#"
            password="#password1#"
            server="#exchangeServerIP#"
            event="#sEvent#"
            result="theUID">
        <!--- Output the UID of the new event. --->
        <cfif isDefined("theUID")>
            <cfoutput>Event Added. UID is#theUID#</cfoutput>
            <cfset Form.eventID = theUID >
        </cfif>
    <cfelse>
    <!--- The form is being resubmitted with new data, so update the event. --->
        <cfexchangecalendar action="modify"
            username ="#user1#"
            password="#password1#"
            server="#exchangeServerIP#"
            event="#sEvent#"
            uid="#Form.eventID#">
        <cfoutput>Event ID #Form.eventID# Updated.</cfoutput>

    </cfif>
</cfif>

<cfform format="xml" preservedata="yes" style="width:500" height="600">
```

```
    <cfinput type="text" label="Subject" name="subject" style="width:435"><br />
    <cfinput type="checkbox" label="All Day Event" name="allDay">
    <cfinput type="datefield" label="Date" name="date" validate="date" style="width:100">
    <cfinput type="text" label="Start Time" name="startTime" validate="time"
        style="width:100">
    <cfinput type="text" label="End Time" name="endTime" validate="time"
        style="width:100"><br />
    <cfinput type="text" label="Location" name="location" style="width:435"><br />
    <cfinput type="text" label="Required Attendees" name="requiredAttendees"
        style="width:435"><br />
    <cfinput type="text" label="Optional Attendees" name="optionalAttendees"
        style="width:435"><br />
    <cfinput type="text" label="Resources" name="resources" style="width:435"><br />
    <cfinput type="text" label="Reminder (minutes)" validate="integer" name="reminder"
        style="width:200">
    <cfselect name="importance" label="Importance" style="width:100">
        <option value="normal">Normal</option>
        <option value="high">High</option>
        <option value="low">Low</option>
    </cfselect>
    <cfselect name="sensitivity" label="Sensitivity" style="width:100">
        <option value="normal">Normal</option>
        <option value="company-confidential">Confidential</option>
        <option value="personal">Personal</option>
        <option value="private">Private</option>
    </cfselect>
    <cfinput type="textarea" label="Message" name="message" style="width:435;
        height:100">
    <cfinput type="hidden" name="eventID" value="#Form.EventID#">
    <cfinput type="Submit" name="submit" value="Submit">
</cfform>
```

cfexchangeconnection

Description

Opens or closes a persistent connection to an Microsoft Exchange server, or gets information about mailbox subfolders. You must have a persistent or temporary connection to use the `cfexchangecalendar`, `cfexchangecontact`, `cfexchangemail`, and `cfexchangetask` tags.

History

ColdFusion 8: Added this tag.

Category

Communications tags

Syntax

```
open
<cfexchangeconnection
    required
    action = "open"
    connection = "connection ID">
    server = "Exchange server ID"
    username = "Exchange user ID">
    optional
    ExchangeServerLanguage = "Language name"
    formBasedAuthentication = "no|yes">
    mailboxName = "Exchange mailbox">
    password = "user password"
    port = "IP port"
    protocol = "http|https"
    proxyHost = "proxy host URL"
    proxyPort = "proxy IP port"

getSubfolders
<cfexchangeconnection
    required
    action = "getSubfolders"
    connection = "connection ID">
    name = "query name"
    optional
    folder = "Exchange folder path">
    recurse = "no|yes">
OR
<cfexchangeconnection
    required
    action = "getSubfolders"
    name = "query name"
    server = "Exchange server ID"
    username = "Exchange user ID">
    optional
    ExchangeServerLanguage = "Language name"
    folder = "Exchange folder path">
    formBasedAuthentication = "no|yes">
    mailboxName = "Exchange mailbox">
    password = "user password"
    port = "IP port"
    protocol = "http|https"
    proxyHost = "proxy host URL"
    proxyPort = "proxy IP port"
```

```
    recurse = "no|yes">

close
<cfexchangeconnection
    required
    action = "close"
    connection = "connection ID">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfexchangecalendar`, `cfexchangecontact`, `cfexchangefilter`, `cfexchangemail`, `cfexchangetask`
"Managing connections to the Exchange server" on page 1014 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Action	Req/Opt	Default	Description
`action`	all	Required		The action to take. Must be one of the following values: • `open`: Open a new persistent named connection • `close`: Close a named connection • `getSubfolders`: Get information about the subfolders of a specific folder.
`connection`	all	Required for `open` and `close` actions		The name of the connection. You can specify this ID in any tag that you use with the open connection.
`ExchangeServerLanguage`	`open` `getSubfolders`	Optional	`english`	The language of the Exchange server. If you are not sure, you can specify the empty string. For all values except english, including the empty string, the tag tries to get folder names from the server in the client's local language. In some cases, such as when there is a large amount of data on the server, it might take significant time to get folder names from Exchange server in the local language.
`folder`	`getSubfolders`	Optional	The root of the mailbox	The forward slash (/) delimited path from the root of the mailbox to the folder for which to get subfolders. If a folder name contains a forward slash, use the _xF8FF_ escape sequence to specify the character in the name.
`formBasedAuthentication`	`open` `getSubfolders`	Optional	`no`	A Boolean value that specifies whether to display a login form and use form based authentication when making the connection. If the attribute value is `no` (the default), and the Exchange server returns a 440 error status when ColdFusion tries to connect, ColdFusion displays the login form and attempts to use form based authentication. Therefore, you can safely omit this attribute if you do not know if the server requires form based authentication.

Attribute	Action	Req/Opt	Default	Description
mailboxName	open getSubfolders	Optional		The ID of the Exchange mailbox to use. Specify this attribute to access a mailbox whose owner has delegated access rights to the account specified in the username attribute.
name	getSubfolders	Required		The name of the ColdFusion query variable that contains information about the subfolders.
password	open getSubfolders	Optional		The user's password for accessing the Exchange server.
port	open getSubfolders	Optional	80	The port on the server connect to, most commonly port 80.
protocol	open getSubfolders	Optional	http	The protocol to use for the connection. Valid values are http and https.
proxyHost	open getSubfolders	Optional		The URL or IP address of a proxy host, if required for access to the network.
proxyPort	open getSubfolders	Optional		The port on the proxy server to connect to, most commonly port 80.
recurse	getSubfolders	Optional	false	A Boolean value: • true: get information on the immediate subfolders of the specified folder only. • false: get information on all levels of subfolders of the specified folder.
server	open getSubfolders	Required		The IP address or URL of the server that is providing access to Exchange.
username	open getSubfolders	Required		The Exchange user ID.

Note: *If you specify the getSubfolders action, you can specify the attributes that are listed as working for both the open and getSubfolders actions only if you do not specify a connection attribute.*

Usage

The cfexchangeconnection tag can open or close a persistent connection with an Exchange server. If you use the cfexchangeconnection to open a connection before you use any cfexchangecalendar, cfexchangecontact, cfexchangemail, or cfexchangetask tags, you can use multiple tags to interact with the Exchange server without incurring the overhead of creating a connection for each tag.

Note: *To establish any connection, the Exchange server must grant you Outlook Web Access. For information on how to enable this access, see "Enabling Outlook web access" on page 1015 in the ColdFusion Developer's Guide.*

Use the cfexchangeconnection tag to close a persistent connection when you are finished accessing the Exchange server. If you do not close the connection, it remains open and does not time out.

The cfexchangecalendar, cfexchangecontact, cfexchangemail, and cfexchangetask tags also let you specify the open action connection attributes (but not the connection attribute) to create a temporary connection that lasts for the duration of the single tag's activities, without requiring you to use the cfexchangeconnection tag to create the connection. In this case, ColdFusion automatically closes the connection when the tag completes processing.

The getSubfolders action can get information about the immediate subfolders of a specified folder (or of the top level of the mailbox), or information about all levels of subfolders. You must have a persistent connection to get the subfolders.

The query returned by the `getSubfolders` action has the following columns:

Column	Contents
FOLDERNAME	The name of the subfolder, for example, ColdFusion.
FOLDERPATH	The forward slash (/) delimited path to the folder from the mailbox root, including the folder name, for example, Inbox/Marketing/ColdFusion.
FOLDERSIZE	Size of the folder in bytes.

Note: *The ColdFusion exchange tags, including* `cfexchangeconnection` *use WebDAV to connect to the exchange server. HTTP access must be enabled on the exchange server to use the tags.*

Example

The following example opens a connection, gets all mail sent from spamsource.com, and deletes the messages from the Exchange server:

```
<cfexchangeConnection
    action="open"
    username="#user1#"
    password="#password1#"
    server="#exchangeServerIP#"
    connection="testconn1">

<cfexchangemail action="get" name="spamMail" connection="testconn1">
    <cfexchangefilter name="fromID" value="spamsource.com">
</cfexchangemail>

<cfloop query="spamMail">
    <cfexchangeMail action="delete" connection="testconn1" uid="#spamMail.uid#">
</cfloop>

<cfexchangeConnection
    action="close"
    connection="testconn1">
```

cfexchangecontact

Description

Creates, deletes, modifies, and gets Microsoft Exchange contact records, and gets contact record attachments.

History

ColdFusion 8: Added this tag.

Category

Communications tags

Syntax

```
create
<cfexchangecontact
    required
    action = "create"
    contact = "#contact information structure#"
    optional
    connection = "connection ID"
    result = "variable for contact UID">

delete
<cfexchangecontact
    required
    action = "delete"
    uid = "contact UID,contact UID, ..."
    optional
    connection = "connection ID">

deleteAttachments
<cfexchangecontact
    required
    action = "deleteAttachments"
    uid = "contact UID"
    optional
    connection = "connection ID">

get
<cfexchangecontact
    required
    action = "get"
    name = "query identifier"
    optional
    connection = "connection ID">

getAttachments
<cfexchangecontact
    required
    action = "getAttachments"
    name = "query identifier"
    uid = "contact UID"
    optional
    attachmentPath = "directory path"
    connection = "connection ID"
    generateUniqueFilenames = "no|yes">

modify
<cfexchangecontact
```

```
required
action = "modify"
contact = "#contact information structure#"
uid = "contact UID"
optional
connection = "connection ID>"
```

Note: *If you omit the* `connection` *attribute, you must create a temporary connection by specifying* `cfexchangeconnection` *tag attributes in the* `cfexchangecontact` *tag. In this case, ColdFusion closes the connection when the tag completes. For details, see the* `cfexchangeconnection` *tag* `open` *action.*

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfexchangecalendar`, `cfexchangeconnection`, `cfexchangefilter`, `cfexchangemail`, `cfexchangetask`, "Interacting with Microsoft Exchange Servers" on page 1013 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Action	Req/Opt	Default	Description
`action`	N/A	Required		The action to take. Must be one of the following values: • `create` • `delete` • `deleteAttachments` • `get` • `getAttachments` • `modify`
`attachmentPath`	`getAttachments`	Optional		The absolute filepath of the directory in which to put the attachments. If the directory does not exist, ColdFusion creates it. **Note:** If you omit this attribute, ColdFusion does not save any attachments.
`connection`	all	Optional		The name of the connection to the Exchange server, as specified in the `cfexchangeconnection` tag. If you omit this attribute, you must create a temporary connection by specifying `cfexchangeconnection` tag connection `open` action attributes in the `cfexchangecontact` tag.
`contact`	`create` `modify`	Required		A reference to the structure that contains the `contact` properties to be set or changed and their values. You must specify this attribute in number signs (#). For more information on the event structure, see Usage.

Attribute	Action	Req/Opt	Default	Description
generateUniqueFilenames	getAttachments	Optional	no	A Boolean value that specifies whether to generate unique filenames if multiple attachments have the same filenames. If two or more attachments have the same filename and this option is yes, ColdFusion appends a number to the filename body (before the extension) of any conflicting filenames. Thus, if three attachments have the name myfile.txt, ColdFusion saves the attachments as myfile.txt, myfile1.txt, and myfile2.txt.
name	get getAttachments	Required		The name of the ColdFusion query variable that contains the returned contact records or information about the attachments that were retrieved. For more information on the returned data, see Usage.
result	create	Optional		The name of a variable that contains the UID of the contact that is created. Use this value in the uid attribute of other actions to identify the contact to be acted on.
uid	getAttachments delete modify	Required		A case-sensitive Exchange UID value that uniquely identifies the contacts on which to perform the action. For the delete action, this attribute can be a comma-delimited list of UID values. The deleteAttachments, getAttachments, and modify actions allow only a single UID value.

When you specify the create or modify action, the contact attribute must specify a structure that contains information that defines the events. The structure can have the following elements. You have to include only the elements that you are setting or changing.

Assistant	Attachments	BusinessAddress	BusinessFax
BusinessPhoneNumber	Categories	Company	Department
Description	DisplayAs	Email1	Email2
Email3	FirstName	HomeAddress	HomePhoneNumber
JobTitle	LastName	MailingAddressType	Manager
MiddleName	MobilePhoneNumber	NickName	Office
OtherAddress	OtherPhoneNumber	Pager	Profession
SpouseName	WebPage		

All fields except the BusinessAddress, HomeAddress, and OtherAddress fields contain text; the three address fields must contain structures with the following text fields:

- Street
- City
- State
- Zip
- Country

The Attachments field must contain the pathnames of any attachments to include in the contact. To specify multiple files, separate filepaths with semicolons (;) for Windows, and colons (:) for UNIX and Linux. You must use absolute paths.

If you specify one or more attachments for a `modify` action, they are added to any existing attachments; the pre-existing attachments are not deleted.

The `Categories` field can have a comma-delimited list of the contact's categories.

If you do not specify a `DisplayAs` field, Exchange sets the display name to `FirstName`, `LastName`.

Usage

The `cfexchangecontact` tag manages contact records on the Exchange server. Use the `cfexchangecontact` tag to perform the following actions:

- Create a contact.
- Delete one or more contacts.
- Get one or more contact records that conform to an optional set of filter specifications, such as the last name, job title, or home phone number, and so on.
- Get the attachments for a specific contact record.
- Modify an existing contact.

To use this tag, you must have a connection to an Exchange server. If you are using multiple tags that interact with the Exchange server, such as if you are creating several contact records, you should use the `cfexchangeconnection` tag to create a persistent connection. You then specify the connection identifier in each `cfexchangecontact`, or any other ColdFusion Exchange tag, if you are also accessing tasks, contacts, or mail. Doing this eliminates the overhead of creating and closing the connection for each tag.

Alternatively, you can create a temporary connection that lasts only for the time that ColdFusion processes the single `cfexchangecontact` tag. To do this, you specify the connection attributes directly in the `cfexchangecontact` tag. For details on the connection attributes, see the `cfexchangeconnection` tag `open` action.

The delete action

When you specify the `delete` action you must specify a `uid` attribute with a comma-delimited list of one or more Exchange UIDs that identify the contacts to delete. You can use the `get` action, with an appropriate filter expression, to determine the UID values to specify.

If all UIDs that you specify are invalid, the `cfexchangecontact` tag generates an error. If at least one UID is valid, the tag ignores any invalid UIDs and deletes the items specified by the valid UID.

The get action

When you specify the `get` action, the query object specified by the `name` attribute contains one record for each retrieved contact. The query object has columns with the same names and data formats as the fields listed for the `contact` attribute structure, with the following changes:

- The query object has a Boolean `HasAttachment` column, and does not have an `Attachments` column. If the `HasAttachment` field is `yes`, use the `getAttachments` action to retrieve the attachments.
- The query object has an additional `UID` column with the unique identifier for the contact record in the Exchange server. Use this value in the `uid` attribute of the `getAttachments`, `delete`, and `modify` actions to identify the required record.
- The query object has an additional `HtmlDescription` column. The `Description` column has a plain-text version of the description, and the `HtmlDescription` column text includes the description's HTML formatting.

You use child `cfexchangefilter` tags to specify the messages to get. For detailed information, see `cfexchange-filter`.

The getAttachments action

When you use the `getAttachments` action, you must specify a single UID and a `name` attribute. The `cfexchangecontact` tag populates a query object with the specified name. Each record has the following information about an attachment to the contact specified by the UID:

Column name	Description
attachmentFileName	The filename of the attachment.
attachmentFilePath	The absolute path of the attachment file on the server. If you omit the `attachmentPath` attribute, this column contains the empty string.
CID	The content-ID of the attachment. Typically used in HTML img tags to embed images in a message.
mimeType	The MIME type of the attachment, such as text/html.
isMessage	A Boolean value specifying whether the attachment is a message.
size	The attachment size in bytes.

If you omit the `attachmentPath` attribute, ColdFusion does not get any attachments; it gets the information about the attachments. This lets you determine the event's attachments without incurring the overhead of getting the attachment files.

The modify action

If you specify the `modify` action, the `uid` attribute must specify a single Exchange UID. The `contact` structure must specify only the fields that you are changing. Any fields that you do not specify remain unchanged.

If a contact has attachments and you specify attachments when you modify the contact, the new attachments are added to the previous attachments, and do not replace them. You must use the `deleteAttachments` action to remove any attachments.

Example

The following example lets a user enter information in a form and creates a contact on the Exchange server with the information:

```
<!--- Create a structure to hold the contact information. --->
<cfset sContact="#StructNew()#">

<!--- A self-submitting form for the contact information --->
<cfform format="flash" width="550" height="460">
    <cfformitem type="html"><b>Name</b></cfformitem>
    <cfformgroup type="horizontal" label="">
        <cfinput type="text" label="First" name="firstName" width="200">
        <cfinput type="text" label="Last" name="lastName" width="200">
    </cfformgroup>
    <cfformgroup type="VBox">
        <cfformitem type="html"><b>Address</b></cfformitem>
        <cfinput type="text" label="Company" name="Company" width="435">
        <cfinput type="text" label="Street" name="street" width="435">
        <cfinput type="text" label="City" name="city" width="200">
        <cfselect name="state" label="State" width="100">
            <option value="CA">CA</option>
            <option value="MA">MA</option>
            <option value="WA">WA</option>
        </cfselect>
        <cfinput type="text" label="Country" name="Country" width="200" Value="U.S.A.">
        <cfformitem type="html"><b>Phone</b></cfformitem>
        <cfinput type="text" validate="telephone" label="Business" name="businessPhone"
```

```
            width="200">
        <cfinput type="text" validate="telephone" label="Mobile" name="cellPhone"
            width="200">
        <cfinput type="text" validate="telephone" label="Fax" name="fax" width="200">
        <cfformitem type="html"><b>Email</b></cfformitem>
        <cfinput type="text" validate="email" name="email" width="200">
    </cfformgroup>

    <cfinput type="Submit" name="submit" value="Submit" >
</cfform>


<!--- If a form was submitted, populate the contact structure from it. --->
<cfif isDefined("Form.Submit")>
    <cfscript>
        sContact.FirstName=Form.firstName;
        sContact.Company=Form.company;
        sContact.LastName=Form.lastName;
        sContact.BusinessAddress.Street=Form.street;
        sContact.BusinessAddress.City=Form.city;
        sContact.BusinessAddress.State=Form.state;
        sContact.BusinessAddress.Country=Form.country;
        sContact.BusinessPhoneNumber=Form.businessPhone;
        sContact.MobilePhoneNumber=Form.cellPhone;
        sContact.BusinessFax=Form.fax;
        sContact.Email1=Form.email;
    </cfscript>

    <!--- Create the contact in Exchange --->
    <cfexchangecontact action="create"
        username ="#user1#"
        password="#password1#"
        server="#exchangeServerIP#"
        contact="#sContact#"
        result="theUID">

    <!--- Display a confirmation that the contact was added. --->
    <cfif isDefined("theUID")>
        <cfoutput>Contact Added. UID is#theUID#</cfoutput>
    </cfif>
</cfif>
```

cfexchangefilter

Description

Specifies filter parameters that control the actions of `cfexchangemail`, `cfexchangecalendar`, `cfexchangetask`, and `cfexchangecontact`, get operations.

History

ColdFusion 8: Added this tag.

Category

Communications tags

Syntax

```
<cfexchangefilter
    name = "filter type"
    value = "filter value">

OR

<cfexchangefilter
    name = "filter type"
    from = "date/time"
    to = "date/time">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfexchangecalendar`, `cfexchangeconnection`, `cfexchangecontact`, `cfexchangemail`, `cfexchangetask`, "Getting Exchange items and attachments" on page 1019 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		The type of filter to use.
`from`	Optional		The start date or date/time combination of the range to use for filtering. Cannot be used with the `value` attribute. If you specify a `from` attribute without a `to` attribute, the filter selects for all entries on or after the specified date or time. The value can be in any date/time format recognized by ColdFusion, but must correspond to a value that is appropriate for the filter type.
`to`	Optional		The end date or date/time combination for the range used for filtering. Cannot be used with the `value` attribute. If you specify a `to` attribute without a `from` attribute, the filter selects for all entries on or before the specified date or time. The value can be in any date/time format recognized by ColdFusion, but must correspond to a value that is appropriate for the filter type.
`value`	Optional		The filter value for all filters that do not take a date or time range. Cannot be used with the `from` and `to` attributes . ColdFusion generates an error if you specify this attribute with an empty contents. Therefore, you cannot use the empty string to search for empty values.

The cfexchangeCalendar tag filters can have the following name attributes and associated value, or to and from attributes that you use to specify the filter parameters for the specified action:

name attribute	Specification attributes	Valid specification attribute values
maxRows	value	A positive integer specifying the maximum number of matching rows to return. By default, the maximum number of rows is 100.
allDayEvent	value	A Boolean value.
duration	value	An integer number of minutes.
endTime	from	A string that ColdFusion can interpret as a date-time value.
	to	
fromID	value	An Exchange user ID.
hasAttachment	value	A Boolean value.
importance	value	One of the following values: • high • normal • low
isRecurring	value	A Boolean value.
location	value	A string.
message	value	A string.
optionalAttendees	value	A comma-delimited list of Exchange user IDs.
organizer	value	A string that identifies the organizer. This value does not need to be an Exchange ID or e-mail address.
requiredAttendees	value	A comma-delimited list of Exchange user IDs.
sensitivity	value	One of the following values: • normal • personal • private • confidential
startTime	from	A string that ColdFusion can interpret as a date-time value.
	to	
subject	value	A string.
UID	value	A case-sensitive Exchange message UID that uniquely identifies one message.

The cfexchangecontact tag filters can have the following name attributes and associated value attributes. Unlike other tags, you do not use from or to attributes.

name attribute	value attribute
maxRows	A positive integer that specifies the maximum number of matching rows to return. By default, the maximum number of rows is 100.
assistant	A string.
businessAddress	A structure with the following fields: Street, City, State, Zip, Country.
businessFax	A string.
businessPhoneNumber	A string.

name attribute	value attribute
categories	A comma-delimited list of categories. The filter searches for contacts that match all the categories in the list.
company	A string.
description	A string.
displayAs	A string.
email1	A string.
email2	A string.
email3	A string.
firstName	A string.
hasAttachment	A Boolean value.
homeAddress	A structure with the following fields: Street, City, State, Zip, Country.
homePhoneNumber	A string.
jobTitle	A string.
lastName	A string.
mailingAddressType	One of the following values: Home, Business, Other.
manager	A string.
middleName	A string.
mobilePhoneNumber	A string.
nickName	A string.
office	A string.
otherAddress	A structure with the following fields: Street, City, State, Zip, Country.
otherPhoneNumber	A string.
pager	A string.
profession	A string.
spouseName	A string.
webPage	A string.

The cfexchangemail tag filters can have the following name attributes and associated value, or to and from attributes that you use to specify the filter parameters for the specified action:

name attribute	Specification attributes	Specification attribute values
maxRows	value	A positive integer that specifies the maximum number of matching rows to return. By default, the maximum number of rows is 100.
bcc	value	A comma-delimited list of Exchange or web e-mail addresses.
cc	value	A comma-delimited list of Exchange or web e-mail addresses.

name attribute	Specification attributes	Specification attribute values
folder	value	The forward slash (/) delimited path from the root of the Exchange mailbox to the folder to search. By default, the filter searches the top level of the Inbox. The cfexchangemail tag searches only the specified folder, and does not search any subfolders. If a folder name contains a forward slash, use the _xF8FF_ escape sequence to specify the character in the name. For the get and move actions, you can use the cfexchangemail tag folder attribute instead of this field; however, this field takes precedence over the value specified in the folder attribute.
fromID	value	An Exchange or web e-mail address.
hasAttachment	value	A Boolean value
importance	value	One of the following values: • high • normal • low
isRead	value	A Boolean value.
message	value	A string.
MessageType	value	One of the following values: Mail, Meeting, Meeting_Cancel, Meeting_Request, Meeting_Response, or All. If you omit this attribute, the filter gets messages of all types. The Meeting attribute gets messages with Meeting_Cancel, Meeting_Request, and Meeting_Response types.
MeetingUID	value	A case-sensitive Exchange calendar event UID. Meeting UIDs are used in Meeting_request or Meeting_response message types only. Do not specify this field if you specify a MessageType field value of Mail.
sensitivity	value	One of the following values: • normal • personal • private • confidential
subject	value	A string.
timeReceived	from	A string that ColdFusion can interpret as a date-time value.
	to	
timeSent	from	A string that ColdFusion can interpret as a date-time value.
	to	
toID	value	A comma-delimited list of Exchange or web e-mail addresses.
uid	value	A case-sensitive Exchange message UID.

The cfexchangetask tag filters can have the following name attributes and associated value, or to and from attributes that you use to specify the filter parameters for the specified action:

name attribute	Specification attributes	Specification attribute values
maxRows	value	A positive integer specifying the maximum number of matching rows to return. By default, the maximum number of rows is 100.
actualWork	value	A number representing the number of hours. Use decimal numbers to specify minutes.
billingInfo	value	A string.
companies	value	A string.
dateCompleted	value	A string that ColdFusion can interpret as a date-time value.
dueDate	from	A string that ColdFusion can interpret as a date-time value.
	to	
mail_ID	value	A comma-delimited list of Exchange mail IDs. This filter value is useful if the connection user has delegate rights for multiple users and you want to select the tasks of a limited number of those users.
message	value	A string.
mileage	value	A string.
percentCompleted	value	A number between 0 and 100.
priority	value	One of the following values: • high • normal • low
reminderDate	value	A string that ColdFusion can interpret as a date-time value.
startDate	from	A string that ColdFusion can interpret as a date-time value.
	to	
status	value	Must be one of the following values: • NOT_STARTED • IN_PROGRESS • COMPLETED • WAITING • DEFERRED
subject	value	A string.
totalWork	value	A number that represents the number of hours. Use decimal numbers to specify minutes.
UID	value	A case-sensitive Exchange UID.

Usage

The cfexchangefilter tag specifies the conditions to match when ColdFusion gets mail messages, calendar entries, tasks, or contacts. Only those entries that match the specified filter conditions are returned in the structure specified by the parent tag's name attribute. If the filter specifies a field that takes a text string, such as Message and or Subject, ColdFusion returns items that contain the exact phrase that you specify in the value attribute.

The cfexchangefilter tag must be a child tag of a cfexchangecalendar, cfexchangecontact, cfexchangemail, or cfexchangetask tag with an action attribute value of get.

If you specify multiple `cfexchangefilter` tags in the body of a ColdFusion exchange tag, such as `cfexchangemail`, the specified filters are cumulative, and the selected records match the conditions specified in all the `cfexchangefilter` tags. If you specify multiple `cfexchangefilter` tags with the same `name` attribute value, the last tag with that attribute specifies the filter conditions.

Example

The following example gets the mail messages that were sent to a user during the last week from any e-mail address that includes adobe.com. To focus on getting messages, rather than on displaying data, the example uses the `cfdump` tag to show the results.

```
<cfset endTime = Now()>
<cfset startTime = DateAdd("d","-7", endTime)>
<cfexchangemail action="get" name="weeksMail" server="#exchangeServerIP#"
        username ="#user1#" password="#password1#">
    <cfexchangefilter name="FromID" value="adobe.com">
    <cfexchangefilter name="TimeSent" from="#startTime#" to="#endTime#">
</cfexchangemail>

<cfdump var="#weeksMail#">
```

cfexchangemail

Description

Gets mail messages and attachments, deletes messages, and sets properties for messages on a Microsoft Exchange server.

History

ColdFusion 8: Added this tag.

Category

Communications tags

Syntax

```
delete
<cfexchangemail
    required
    action = "delete"
    uid = "message UID,message UID,..."
    optional
    connection = "connection ID"
    folder = "Exchange folder path">

deleteAttachments
<cfexchangemail
    required
    action = "deleteAttachments"
    uid = "message UID"
    optional
    connection = "connection ID">
    folder = "Exchange folder path">

get
<cfexchangemail
    required
    action = "get"
    name = "query identifier"
    optional
    connection = "connection ID"
    folder = "Exchange folder path">

    <cfexchangefilter name = "filter type" value = "filter value">
    <cfexchangefilter name = "filter type" value = "filter value">
    ...
</cfexchangemail>

getAttachments
<cfexchangemail
    required
    action = "getAttachments"
    name = "query identifier"
    uid = "message UID"
    optional
    attachmentPath = "directory path"
    connection = "connection ID"
    folder = "Exchange folder path"
    generateUniqueFilenames = "no|yes">

getMeetingInfo
```

```
<cfexchangemail
    required
    action = "getMeetingInfo"
    meetingUID = "meeting UID"
    name = "query identifier"
    optional
    connection = "connection ID"
    mailUID = "message UID">

move
<cfexchangemail
    required
    action = "move"
    destinationFolder = "Exchange folder path"
    optional
    connection = "connection ID"
    folder = "Exchange folder path">

    <cfexchangefilter name = "filter type" value = "filter value">
    <cfexchangefilter name = "filter type" value = "filter value">
    ...
</cfexchangemail>

set
<cfexchangemail
    required
    action = "set"
    message = "#structure with values to set#">
    uid = "message UID">
    optional
    connection = "connection ID"
    folder = "Exchange folder path">
```

Note: *If you omit the* `connection` *attribute, you must create a temporary connection by specifying* `cfexchangeconnection` *tag attributes in the* `cfexchangemail` *tag. In this case, ColdFusion closes the connection when the tag completes. For details, see the* `cfexchangeconnection` *tag* `open` *action.*

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfexchangecalendar`, `cfexchangeconnection`, `cfexchangecontact`, `cfexchangefilter`, `cfexchangetask`, "Interacting with Microsoft Exchange Servers" on page 1013 in the *ColdFusion Developer's Guide*

Attributes

Note: *If an attribute, such as* `folder` *or* `destinationFolder` *takes a folder path, and the folder name contains forward slashes (/), specify the folder name by using the _xF8FF_ escape character to prevent exchange from interpreting the character as a path delimiter.*

Attribute	Action	Req/Opt	Default	Description
`action`	all	Required		The action to take. Must be one of the following values: • `delete` • `deleteAttachments` • `get` • `getAttachments` • `getMeetingInfo` • `move` • `set`
`attachmentPath`	`getAttachments`	Optional		The filepath of the directory in which to put the attachments. If the directory does not exist, ColdFusion creates it. **Note:** If you omit this attribute, ColdFusion does not save any attachments. If you specify a relative path, the path root is the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`connection`	all	Optional		The name of the connection to the Exchange server, as specified in the `cfexchangeconnection` tag. If you omit this attribute, you must create a temporary connection by specifying `cfexchangeconnection` tag `open` action attributes in the `cfexchangecalendar` tag.
`destinationFolder`	`move`	Required		The forward slash (/) delimited path, relative to the root of the mailbox, of the folder to which to move the message or messages.
`folder`	all *except* `getMeetingInfo`	Optional		The forward slash (/) delimited path, relative to the root of the mailbox, of the folder that contains the message or messages. The `cfexchangemail` tag looks in the specified folder only, and does not search subfolders. For the `get` and `move` actions specifying a `cffexchangefilter` child tag with a `name="folder"` attribute is equivalent to setting this attribute, and takes precedence over this attribute's value. If you omit this attribute, or for `get` and `move` actions, if you do not use the corresponding `cfexchangefilter` setting, Exchange looks in the top level of the Inbox.
`generateUniqueFilenames`	`getAttachments`	Optional	no	A Boolean value that specifies whether to generate unique filenames if multiple attachments have the same filenames. If two or more attachments have the same filename and this option is `yes`, ColdFusion appends a number to the filename body (before the extension) of any conflicting filenames. Thus, if three attachments have the name myfile.txt, ColdFusion saves the attachments as myfile.txt, myfile1.txt, and myfile2.txt.
`mailUID`	`getMeetingInfo`	Optional		The case-sensitive UID of the mail message that contains the meeting request, response, or cancellation notification. Use this attribute if there are multiple messages about a single meeting.
`meetingUID`	`getMeetingInfo`	Required		The case-sensitive UID of the meeting for which you received the notification.

Attribute	Action	Req/Opt	Default	Description
message	set	Required		A reference to a structure that contains the properties to be set and their values. You must specify this attribute in number signs (#). For more information on the message structure, see Usage.
name	get getAttachments getMeetingInfo	Required		The name of the ColdFusion query variable that contains the returned mail messages or the retrieved information about the attachments or meeting. For more information on the returned data, see Usage.
uid	delete getAttachments set	Required		The case-sensitive UIDs of the messages on which to perform the action. For the delete action, this attribute can be a comma-delimited list of UID values. The deleteAttachments, getAttachments, and set actions allow only a single UID value.

Usage

The cfexchangemail tag performs mail actions on an Exchange server that you cannot do by using the cfmail tag. (You must use the cfmail tag to send, forward, and reply to mail messages.) Use the cfexchangemail tag to perform the following actions:

- Permanently delete one or more mail messages from the server.
- Get the attachments for a specific message.
- Get one or more messages that conform to an optional set of filter specifications, such as the subject, sender or recipient ID, time received, and so on.
- Get the attachments for a specific message.
- Get detailed information about a meeting for which you have a notification, such as a meeting request or cancellation notice.
- Move one or more messages from one folder to another, including to the Deleted Items folder.
- Set the properties of a specific mail message.

To use this tag, you must have a connection to an Exchange server. If you are using multiple tags that interact with the exchange server, such as if you are creating several contact records, you should use the cfexchangeconnection tag to create a persistent connection. You then specify the connection identifier in each cfexchangemail tag, or any other ColdFusion Exchange tag, if you are also accessing tasks, contacts, or connections. Doing this saves the overhead of creating and closing the connection for each tag.

Alternatively, you can create a temporary connection that lasts only for the time that ColdFusion processes the single cfexchangemail tag. To do this, you specify the connection attributes directly in the cfexchangemail tag. For details on the connection attributes, see the cfexchangeconnection tag.

The delete action

The delete action permanently deletes a message from the server, and is equivalent to the Outlook Shift-Delete keystroke action. Use the move action to move a message to the Deleted Items folder, which is equivalent to the Outlook Delete keystroke action.

When you specify the delete action you must specify a uid attribute with a comma-delimited list of one or more Exchange UIDs that identify the tasks that you want to delete. You can use the get action, with an appropriate filter expression, to determine the UID values to specify.

If all UIDs that you specify are invalid, the `cfexchangemail` tag generates an error. If at least one UID is valid, the tag ignores any invalid UIDs and deletes the items specified by the valid UID.

The get action

When you specify the `get` action, you use child `cfexchangefilter` tags to specify the messages to get. For detailed information, see `cfexchangefilter`. When the tag completes processing, the query object specified by the `name` attribute contains one record for each matching message that was found. Each record has the following columns:

Column	Description
BCC	A comma-delimited list of Exchange user IDs or web e-mail .
CC	A comma-delimited list of Exchange user IDs or web e-mail addresses.
Folder	The forward slash (/) delimited path from the root of the Exchange mailbox to the mail folder containing the message.
FromID	An Exchange user IDs or web e-mail addresses.
HasAttachment	A Boolean value that indicates whether the message has at least one attachment.
HTMLMessage	A string containing a HTML-formatted version of the message.
IsRead	A Boolean value.
Message	A string with a plain-text version of the message contents.
MessageType	One of the following strings: • `Mail` • `Meeting_Cancel` • `Meeting_Request` • `Meeting_Response`
MeetingResponse	If the message type is `Meeting_response`, this column contains the response code as one of the following strings: `Accept`, `Decline`, or `Tentative`. This field is not used for other message types.
MeetingUID	If the message type is `Meeting_Cancel`, `Meeting_request`, or `Meeting_response` this column contains the UID of the calendar event for which this message was sent. Use this value in the `cfexchangecalendar` tag to respond to a request. This field is not used for the `Mail` message type.
Sensitivity	One of the following strings: • `public` • `private` • `normal` • `company-confidential`
Subject	A string.
TimeReceived	A Coldfusion date-time object.
TimeSent	A Coldfusion date-time object.
ToId	A comma-delimited list of Exchange user IDs or web mail IDs.
UID	The Exchange UID of the message.

Note: *An invitation sender can get a meeting request message only if the sender is on the attendee list.*

The getAttachments action

When you use the `getAttachments` action, you must specify a single UID and a `name` attribute. The `cfexchangecontact` tag populates a query object specified by the `name` attribute with one record for each attachment. Each record has the following information about the mail attachment specified by the UID:

Column name	Description
attachmentFileName	The filename of the attachment.
attachmentFilePath	The absolute path of the attachment file on the server. If you omit the `attachmentPath` attribute, this column contains the empty string.
CID	The content-ID of the attachment. Used in HTML img tags to embed images in a message.
mimeType	The MIME type of the attachment, such as text/html.
isMessage	A Boolean value that specifies whether the attachment is a message.
size	The attachment size in bytes.

If you omit the `attachmentPath` attribute, ColdFusion does not get any attachments; it gets the information about the attachments. This lets you determine the event's attachments without incurring the overhead of getting the attachment files.

If a message has multiple attachments with the same name, the attachment information structure always lists the attachments with their original, duplicate, names, even if you specify `generateUniqueFilenames="yes"`. The `generateUniqueFilenames` attribute only affects the names of the files on disk.

The getMeetingInfo action

You use the `getMeetingInfo` action to get meeting-specific information, such as the meeting start and end times, location, and so on, about a meeting for which you have received a notification message, such as an invitation request or cancellation notice. This information is not available directly in the notification message query object that is returned by the `get` action.

Note: *At the time of publication, the following information does not completely reflect the behavior of the `getMeetingInfo` action. For updated information, see `cfexchangemail` in ColdFusion 8 LiveDocs on the Adobe website.*

When you specify the `getMeetingInfo` action, you specify a `meetingUID` attribute with the UID of the meeting. You get this UID value from the query record that is returned by the `get` action. You can optionally specify a `messageUID` attribute with the UID of the specific message that contains the notification; if you receive multiple messages about a single meeting, you can use this attribute to select a single notification message.

When the tag completes processing, the query object specified by the `name` attribute contains one record for each matching message that was found. Each record has the following columns:

Field	Description
AllDayEvent	A Boolean value that indicates whether this is an all day event.
Duration	The duration of the event in minutes.
EndTime	The end time of the event, in ColdFusion ODBC date-time format.
From	The mail ID of the person who sent the meeting notification.
HasAttachment	A Boolean value that indicates whether this event has attachments.

Field	Description
Importance	One of the following values: • high • normal • low
IsRecurring	A Boolean value that indicates whether this event repeats.
Location	A string that specifies the location of the event.
MeetingUID	The UID of the event in the calendar.
Message	A string that contains a message about the event.
OptionalAttendees	A comma-delimited list of mail IDs.
Organizer	A string. This value is not necessarily an Exchange ID or e-mail address.
Reminder	The time, in minutes before the event, at which to display a reminder message.
RequiredAttendees	A comma-delimited list of mail IDs.
Resources	A comma-delimited list of mail IDs for Exchange scheduling resources, such as conference rooms and display equipment.
Sensitivity	One of the following values: • normal • company-confidential • personal • private
StartTime	The start time of the event, in ODBC date-time format.
Subject	A string describing the event subject.
TimeReceived	The time the message was received, in ODBC date-time format.
UID	The UID of the message that contains the event notification.

The move action

Use the `move` action to move one or more messages from one folder to another folder. You can use this action to move messages to the Deleted Items folder, which is equivalent to the Outlook Delete keystroke action.

When you specify the `move` action you specify the destination folder, and optionally the folder containing the messages to move. (The default source folder is the Inbox). You use child `cfexchangefilter` tags to specify the messages to get. For detailed information, see `cfexchangefilter`.

The set action

When you specify the `set` action, the structure specified by the `message` attribute contains key-value pairs that specify the message properties to set. The following table lists the key names and their valid values:

Key name	Valid values
IsRead	`yes`, `no`
Importance	`high`, `normal`, `low`
Sensitivity	`normal`, `company-confidential`, `personal`, `private`

Example

The following example gets the attachments to all mail messages in the Inbox from docuser2 in the last week. It puts each message's attachments in a directory with a unique name. It cannot use the UID as a filename because, for each message with attachments, the UID can contain the application reports of the UID, directory path, subject, date, and sender of the message, followed by a table that lists the message's attachments. The table includes the attachment name, size, and MIME type.

```
<!--- Index for message attachement directory --->
<cfset i=1>
<!--- Dates for date range --->
<cfset rightNow = Now()>
<cfset lastWeek = DateAdd("d","-7", rightNow)>

<cfexchangeconnection
    action="open"
    username ="#user1#"
    password="#password1#"
    server="#exchangeServerIP#"
    connection="testconn1">

<cfexchangemail action="get" folder="Inbox " name="weeksMail" connection="testconn1">
    <cfexchangefilter name="FromID" value="docuser2">
    <cfexchangefilter name="TimeSent" from="#lastWeek#" to="#rightNow#">
</cfexchangemail>

<cfloop query="weeksMail">
    <cfif weeksmail.HasAttachment>
        <cfexchangemail action="getAttachments"
            connection="testconn1"
            folder="Inbox/MailTest"
            uid="#weeksmail.uid#"
            name="attachData"
            attachmentPath="C:\temp\cf_files\attachments\msg_#i#"
            generateUniqueFilenames="yes">
        <cfoutput>
            Message ID #weeksmail.uid# attachments are in the directory
                C:\temp\cf_files\attachments\Msg_#i#<br />
            <br />
                Message information:<br />
                Subject: #weeksmail.Subject#<br />
                Sent: #dateFormat(weeksmail.TimeSent)#<br />
                From:  #weeksmail.FromID#<br />
            <br />
            Attachments<br />
            <cftable query="attachData" colheaders="yes">
                <cfcol header="File Name" text="#attachmentFilename#">
                <cfcol header="Size" text="#size#">
                <cfcol header="MIME type" text="#mimeType#">
            </cftable>
        </cfoutput>
        <cfset i++>
    </cfif>
</cfloop>
<cfexchangeconnection action="close" connection="testconn1">
```

cfexchangetask

Description

Creates, deletes, modifies, and gets Microsoft Exchange tasks, and gets task attachments.

Note: *For all actions, see* `cfexchangeconnection` *for additional attributes that you use if you do not specify the connection attribute.*

History

ColdFusion 8: Added this tag.

Category

Communications tags

Syntax

```
create
<cfexchangetask
    required
    action = "create"
    task = "#task information structure#"
    optional
    connection = "connection ID"
    result = "variable for event UID">

delete
<cfexchangetask
    required
    action = "delete"
    uid = "task UID,task UID, ..."
    optional
    connection = "connection ID">

deleteAttachments
<cfexchangetask
    required
    action = "deleteAttachments"
    uid = "task UID"
    optional
    connection = "connection ID">

get
<cfexchangetask
    required
    action = "get"
    name = "query identifier"
    optional
    connection = "connection ID">

getAttachments
<cfexchangetask
    required
    action = "getAttachments"
    name = "query identifier"
    uid = "task UID"
    optional
    attachmentPath = "directory path"
    connection = "connection ID">
```

```
modify
<cfexchangetask
    required
    action = "modify""
    task = "#task information structure#"
    uid = "task UID">
    optional
    connection = "connection ID">
```

Note: *If you omit the* `connection` *attribute, you must create a temporary connection by specifying* `cfexchangeconnection` *tag attributes in the* `cfexchangetask` *tag. In this case, ColdFusion closes the connection when the tag completes. For details, see the* `cfexchangeconnection` *tag* `open` *action.*

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfexchangecalendar`, `cfexchangeconnection`, `cfexchangecontact`, `cfexchangefilter`, `cfexchangemail`, "Interacting with Microsoft Exchange Servers" on page 1013 in the *ColdFusion Developer's Guide*.

Attributes

The following table provides detailed information about each attribute. It lists the attribute name, the actions (`action` attribute values) to which it apples, whether it is required or optional for those actions, and its default value, if any, and provides a detailed description of the attribute and its valid values.

Attribute	Action	Req/Opt	Default	Description
`action`	all	Required		The action to take. Must be one of the following values: • `create` • `delete` • `deleteAttachments` • `get` • `getAttachments` • `modify`
`attachmentPath`	`getAttachments`	Optional		The filepath of the directory in which to put the attachments. If the directory does not exist, ColdFusion creates it. **Note:** If you omit this attribute, ColdFusion does not save any attachments. If you specify a relative path, the path root is the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`connection`	all	Optional		The name of the connection to the Exchange server, as specified in the `cfexchangeconnection` tag. If you omit this attribute, you must create a temporary connection by specifying `cfexchangeconnection` tag connection attributes in the `cfexchangetask` tag.
`name`	`get` `getAttachments`	Required		The name of the ColdFusion query variable that contains the returned task records or information about the attachments that were retrieved. For more information on the returned data, see Usage.
`result`	`create`	Optional		The name of a variable that contains the UID of the task that is created. You use this value in the `uid` attribute of other actions to identify the task to be acted on.

Attribute	Action	Req/Opt	Default	Description
task	create modify	Required		A reference to the structure that contains the task properties to be set or changed and their values. You must specify this attribute in number signs (#). For more information on the event structure, see Usage.
uid	delete getAttachments modify	Required		A case-sensitive Exchange UID value that uniquely identifies the tasks on which to perform the action. For the delete action, this attribute can be a comma-delimited list of UID values. The deleteAttachments, getAttachments, and modify actions allow only a single UID value.

When you specify the create or modify action, the task attribute must specify a structure that contains information that defines the events. The structure can have the following fields. You have to include only the fields that you are setting or changing.

Column	Description
ActualWork	A number in minutes. Cannot be less than zero.
Attachments	The pathnames of any attachments to include in the task. To specify multiple files, separate filepaths with semicolons (;) for Windows, and colons (:) for UNIX and Linux. You must use absolute paths. If you specify one or more attachments for a modify action, these are added to any existing attachments; the pre-existing attachments are not deleted.
BillingInfo	A string.
Companies	A string.
DateCompleted	A string in a date format that is valid in ColdFusion. If you omit this field and set the Status field to completed, or set the PercentCompleted field to 100, this value is set to the current date. If you set this date, the Status value is set to Completed and the PercentCompleted field is set to 100.
DueDate	A string in a date format that is valid in ColdFusion.
Message	A string containing the task description.
Mileage	A string.
PercentCompleted	A number in the range 0–100. If you set this field to 100, The following values are set: • The Status value is set to Completed. • If the DateCompleted value is or was not set, it is set to the current date. If you set this value to a number with a value less than 100, the following values are set: • If Status field is or was set to Completed, the Status is set to In_Progress. • The DateCompleted value is cleared.
Priority	One of the following values: • low • normal • high
ReminderDate	A string in a date format that is valid in ColdFusion.

Column	Description
StartDate	A string in a date format that is valid in ColdFusion. When you create a task, the default value `defaults` is the current date.
Status	The following values are valid: `Not_Started`, `In_Progress`, `Completed`, `Waiting`, or `Deferred`. If you omit this field and the `PercentCompleted` value is less than 100, the `Status` value it is set to `In_Progress`. If you set this field to `Completed`, the following values are also set: • The `PercentCompleted` value is set to 100. • If the `DateCompleted` value is not set, it is set to the current date. If you set this field to a value other than `Completed`, the following values are also set: • If the `PercentCompleted` field is or was 100, the `PercentCompleted` value is reset to 0. • The `DateCompleted` value is set to 0.
Subject	A String.
TotalWork	A number in minutes. Cannot be less than zero.

Usage

The `cfexchangetask` tag manages task records on the Exchange server. Use the `cfexchangetask` tag to perform the following actions:

- Create a task.
- Delete one or more task.
- Get one or more task records that conform to an optional set of filter specifications, such as the last name, job title, or home phone number, and so on.
- Get the attachments for a specific task record.
- Modify an existing task

To use this tag, you must have a connection to an Exchange server. If you are using multiple tags that interact with the exchange server, such as if you are creating several task records, you should use the `cfexchangeconnection` tag to create a persistent connection. You then specify the connection identifier in each `cfexchangetask`, or any other ColdFusion Exchange tag, if you are also accessing calendar entries, contacts, or mail. Doing this saves the overhead of creating and closing the connection for each tag.

Alternatively, you can create a temporary connection that lasts only for the time that ColdFusion processes the single `cfexchangetask` tag. To do this, you specify the connection attributes directly in the `cfexchangetask` tag. For details on the connection attributes, see the `cfexchangeconnection` tag.

The delete action

When you specify the `delete` action, you must specify a `uid` attribute with a comma-delimited list of one or more Exchange UIDs that identify the tasks to delete. You can use the `get` action, with an appropriate filter expression, to determine the UID values to specify.

If all UIDs that you specify are invalid, the `cfexchangetask` tag generates an error. If at least one UID is valid, the tag ignores any invalid UIDs and deletes the items specified by the valid UID.

The get action

When you specify the `get` action, the query object specified by the `name` attribute contains one record for each retrieved task. The query object has columns with the same names and data formats as the fields listed for the `task` attribute structure, with the following differences:

- The query object has a Boolean `HasAttachment` column, and does not have an `Attachments` column. If the `HasAttachment` field value is `yes`, use the `getAttachments` action to retrieve the attachments.
- The query object has an additional `UID` column with the unique identifier for the task in the Exchange server. You can use this value in the `uid` attribute of the `getAttachments`, `delete`, and `modify` actions to identify the required task.
- The query object has an additional `HtmlMessage` column. The `Message` column has a plain-text version of the task description, and the `HtmlMessage` column text includes the description's HTML formatting.

You use child `cfexchangefilter` tags to specify the messages to get. For detailed information, see `cfexchange-filter`.

The getAttachments action

When you use the `getAttachments` action, you must specify a single UID and a `name` attribute. The `cfexchangetask` tag populates a query object specified by the `name` attribute with the specified name. Each record has the following information about an attachment to the specified task:

Column name	Description
attachmentFileName	The filename of the attachment.
attachmentFilePath	The absolute path of the attachment file on the server. If you omit the `attachmentPath` attribute, this column contains the empty string.
CID	The content-ID of the attachment. Typically used in HTML img tags to embed images in a message.
mimeType	The MIME type of the attachment, such as text/html
isMessage	A Boolean value that specifies whether the attachment is a message.
size	The attachment size, in bytes.

If you omit the `attachmentPath` attribute, ColdFusion does not get any attachments; it gets only the information about the attachments. This lets you determine the event's attachments without incurring the overhead of getting the attachment files.

The modify action

If you specify the `modify` action, the `uid` attribute must specify a single Exchange UID. The `task` structure needs to specify only the fields that you are changing. Any fields that you do not specify remain unchanged. For a detailed description of the contents of the task structure, see the Attributes section.

If a task has attachments and you specify attachments when you modify the task, the new attachments are added to the previous attachments, and do not replace them. You must use the `deleteAttachments` action to remove any attachments.

Example

The following example uses a transient connection to create a single task:

```
<!--- Create a structure with the task fields --->
<cfscript>
    stask=StructNew();
```

```
    stask.Priority="high";
    stask.Status="Not_Started";
    stask.DueDate="3:00 PM 09/14/2007";
    stask.Subject="My New Task";
    stask.PercentCompleted=0;
    Message="Do this NOW!";
</cfscript>

<!--- Create the task using a transient connection. --->
<cfexchangetask action="create"
    username ="#user1#"
    password="#password1#"
    server="#exchangeServerIP#"
    task="#stask#"
    result="theUID">

<!--- display the UID to confirm that the action completed. --->
<cfdump var="#theUID#">
```

cfexecute

Description

Executes a ColdFusion developer-specified process on a server computer.

Category

Extensibility tags, Flow-control tags

Syntax

```
<cfexecute
    name = "application name"
    arguments = "command line arguments"
    outputFile = "output filename"
    timeout = "timeout interval"
    variable = "variable name">
    ...
</cfexecute>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcollection`, `cfindex`, `cfobject`, `cfreport`, `cfsearch`, `cfwddx`

History

ColdFusion MX 6.1:

- Added the `variable` attribute.
- Changed filepath behavior for the `outputFile` attribute: if you do not specify an absolute filepath in the `outputFile` attribute, the path is relative to the ColdFusion temporary directory.

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		Absolute path of the application to execute. On Windows, you must specify an extension, for example, C:\myapp.exe.
`arguments`	Optional		Command-line variables passed to application. If specified as string, it is processed as follows: • Windows: passed to process control subsystem for parsing. • UNIX: tokenized into an array of arguments. The default token separator is a space; you can delimit arguments that have embedded spaces with double-quotation marks. If passed as array, it is processed as follows: • Windows: elements are concatenated into a string of tokens, separated by spaces. Passed to process control subsystem for parsing. • UNIX: elements are copied into an array of exec() arguments.
`outputFile`	Optional		File to which to direct program output. If no `outputfile` or `variable` attribute is specified, output is displayed on the page from which it was called. If not an absolute path (starting a with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.

Attribute	Req/Opt	Default	Description
timeout	Optional	0	Length of time, in seconds, that ColdFusion waits for output from the spawned program. • 0: equivalent to nonblocking mode. • A very high value: equivalent to blocking mode. If the value is 0: • ColdFusion starts a process and returns immediately. ColdFusion may return control to the calling page before any program output displays. To ensure that program output displays, set the value to 2 or higher. • If the outputFile attribute is not specified, any program output is discarded
variable	Optional		Variable in which to put program output. If no outputfile or variable attribute is specified, output is displayed on page from which it was called.

Usage

Do not put other ColdFusion tags or functions between the start and end tags of cfexecute. You cannot nest cfexecute tags.

Exceptions

Throws the following exceptions:

- If the application name is not found: java.io.IOException
- If the effective user of the ColdFusion executing thread does not have permissions to execute the process: a security exception

The time-out values must be between zero and the longest time-out value supported by the operating system.

Example

```
<h3>cfexecute</h3>
<p>This example executes the Windows NT version of the netstat network monitoring program,
and places its output in a file.

<cfexecute name = "C:\WinNT\System32\netstat.exe"
    arguments = "-e"
    outputFile = "C:\Temp\output.txt"
    timeout = "1">
</cfexecute>
```

cfexit

Description

This tag aborts processing of the currently executing CFML custom tag, exits the page within the currently executing CFML custom tag, or re-executes a section of code within the currently executing CFML custom tag.

Category

Debugging tags, Flow-control tags

Syntax

```
<cfexit
    method = "method">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfabort`, `cfbreak`, `cfexecute`, `cfif`, `cflocation`, `cfloop`, `cfswitch`, `cfthrow`, `cftry`; "cfabort and cfexit" on page 20 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`method`	Optional	`exitTag`	• `exitTag`: aborts processing of currently executing tag. • `exitTemplate`: exits page of currently executing tag. • `loop`: re-executes body of currently executing tag.

Usage

If this tag is encountered outside the context of a custom tag, for example in the base page or an included page, it executes in the same way as `cfabort`. The `cfexit` tag can help simplify error checking and validation logic in custom tags.

The `cfexit` tag function depends on its location and execution mode:

Method value	Location of cfexit call	Behavior
`exitTag`	Base page	Terminate processing
	Execution mode = Start	Continue after end tag
	Execution mode = End	Continue after end tag
`exitTemplate`	Base page	Terminate processing
	Execution mode = Start	Continue from first child in body
	Execution mode = End	Continue after end tag
`loop`	Base page	Error
	Execution mode = Start	Error
	Execution mode = End	Continue from first child in body

Example

```
<h3>cfexit Example</h3>
```

```
<p>cfexit can be used to abort the processing of the currently executing CFML custom tag.
Execution resumes following the invocation of the custom tag in the page that called the tag.
<h3>Usage of cfexit</h3>
<p>cfexit is used primarily to perform a conditional stop of processing inside a custom tag.
cfexit returns control to the page that called that custom tag, or in the case of a tag
called by another tag, to the calling tag.</p>

<!--- cfexit can be used within a CFML custom tag, as follows: --->
<!--- Place this code (uncomment the appropriate sections) within the customtags directory.
--->

<!--- MyCustomTag.cfm --->
<!--- This simple custom tag checks for the existence of myValue1 and myValue2. If they are
both defined, the tag adds them and returns the result to the calling page in the variable
"result". If either or both of the expected attribute variables is not present, an error
message is generated, and cfexit returns control to the calling page. --->

<!--- <cfif NOT IsDefined("attributes.myValue2")>
            <cfset caller.result = "Value2 is not defined">
            <cfexit method = "exitTag">
     <cfelseif NOT IsDefined("attributes.myValue1")>
            <cfset caller.result = "Value1 is not defined">
            <cfexit method = "exitTag">
     <cfelse>
            <cfset value1 = attributes.myValue1>
            <cfset value2 = attributes.myValue2>
            <cfset caller.result = value1 + value2>
     </cfif> --->
<!--- End MyCustomTag.cfm --->

<!--- Place this code within your page --->

<!--- <p>The call to the custom tag, and then the result: </p>
<CF_myCustomTag
        myvalue2 = 4>
<cfoutput>#result#</cfoutput> --->
<p>If cfexit is used outside a custom tag, it functions like a cfabort. For example, the
text after this message is not processed:</p>
<cfexit>
<p>This text is not executed because of the cfexit tag above it.</p>
```

cffeed

Description

Reads or creates an RSS or Atom syndication feed. This tag can read RSS versions 0.90, 0.91, 0.92, 0.93, 0.94, 1.0, and 2.0, and Atom 0.3 or 1.0. It can create RSS 2.0 or Atom 1.0 feeds.

Category

Communications tags, Internet protocol tags

Syntax

```
create
    required
    <cffeed
    action = "create"
    name = "#structure#"
        One or both of the following:
    outputFile = "path"
    xmlVar = "variable name"
    optional
    overwrite = "no|yes">

OR

    required
    <cffeed
    action = "create"
    properties = "#metadata structure#"
    query = "#items/entries query name#"
        One or both of the following:
    outputFile = "path"
    xmlVar = "variable name"
    optional
    columnMap = "mapping structure"
    overwrite = "no|yes">

read
    required
    <cffeed
    source = "feed source"
        One or more of the following:
    name = "structure"
    properties = "metadata structure"
    query = "items/entries query"
    outputFile = "path"
    xmlVar = "variable name"
    optional
    action = "read"
    enclosureDir = "path"
    ignoreEnclosureError = "no|yes"
    overwrite = "no|yes"
    overwriteEnclosure = "no|yes"
    proxyServer = "IP address or server name for proxy host"
    proxyPassword = "password for the proxy host"
    proxyPort = "port of the proxy host"
    proxyUser = "user name for the proxy host"
    timeout = "request time-out in seconds"
    userAgent = "HTTP user agent identifier">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Optional	read	The action to take, one of the following values: • `create`: creates an RSS 2.0 or Atom 1.0 feed XML document and saves it in a variable, writes it to a file, or both. • `read`: parses an RSS or Atom feed from a URL or an XML file and saves it in a structure or query. You can also get feed metadata in a separate structure.
`columnMap`	Optional		Used only for the `create` action with a `query` attribute. A structure that specifies a mapping between the names of the columns in the object specified by the `query` attribute and the columns of the ColdFusion feed format (see the Query object rules section. The key for each field must be a column name (see the table in the Query object rules section). The value of the field must be the name of the corresponding column in the query object used as input to the `create` action.
`enclosureDir`	Optional		Used only for the `read` action. Path to the directory in which to save any enclosures that are available in the feed being read. The path can be absolute or relative to the CFML file. If the directory does not exist, ColdFusion generates an error. If you omit this attribute, ColdFusion does not save enclosures. To specify the directory that contains the current page, set this attribute to " . " (period).
`ignoreEnclosureError`	Optional	no	If this attribute is `yes`, ColdFusion attempts to save all enclosures. If it encounters an error downloading one enclosure, it continues downloading other enclosures and writes the error information in the server log. If this attribute is `no`, when ColdFusion encounters an error downloading an enclosure, it stops downloading all enclosures and generates an error. **Note:** Enclosure errors can occur if the specified enclosure is of a type that the web server does not allow to be downloaded.
`name`	See Note		A structure that contains complete feed data: • The output of a `read` action. • The input definition of the feed to create. When you specify the `name` attribute for a `create` action, you must enclose it in number signs (#). For more information, see "Name and properties structure rules" on page 189.
`outputFile`	See Note		Path of the file in which to write the feed as XML text. The path can be absolute, or relative to the CFML file.
`overwrite`	Optional	no	Whether to overwrite the XML feed file if it already exists. If you do not set this attribute to `yes` and the `cffeed` tag tries to write to a file that exists, ColdFusion generates an error.

Attribute	Req/Opt	Default	Description
overwriteEnclosure	Optional	no	Used only for the read action. Whether to overwrite files in the enclosure directory if they already exist. If you do not set this attribute to yes and the cffeed tag tries to write to a file that exists, ColdFusion generates an error.
properties	See Note		A structure that contains the feed metadata, the information about the entire feed. Can contain either of the following: • The output of a read action. • Input to a create action. The properties and query attributes combined provide complete feed information. When you specify the properties attribute for a create action, you must enclose it in number signs (#). For more information, see "Name and properties structure rules" on page 189.
proxyPassword	Optional		Password required by the proxy server.
proxyPort	Optional	80	The port to connect to on the proxy server.
proxyServer	Optional		Host name or IP address of a proxy server to which to send the request.
proxyUser	Optional		User name to provide to the proxy server.
query	See Note		A query object that contains the Atom entries or RSS items in the feed. Can contain either of the following: • The output of a read action. • Input to a create action. The properties and query attributes combined provide complete feed information. When you specify the query attribute for a create action, you must enclose it in number signs (#). For more information, see "Query object rules" on page 190.
source	Required		Used only for the read action. The URL of the feed or the path to the XML file that contains the feed contents. A path can be absolute, or relative to the CFML file.
timeout	Optional	Request time-out	The number of seconds to wait for a response from the feed source. A value of 0 specifies that the request does not time out. By default, ColdFusion uses the request time-out setting of the ColdFusion Administrator Server Settings > Settings page.
userAgent	Optional	Cold Fusion	Text to put in the HTTP User-Agent request header field. Used to identify the request client software.
xmlVar	See Note		A variable in which to save the read or created feed as XML text.

Usage

Setting and getting feed information

The cffeed tag lets you specify and save feed data in many, flexible ways.

When you create a feed

- You specify the feed data in either of the following ways:
  - By putting all metadata and entry or item data in a single structure specified by the name attribute.

  - By putting the metadata in a structure specified by the `properties` structure and the entries or items as rows in a query object specified by the `query` attribute .
- You save the resulting feed XML in one or both of the following places:
  - A file specified by the `OutputFile` attribute. The `cffeed` tag saves the data in UTF-8 encoding.
  - An variable specified by the `xmlVar` attribute

When you read a feed

You can save the feed data in any combination of the following forms:

- By saving all entry or item data and metadata in a single structure specified by the `name` attribute
- By saving entries or items as rows in a query object specified by the `query` attribute
- By saving the metadata in a structure specified by the `properties` structure
- By writing the feed XML in a file specified by the by the `OutputFile` attribute. The `cffeed` tag saves the data in UTF-8 encoding.
- By saving the feed XML in a ColdFusion XML variable specified by the `xmlVar` attribute

When you save feed data, you do not have to save both the metadata and the entry or item data. You can specify only the `properties` attribute, or only the `query` attribute.

Name and properties structure rules

The `name` and `properties` structures must conform to the following rules. For more information on requirements for specific metadata entries, see "Representing feed metadata" on page 192.:

- All structure key names must be identical to the corresponding feed element names, with the exception of the `version` and `encoding` fields. Also, the key names for Dublin Core and Apple® iTunes extension elements start with `DC_` and `ITUNES_` respectively.
- The `properties` structure fields are identical to the metadata fields in the `name` structure.
- When you read a feed, the structure contains only those elements and attribute values that exist in the feed. For requirements for the `create` action, see "Creating feeds" on page 194.
- If the feed can have multiple elements of the same type (such as `entry`, `item`, or `link`), the `name` or `property` structure has a single entry that contains the data for all of the elements. The structure entry has the following format:
  - The key is the element name (for example, `item`)
  - The value is an array of structures
  - Each structure in the array represents one element.

  ColdFusion uses an array even if there is only a single element. If an Atom feed has only one `link` element, for example, you must specify that element in a `name` attribute structure by using the following format:

  ```
  structureName.link[1]
  ```

  For example, to specify a `link` metadata entry in an Atom 1.0 feed, you could use the following code:

  ```
  <cfset meta.link = arrayNew(1)>
  <cfset meta.link[1] = structNew()>
  <cfset meta.link[1].href = "http://www.myCo,com">
  ```

- If an element can have multiple attributes, or can have at least one attribute and a value, the element is represented as a structure, even if the element specifies only one attribute or only a value.

- If an element has one or more attributes and a value (body), the value is in a field of the element structure named `value`. For example, the text of the `summary` element for the third `entry` in an Atom feed would go in a field whose name has the following format:

```
structureName.entry[3].summary.value.
```

- When the `cffeed` tag reads a feed, it reports dates as follows:

  Atom: W3C date format, such as 2006-07-11T18:19:00Z.

  RSS: in RFC 822 Format, such as Thu, 05 Oct 2006 18:19:00 GMT.

- When the `cffeed` tag creates a feed, you can use W3C or RFC 822 formats for both feed types. You can also use any standard date or date/time format accepted by ColdFusion.

Query object rules

The query object specified by the `query` attribute conforms to the following rules:

- The query object format supports multiple feed formats, and many feeds do not include all optional feed attributes or elements. As a result:
  - When you read a feed, the returned query object contains entries for all standard RSS and Atom fields, even for fields that are not supported by the feed type. Any columns that are not used by the feed format, or are not used in that specific feed, contain empty strings or undefined values.
  - When you read a feed, the query object contains all iTunes extension fields if the feed contains any iTunes extension elements, and the query object contains all Dublin Core extension fields if the feed contains any Dublin Core extension elements. Otherwise, the query results do not contain any of the extension fields.
  - When you create a feed, the query that you define requires only those columns that contain data for your feed; you can omit unused columns.
- If a feed entry or item has multiple child elements with the same name, the query column represents the element values as a comma-delimited list. RSS 2.0 items can have multiple `category` elements. Atom 1.0 entries can have multiple `category`, `author`, `contributor`, and `link` elements. The Dublin Core extensions allow all multiples of all element types.
- Many entry or item elements that can have multiple instances have multiple attributes, not all of which are required for any particular element instance. If an entry or item has multiple instances of an element, and any of those elements omit attributes, ColdFusion represents the omitted attribute in the lists by a space. In XML, an Atom entry, for example, might contain three author elements, as follows:

```
<author>
    <person>Anthony</person>
    <uri>http://www.MyCo.com</uri>
    <email>Tony@MyCo.com</email>
</author>
<author>
    <person>Beverly</person>
</author>
<author>
    <person>Cathy</person>
    <email>cathy@MyCo.com</email>
</author>
```

The ColdFusion query represents these columns as follows:

AUTHOR_PERSON	AUTHOR_URI	AUTHOR_EMAIL
Anthony,Beverly,Cathy	http://www.MyCo.com, ,	Tony@MyCo.com, ,cathy@MyCo.com

The following table lists the columns of the standard query object specified by the `query` attribute. If an RSS feed includes either Dublin Core extensions or iTunes extensions, the query includes additional columns. For information on these fields, see Dublin Core Extensions and Apple iTunes Extensions.

Column	Atom entry	RSS item
AUTHOREMAIL	`author` element `email` attribute	`author` item
AUTHORNAME	`author` element `name` attribute	Not used
AUTHORURI	`author` element `uri` attribute	Not used
CATEGORYLABEL	`category` element `label` attribute	`category` item value
CATEGORYSCHEME	`category` element `scheme` attribute	`category` item `domain` attribute
CATEGORYTERM	`category` element `term` attribute	Not used
COMMENTS	Not used	`comments` item value
CONTENT	`content` element value	`description` item value
CONTENTMODE	`content` element `mode` attribute (Atom 0.3 only)	Not used
CONTENTSRC	`content` element `src` attribute	Not used
CONTENTTYPE	`content` element `type` attribute	Not used
CONTRIBUTOREMAIL	`contributor` element `email` attribute	Not used
CONTRIBUTORNAME	`contributor` element `name` attribute	Not used
CONTRIBUTORURI	`contributor` element `uri` attribute	Not used
CREATEDDATE	`created` element value (Atom 0.3 only)	Not used
EXPIRATONDATE	Not used	`expirationDate` item value (RSS 0.93 only)
ID	`id` element value	`guid` item value
IDPERMALINK	Not used	`guid` item `ispermalink` attribute
LINKHREF	`link` element `href` attribute	`enclosure` item `url` attribute
LINKHREFLANG	`link` element `hreflang` attribute	Not used
LINKLENGTH	`link` element `length` attribute	`enclosure` item `length` attribute
LINKREL	`link` element `rel` attribute	Not used
LINKTITLE	`link` element `title` attribute	Not used
LINKTYPE	`link` element `type` attribute	`enclosure` item `type` attribute
PUBLISHEDDATE	`published` element value (`issued` in Atom 0.3)	`pubDate` item value
RIGHTS	`rights` element value (`copyright` in Atom 0.3)	Not used
RSSLINK	Not used	`link` item value
SOURCE	Not used	`source` item value
SOURCEURL	Not used	`source` item `url` attribute
SUMMARY	`summary` element value	Not used
SUMMARYMODE	`summary` element `mode` attribute (Atom 0.3 only)	Not used

Column	Atom entry	RSS item
SUMMARYSRC	Blank for all well-formed Atom feeds. Contains data only if an Atom 1.0 feed uses a `content` element format for the `summary` element.	Not used
SUMMARYTYPE	`summary` element `type` attribute	Not used
TITLE	`title` element value	`title` item value
TITLETYPE	`title` element `type` attribute	Not used
UPDATEDDATE	`updated` element value (`modified` in Atom 0.3)	Not used
URI	Not used	RSS 1.0 `link` item `rdf:about` attribute
XMLBASE	`content` element `xml:base` attribute	Not used

Representing feed metadata

When you create a feed, the `name` and `properties` structures can represent all standard metadata for RSS 2 or Atom 1 feeds, in the format described in the Name and properties structure rules section. Similarly, when you read a feed, the structures represent all received metadata. The following rules apply to specific feed metadata fields in the `name` and `properties` structures:

- The `version` field identifies or specifies the feed version in the form *format_versionNumber*. For the `create` action, you must specify `atom_1.0` or `rss_2.0`. When you read an RSS 0.91 feed, the version field value is `rss_0.91U`, not `rss_0.91`.
- The `feedExtension` field identifies whether the feed includes iTunes or Dublin Core extension content. Valid values are `itunes` and `DublinCore`. You do not have to specify this field when you create a feed with iTunes extensions; ColdFusion automatically determines that you have specified extension fields. (You cannot create a feed with Dublin Core extensions.)
- For the `read` action, an `encoding` field identifies the XML encoding attribute, such as `iso-8859-1`. Do not specify an `encoding` field for a `create` action. Currently, ColdFusion generates all feeds in UTF-8 format and ignores any `encoding` value that you specify.
- For RSS feeds, the `skiphours` field contains a comma-delimited list of up to 24 numbers in the range 0–23, specifying hours of the day when aggregators should not read the feed. The hour beginning at midnight is hour zero. Your application can use the field to decide when to read the feed.
- For RSS feeds, the `skipdays` field contains a comma-delimited list of up to seven day-name values, specifying days of the week when aggregators should not read the feed. The valid names are `Monday`, `Tuesday`, `Wednesday`, `Thursday`, `Friday`, `Saturday` and `Sunday`. Your application can use the field to decide when to read the feed.

Dublin Core Extensions

Dublin Core extension elements provide additional metadata about the feed or an item. You can use the `cffeed` tag to read feeds that include elements that conform to the Dublin Core Metadata Element Set specification as metadata (channel elements) or as item elements. For detailed information Dublin Core extension elements, see the Dublin Core Metadata Element Set specification. At the time this topic was written, this specification was available at http://dublincore.org/documents/dces/.

ColdFusion support for Dublin Core extensions has the following limitations:

- You cannot create feeds containing these elements.
- You cannot get Dublin Core extension elements that are contained in a top-level (metadata) `image` element. ColdFusion ignores these elements.

• ColdFusion supports only the Dublin Core Metadata Element Set. It does not support the additional Dublin Core Metadata Initiative elements and element refinements.

When feed items include the Dublin Core extensions, the query specified by a `query` attribute includes all of the columns listed in the following table. If the feed does not include any Dublin Core extension elements, the query does not include the columns. With the exception of the `DC_SUBJECT_TAXONOMURI` and `DC_SUBJECT_VALUE` columns, each column name (without the `DC_` prefix) corresponds directly to a Dublin Core extension element name.

Column	Description
DC_CONTRIBUTOR	The people or organizations responsible for contributing to the resource
DC_COVERAGE	The extent of the content in the resource
DC_CREATOR	The person or organization responsible for creating this resource
DC_DATE	A date or date and time associated with this resource
DC_DESCRIPTION	A summary of the resource contents
DC_FORMAT	The file format, physical medium, or dimensions of the resource
DC_IDENTIFIER	A string that can be used to unambiguously identify the resource
DC_LANGUAGE	The language in which the resource is written
DC_PUBLISHER	The person or organization responsible for making the resource available.
DC_RELATION	The identifier of a related resource, typically.
DC_RIGHT	Information about the property rights for the resource.
DC_SOURCE	A reference to the material from which this resource was derived.
DC_SUBJECT_TAXONOMYURI	The `taxonomy URI` attribute of the Dublin Core `subject` element
DC_SUBJECT_VALUE	The value of the Dublin Core `subject` element; a string that the topic of the resource
DC_TITLE	A name to use for the resource
DC_TYPE	The nature or genre of the resource

When you get data for a feed that includes Dublin Core elements as a structure, the element names are identical to the query column names listed above, with the exception of the representation of the Dublin Core `subject` element. The structure format represents the `subject` element as a `dc_subject` entry, which consists of an array of structures. The structures in the array have keys with the names `value`, for the element value, and `taxononmyURI`, for the `taxonomyURI` attribute.

Apple iTunes Extensions

You can use the `cffeed` tag to create or read feeds that contain elements defined in the Apple iTunes RSS podcast specification. For detailed information on iTunes extension format, see the Apple iTunes RSS specification. At the time this topic was written, this specification was available at http://www.apple.com/itunes/store/podcaststechspecs.html.

You can create feeds with only a subset of the iTunes RSS extensions. When you read a feed, ColdFusion ignores all iTunes extension elements that are not in the supported subset.

The following table lists the names of structure entries or query column names for the supported elements. (These names consist of the `ITUNES_` prefix followed by the iTunes extension element name.) The table also indicates which elements are used in the metadata, which are used in the individual items, and which can be used in both:

Element	Used in	Description
ITUNES_AUTHOR	Both	Artist name
ITUNES_BLOCK	Both	a value of `yes` requests to prevent the podcast or item (episode) from appearing. When ColdFusion reads a feed your application should determine this field's value and take any appropriate action.
ITUNES_DURATION	Item	The length of the item in second, or in HH:MM:SS format.
ITUNES_EXPLICIT	Both	A string indicating whether the item or items contain explicit material. Valid values are `yes`, `no`, and `clean`.
ITUNES_KEYWORDS	Both	A comma-delimited list of words or phrases used when searching in the iTunes music store.
ITUNES_SUBTITLE	Both	Short description text, usually only a few words.
ITUNES_SUMMARY	Both	A longer description (up to 4000 characters)/

You can also use the following channel elements in the `name` or `properties` structures.

Element	Description
itunes_category	A structure that specifies the iTunes Music Store category. The structure has two fields: • `category` • `subcategory` Notice that these element names do *not* have the `itunes_` prefix.
itunes_image	The URL of the artwork for the podcast.
itunes_owner	A structure that contains contact information about the owner of the podcast for communication. The structure has two fields: • `itunes_email` • `itunes_mail`

Creating feeds

When you create a feed, you specify the feed contents in a `name` structure or in the combination of a `query` object and a `properties` structure. The `cffeed` tag generates the feed XML and saves in to the variable specified by the `xmlVar` attribute, the file specified by the `outputFile` attribute, or both.

To create an RSS 2.0 feed you must specify the following metadata fields in a `name` structure or in a `properties` structure. All other RSS2.0 metadata fields, and all item fields, are optional.

- title
- link
- description
- version (must be "rss_2.0")

The `cffeed` tag does not enforce any rules on the Atom feed structure that it creates. You are responsible for ensuring that the feed is valid.

In most cases, a database table uses column names that differ from the column names you must use to create the feed. Therefore, you must use the `columnmap` attribute to map the input query column names to the required column names. The attribute is a structure whose keys are the column names required by the `cffeed` tag and whose values are the corresponding input query columns. The following example creates a feed using the cfartgallery data source's orders table. It maps the orders table ORDERDATE column to the query publisheddate column, the ADDRESS column to the content column, and so on. The sample code then displays the generated query XML to show the results.

```
<!--- Get the feed data as a query from the orders table. --->
<cfquery name="getOrders" datasource="cfartgallery">
    SELECT * FROM orders
</cfquery>

<!--- Map the orders column names to the feed query column names. --->
<cfset columnMapStruct = StructNew()>
<cfset columnMapStruct.publisheddate = "ORDERDATE">
<cfset columnMapStruct.content = "ADDRESS">
<cfset columnMapStruct.title = "CUSTOMERFIRSTNAME">
<cfset columnMapStruct.rsslink = "ORDERID">

<!--- Set the feed metadata. --->
<cfset meta.title = "Art Orders">
<cfset meta.link = "http://feedlink">
<cfset meta.description = "Orders at the art gallery">
<cfset meta.version = "rss_2.0">

<!--- Create the feed. --->
<cffeed action="create"
    query="#getOrders#"
    properties="#meta#"
    columnMap="#columnMapStruct#"
    xmlvar="rssXML">

<cfdump var="#XMLParse(rssXML)#">
```

Reading feeds

The `cffeed` tag does not validate the feeds that it reads. It can read invalid or loosely formatted feeds, but ignores some or all of the invalid content. For example, if you put more than one `rights` element in the Atom feed (which invalidates the feed), the `cffeed` tag ignores the elements after the first one, and doesn't generate an error.

Dates and times in feeds that are being read must be in W3C or RFC 822 format. ColdFusion can also read iTunes extension dates in the format normally used by the iTunes music store.

Example

The following example creates an RSS feed. You must enter fields for the feed title, link, and description elements. You must also enter title, link, and description fields for one item. A second item is optional. The application saves the feed in a createRSSOutput.xml file in the feedTest subdirectory of the directory that contains the CFML page.

```
<!--- Generate the feed when the user submits a filled in form. --->
<cfif isDefined("Form.Submit")>
    <cfscript>

        // Create the feed data structure and add the metadata.
        myStruct = StructNew();
        mystruct.link = form.link;
        myStruct.title = form.title;
        mystruct.description = form.description;
        mystruct.pubDate = Now();
        mystruct.version = "rss_2.0";

        /* Add the feed items. A more sophisticated application would use dynamic variables
            and support varying numbers of items. */
        myStruct.item = ArrayNew(1);
        myStruct.item[1] = StructNew();
        myStruct.item[1].description = StructNew();
        myStruct.item[1].description.value = form.item1text;
        myStruct.item[1].link = form.item1link;
```

```
        myStruct.item[1].pubDate = Now();
        myStruct.item[1].title = form.item1title;
        myStruct.item[2] = StructNew();
        myStruct.item[2].description = StructNew();
        myStruct.item[2].description.value = form.item2text;
        myStruct.item[2].link = form.item2link;
        myStruct.item[2].pubDate = Now();
        myStruct.item[2].title = form.item2title;

    </cfscript>

    <!--- Generate the feed and save it to a file and variable. --->
    <cffeed action = "create"
        name = "#myStruct#"
        outputFile = "feedTest/createRSSOutput.xml"
        overwrite = "yes"
        xmlVar = "myXML">

</cfif>

<!--- The user input form. --->
<cfform format="xml" preservedata="yes" style="width:500" height="700">
    <cfformitem type = "text"> Enter The Feed Metadata</cfformitem>
    <cfinput type = "text" label = "title" name = "title"
        style = "width:435" required = "yes"> <br />
    <cfinput type = "text" label = "link" name = "link"
        style = "width:435" required = "yes" validate = "url"> <br />
    <cftextarea name = "description"
        style = "width:435; height:70" required = "yes" />

    <cfformitem type = "text"> Enter Item 1</cfformitem>
    <cfinput type="text" label="title" name="item1title"
        style="width:435" required="yes"> <br />
    <cfinput type="text" label="link" name="item1link"
        style="width:435" required="yes" validate="url"> <br />
    <cftextarea name = "item1text"
        style = "width:435; height:70" required = "yes" /> <br />

    <cfformitem type = "text"> Enter Item 2</cfformitem>
    <cfinput type = "text" label = "title" name = "item2title" style = "width:435"> <br />
    <cfinput type = "text" label = "link" name = "item2link" style = "width:435"
        validate = "url"> <br />
    <cftextarea name = "item2text" style = "width:435; height:70" /> <br />

    <cfinput type = "Submit" name = "submit" value = "Submit" >
</cfform>
```

The following application is a simple feed reader that handles RSS and Atom feeds. It displays the feed title; for each item or entry, it shows the title as a link, and shows the published date and the item or entry contents. To use this example to read the feed created by the first application, enter the URL for the file the application created, for example, http://localhost:8500/cffeed/feedTest/createRSSOutput.xml.

```
<!--- Process the feed data if the user submitted the form --->
<cfif isDefined("Form.Submit")>
    <cffeed source = "#theURL#"
        properties = "myProps"
        query = "myQuery">

    <!--- Display the feed output.
            Use conditional logic for to handle different feed formats. --->
    <cfoutput>
```

```
        <h2>#myProps.title#</h2>
    </cfoutput>
    <cfoutput query = "myQuery">
        <cfif myProps.version IS "atom_1.0">
            <h3><a href = "#linkhref#">#title#</a></h3>
            <p><b>Published:</b> #DateFormat(publisheddate)#</p>
        <cfelse>
            <h3><a href = "#rsslink#">#title#</a></h3>
            <p><b>Published:</b> #publisheddate#</p>
        </cfif>
        <p>#content#</p>
    </cfoutput>
</cfif>

<!--- The form for specifying the feed URL or file --->
<cfform name = "SetFeed" preserveData = "yes">
        Enter Feed URL:
        <cfinput type = "text" size = "60" name = "theURL"><br><br>
        <cfinput type = "Submit" name = "submit" value = "Submit">
</cfform>
```

cffile

Description

Manages interactions with server files.

The following sections describe the actions of the `cffile` tag:

- "cffile action = "append"" on page 201
- "cffile action = "copy"" on page 203
- "cffile action = "delete"" on page 204
- "cffile action = "move"" on page 205
- "cffile action = "read"" on page 207
- "cffile action = "readBinary"" on page 209
- "cffile action = "rename"" on page 210
- "cffile action = "upload"" on page 212
- "cffile action = "write"" on page 215

Note: *To execute, this tag must be enabled in the ColdFusion Administrator. For more information, see Configuring and Administering ColdFusion.*

If your ColdFusion applications run on a server used by multiple customers, consider the security of the files that could be uploaded or manipulated by `cffile`. For more information, see *Configuring and Administering ColdFusion*.

Category

File management tags

Syntax

The tag syntax depends on the `action` attribute value. See the following sections.

See also

`cfdirectory`

History

ColdFusion 8: Support for reading and writing `cfimages`.

ColdFusion MX 7:

- Added the `result` attribute, which allows you to specify an alternate variable in which to receive result parameters. Used for `action = "upload"` action.
- Added the `fixnewline` attribute for `action = "append"` and `action = "write"` actions.

ColdFusion MX 6.1:

- Changed file path requirements: if you do not specify an absolute file path, the path is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
- Changed behavior for `action="read"`: if the file starts with a byte order mark (BOM) ColdFusion uses it to determine the character encoding.

• Changed behavior for `action="upload"` `nameConflict="MakeUnique"` ColdFusion now makes filenames unique by appending a incrementing number, 1 for the first file, 2 for the second and so on, to the name. In ColdFusion, filenames were made unique by appending an additional "1" for each file, as in 1, 11, 111, and so on.

ColdFusion MX:

• Changed use of slashes in paths: you can use forward (/) or backward (\) slashes in paths on both UNIX and Windows systems.

• Changed file hierarchy requirements: ColdFusion does not require that you put files and directories that you manipulate with this tag below the root of the web server document directory.

• Changed directory path requirements for the `destination` attribute: a directory path that you specify in the `destination` attribute does not require a trailing slash.

• Deprecated the `system` value of the `attributes` attribute.

• Deprecated the `temporary` value of the `attributes` attribute. In ColdFusion, it is a synonym for `normal`. It might not work in later releases.

• Changed the `action` attribute options `read`, `write`, `append` and `move`: they support a new attribute, `charset`.

• The `archive` value of the `attributes` attribute is obsolete and has no effect.

Example

```
<!--- This shows how to write, read, update, and delete a file using CFFILE.
This is a view-only example. --->
<!---
<cfif IsDefined("form.formsubmit") is "Yes">
    <!--- The form has been submitted, now do the action. --->
    <cfif form.action is "new">
        <!--- Make a new file. --->
        <cffile action="Write"
            file="#GetTempDirectory()#foobar.txt"
            output="#form.the_text#">
    </cfif>
    <cfif form.action is "read">
        <!--- Read existing file. --->
        <cffile action="Read"
            file="#GetTempDirectory()#foobar.txt"
            variable="readText">
    </cfif>

    <cfif form.action is "add">
        <!--- Update existing file. --->
        <cffile action="Append"
            file="#GetTempDirectory()#foobar.txt"
            output="#form.the_text#">
    </cfif>

    <cfif form.action is "delete">
        <!--- Delete existing fil. --->
        <cffile action="Delete"
            file="#GetTempDirectory()#foobar.txt">
    </cfif>
</cfif>
<!--- Set some variables. --->
<cfparam name="fileExists" default="no">
<cfparam name="readText" default="">
<!--- First, check whether canned file exists. --->
<cfif FileExists("#GetTempDirectory()#foobar.txt") is "Yes">
```

```
    <cfset fileExists="yes">
</cfif>
<!--- Now, make the form that runs the example. --->
<form action="index.cfm" method="POST">
<h4>Type in some text to include in your file:</h4> <p>
<cfif fileExists is "yes">
    <p>A file exists (foobar.txt, in <cfoutput>#GetTempDirectory()#</cfoutput>).
    You may add to it, read from it, or delete it. </p>
</cfif>
<!--- If reading from a form, let that information display in textarea. --->
<textarea name="the_text" cols="40" rows="5">
    <cfif readText is not "">
        <cfoutput>#readText#</cfoutput>
    </cfif></textarea>
<!--- Select from the actions depending on whether the file exists. --->
<select name="action">
<cfif fileExists is "no">
    <option value="new">Make new file
</cfif>
<cfif fileExists is "yes">
    <option value="add">Add to existing file
    <option value="delete">Delete file
    <option value="read">Read existing file
</cfif>
</select>
<input type="Hidden" name="formsubmit" value="yes">
<input type="Submit" name="" value="make my changes">
</form> --->
```

cffile action = "append"

Description

Appends text to a text file on the server.

Syntax

```
<cffile
    action = "append"
    file = "full pathname"
    output = "string"
    addNewLine = "yes|no"
    attributes = "file attributes list"
    charset = "characterset option"
    fixnewline = "yes|no"
    mode = "mode">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdirectory`

History

See the History section of the main `cffile` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`file`	Required		Pathname of the file to which to append content of `output` attribute. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`output`	Required		String to append to the file.
`addNewLine`	Optional	`yes`	• `yes`: appends newline character to text written to file. • `no`
`attributes`	Optional		Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify `attributes = "readOnly"`, all other attributes are overwritten. • `readOnly` • `hidden` • `normal`

Attribute	Req/Opt	Default	Description
charset	Optional	JVM default file character set	The character encoding in which the file contents is encoded. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information character encodings, see www.w3.org/International/O-charset.html.
fixnewline	Optional	No	• yes: changes embedded line-ending characters in string variables to operating-system specific line endings • no: (default) do not change embedded line-ending characters in string variables. For an example that uses this attribute, see cffile action = "write".
mode	Optional		Applies only to UNIX and Linux. Permissions. Octal values of UNIX chmod command. Assigned to owner, group, and other, respectively; for example: • 644: assigns read/write permission to owner; read permission to group and other. • 777: assigns read/write/execute permission to all.

Example

```
<!--The first example creates the file \temp\foo on a windows system and sets attributes to
normal. --->
<cffile action = "write" file = "\temp\foo" attributes = normal output = "some text">

<!--- The second example appends to the file. --->
<cffile action = "append" file = "\temp\foo" attributes = normal output = "Is this a test?">
```

cffile action = "copy"

Description

Copies a file from one directory to another on the server.

Syntax

```
<cffile
    action = "copy"
    destination = "full pathname"
    source = "full pathname"
    attributes = "file attributes list"
    mode = "mode">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdirectory`

History

See the History section of the main `cffile` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`destination`	Required		Pathname of a directory or file on web server where the file is copied. If you specify a filename without a directory path, ColdFusion copies it relative to the source directory.
`source`	Required		Pathname of the file to copy. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`attributes`	Optional		Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify `attributes = "readOnly"`, all other attributes are overwritten. • `readOnly` • `hidden` • `normal`
`mode`	Optional		Applies only to UNIX and Linux. Permissions. Octal values of UNIX `chmod` command. Assigned to owner, group, and other, respectively; for example: • 644: assigns read/write permission to owner; read permission to group and other. • 777: assigns read/write/execute permission to all.

Example

This example copies the keymemo.doc file to the c:\files\backup\ directory:

```
<cffile action = "copy" source = "c:\files\upload\keymemo.doc"
    destination = "c:\files\backup\">
```

cffile action = "delete"

Description

Deletes a file on the server.

Syntax

```
<cffile
    action = "delete"
    file = "full pathname">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdirectory`

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`file`	Required		Pathname of the file to delete. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.

Example

The following example deletes the specified file:

```
<cffile action = "delete"
    file = "c:\files\upload\#Variables.DeleteFileName#">
```

cffile action = "move"

Description

Moves a file from one location to another on the server.

Syntax

```
<cffile
    action = "move"
    destination = "full pathname"
    source = "full pathname"
    attributes = "file attributes list"
    charset = "character set option"
    mode = "mode">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdirectory`

History

See the History section of the main `cffile` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`destination`	Required		Pathname of the destination directory or file. If not an absolute path, it is relative to the source directory.
`source`	Required		Pathname of the file to move. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`attributes`	Optional		Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify `attributes = "readOnly"`, all other attributes are overwritten. • `readOnly` • `hidden` • `normal`

Attribute	Req/Opt	Default	Description
charset	Optional	JVM default file character set	The character encoding in which the file contents is encoded. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information character encodings, see www.w3.org/International/O-charset.html.
mode	Optional		Applies only to UNIX and Linux. Permissions. Octal values of UNIX chmod command. Assigned to owner, group, and other, respectively; for example: • 644: assigns read/write permission to owner; read permission to group and other. • 777: assigns read/write/execute permission to all.

Example

The following example moves the keymemo.doc file from the c:\files\upload\ directory to the c:\files\memo\ directory in Windows:

```
<cffile
    action = "move"
    source = "c:\files\upload\keymemo.doc"
    destination = "c:\files\memo\">
```

In this example, the destination directory is “memo.”

cffile action = "read"

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

Description

Reads a text file on the server. The file is read into a dynamic, local variable that you can use in the page. For example:

- Read a text file; insert the file's contents into a database
- Read a text file; use the find and replace function to modify the file's contents

Note: *This action reads the file into a variable in the local Variables scope. It is not intended for use with large files, such as logs, because this can bring down the server.*

Syntax

```
<cffile
    action = "read"
    file = "full pathname"
    variable = "variable name"
    charset = "character set option">
```

See also

`cfdirectory`

History

See the History section of the main `cffile` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`file`	Required		Pathname of the file to read. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.

Attribute	Req/Opt	Default	Description
variable	Required		Name of variable to contain contents of text file.
charset	Optional	Character encoding identified by the file's byte order mark, if any; otherwise, JVM default file character set.	The character encoding in which the file contents is encoded. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 If the file starts with a byte order mark and you set this attribute to a conflicting character encoding, ColdFusion generates an error. For more information character encodings, see www.w3.org/International/O-charset.html.

Usage

The following example creates a variable named Message for the contents of the file message.txt:

```
<cffile action = "read"
    file = "c:\web\message.txt"
    variable = "Message">
```

The variable `Message` can be used in the page. For example, you could display the contents of the message.txt file in the final web page as follows:

```
<cfoutput>#Message#</cfoutput>
```

ColdFusion supports functions for manipulating the contents of text files. You can also use the variable that is created by a `cffile action = "read"` operation in the `ArrayToList` and `ListToArray` functions.

Note: *If you use this tag to read a file that is encoded using the Windows Cp1252 (windows-1252) encoding of the Latin-1 character set on a system whose default character encoding is Cp1252, and the files has characters encoded in the Hex 8x or 9x range, you must specify* `charset="windows-1252"` *attribute, even though this is the default encoding. Otherwise, some characters in the Hex8x and 9x ranges that do not map correctly and display incorrectly.*

cffile action = "readBinary"

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

Description

Reads a binary file (such as an executable or image file) on the server, into a binary object parameter that you can use in the page. To send it through a web protocol (such as HTTP or SMTP) or store it in a database, first convert it to Base64 using the `ToBase64` function.

Note: *This action reads the file into a variable in the local Variables scope. It is not intended for use with large files, such as logs, because they can bring down the server.*

Syntax

```
<cffile
    action = "readBinary"
    file = "full pathname"
    variable = "variable name">
```

See also

`cfdirectory`

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`file`	Required		Pathname of a binary fine to read. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`variable`	Required		Name of variable to contain contents of binary file.

Usage

You convert the binary file to Base64 to transfer it to another site.

ColdFusion supports reading an image file as a binary and passing the result to a `cfimage`, for example:

```
<!--- Convert a JPG image to a binary object. --->
<cffile action="readBinary" file="maxwell05.jpg" variable="binaryObject">
<!--- Create a cfimage from the binary object variable. --->
<cfset myImage=ImageNew(binaryObject)>
```

Example

The following example reads the binary file `somewhere.jpg`, writes it to a different folder as `somewhereB.jpg`, and then displays the new file:

```
<cffile action = "readBinary" file =
"C:\inetpub\wwwroot\cfdocs\getting_started\photos\somewhere.jpg" variable = "aBinaryObj">

<!--- Output binary object to JPEG format for viewing. --->
<cffile action="write" file = "c:\files\updates\somewhereB.jpg"
    output = "#toBinary(aBinaryObj)#">

<!--- HTML to view image. --->
<img src="C:\files\updates\somewhereB.jpg">
```

cffile action = "rename"

Description

Renames or moves a file on the server.

Syntax

```
<cffile
    action = "rename"
    destination = "pathname"
    source = "full pathname"
    attributes = "file attributes list"
    mode = "mode">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdirectory`

History

See the History section of the main `cffile` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`destination`	Required		Destination file or directory. If not an absolute path, it is relative to the source directory.
`source`	Required		Pathname of file to rename. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`attributes`	Optional		Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if `attributes = "readOnly"`, all other attributes are overwritten. • `readOnly` • `hidden` • `normal`
`mode`	Optional		Applies only to UNIX and Linux. Permissions. Octal values of UNIX `chmod` command. Assigned to owner, group, and other. For example: • 644: assigns read/write permission to owner; read permission to group and other. • 777: assigns read/write/execute permission to all.

Usage

The `rename` action renames or move a file. The `destination` attribute must be a pathname, not just a new name for the file. If the destination is a directory, the file is moved and not renamed.

Example

Windows example:

```
<!--- Source Document is read-only but when renamed it becomes normal (not hidden or
    read-only). --->
<cffile action = "rename" source = "c:\files\memo\readonlymemo.doc"
    destination = "c:\files\memo\normalmemo.doc" attributes="normal">
```

UNIX example:

```
<cffile action = "rename" source = "#myWR#/memo/sample.txt"
    destination = "#myWR#/memo/other_sample.txt" mode="666">
```

cffile action = "upload"

Description

Copies a file to a directory on the server.

Syntax

```
<cffile
    action = "upload"
    destination = "full pathname"
    fileField = "form field"
    accept = "MIME type|file type"
    attributes = "file attribute or list"
    mode = "permission"
    nameConflict = "behavior"
    result = "result name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdirectory`

History

See the History section of the main `cffile` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`destination`	Required		Pathname of directory in which to upload the file. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`fileField`	Required		Name of form field used to select the file. Do not use number signs (#) to specify the field name.
`accept`	Optional		Limits the MIME types to accept. Comma-delimited list. For example, the following code permits JPEG and Microsoft Word file uploads: `accept = "image/jpg, application/msword"` The browser uses the file extension to determine file type.
`attributes`	Optional		Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify `attributes = "readOnly"`, all other attributes are overwritten. • `readOnly` • `hidden` • `normal` (if you use this option with other attributes, it is overridden by them)
`mode`	Optional		Applies only to UNIX and Linux. Permissions. Octal values of `chmod` command. Assigned to owner, group, and other, respectively, for example: • 644: assigns read/write permission to owner; read permission to group and other. • 777: assigns read/write/execute permission to all.

Attribute	Req/Opt	Default	Description
nameConflict	Optional	Error	Action to take if filename is the same as that of a file in the directory. • Error: file is not saved. ColdFusion stops processing the page and returns an error. • Skip: file is not saved. This option permits custom behavior based on file properties. • Overwrite: replaces file. • MakeUnique: forms a unique filename for the upload; name is stored in the file object variable serverFile.
result	Optional		Lets you specify a name for the variable in which cffile returns the result (or status) parameters. If you do not specify a value for this attribute, cffile uses the prefix cffile. For more information, see Usage.

Usage

After a file upload is completed, you can get status information using file upload parameters. To refer to parameters, use either the cffile prefix or, if you specified an alternate name in the result attribute, the name you specified there. For example, if you did not specify a name in the result attribute, access the fileExisted parameter as #cffile.fileExisted#. If you set the result attribute to myResult, however, access fileExisted as #myResult.fileExisted#.

Status parameters can be used anywhere that other ColdFusion parameters can be used.

When you use a cfform tag or an HTML form tag to submit the form with the file to be uploaded, you must specify enctype="multipart/form-data" in the tag, as shown in the example for this tag. By default, ColdFusion MX 7 sends the form with the encoding type of application/x-www-form-urlencoded, which causes an error in the cffile tag.

The result attribute allows functions or CFCs that get called from multiple pages at the same time to avoid overwriting the results of one call with another.

Note: *The file prefix is deprecated, in favor of the cffile prefix. Do not use the file prefix in new applications.*

If your page is uploading a file that was selected on a form or was otherwise sent to your page via a multipart/form-data HTTP message, you can determine the approximate size of the file by checking the value of the CGI.content_length variable. This variable includes the file length plus the length of any other request content.

The following file upload status parameters are available after an upload:

Parameter	Description
attemptedServerFile	Initial name ColdFusion used when attempting to save a file
clientDirectory	Directory location of the file uploaded from the client's system
clientFile	Name of the file uploaded from the client's system
clientFileExt	Extension of the uploaded file on the client system (without a period)
clientFileName	Name of the uploaded file on the client system (without an extension)
contentSubType	MIME content subtype of the saved file
contentType	MIME content type of the saved file
dateLastAccessed	Date and time the uploaded file was last accessed
fileExisted	Whether the file already existed with the same path (yes or no)
fileSize	Size of the uploaded file

Parameter	Description
fileWasAppended	Whether ColdFusion appended uploaded file to a file (yes or no)
fileWasOverwritten	Whether ColdFusion overwrote a file (yes or no)
fileWasRenamed	Whether uploaded file renamed to avoid a name conflict (yes or no)
fileWasSaved	Whether ColdFusion saves a file (yes or no)
oldFileSize	Size of a file that was overwritten in the file upload operation
serverDirectory	Directory of the file saved on the server
serverFile	Filename of the file saved on the server
serverFileExt	Extension of the uploaded file on the server (without a period)
serverFileName	Name of the uploaded file on the server (without an extension)
timeCreated	Time the uploaded file was created
timeLastModified	Date and time of the last modification to the uploaded file

Note: *File status parameters are read-only. They are set to the results of the most recent* `cffile` *operation. If two* `cffile` *tags execute, the results of the second overwrite the first, unless you have specified a different result variable in the* `result` *attribute.*

Example

The following example creates a unique filename, if there is a name conflict when the file is uploaded on Windows:

```
<!--- Windows Example --->
<!--- Check to see if the Form variable exists. --->
<cfif isDefined("Form.FileContents") >
    <!--- If TRUE, upload the file. --->
    <cffile action = "upload"
        fileField = "FileContents"
        destination = "c:\files\upload\"
        accept = "text/html"
        nameConflict = "MakeUnique">
<cfelse>
    <!--- If FALSE, show the Form. --->
    <form method="post" action=<cfoutput>#cgi.script_name#</cfoutput>
        name="uploadForm" enctype="multipart/form-data">
        <input name="FileContents" type="file">
        <br>
        <input name="submit" type="submit" value="Upload File">
    </form>
</cfif>
```

cffile action = "write"

Description

Writes a text file on the server, based on dynamic content. You can create static HTML files from the content, or log actions in a text file.

Syntax

```
<cffile
    action = "write"
    file = "full pathname"
    output = "content"
    addNewLine = "yes|no"
    attributes = "file attributes list"
    charset = "character set option"
    fixnewline = "yes|no"
    mode = "permission">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdirectory`

History

See the History section of the main `cffile` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Type of file manipulation that the tag performs.
`file`	Required		Pathname of the file to write. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`output`	Required		Content of the file to be created.
`addNewLine`	Optional	yes	• `yes`: appends newline character to text written to file. • `no`
`attributes`	Optional		Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify `attributes = "readOnly"`, all other attributes are overwritten. • `readOnly` • `hidden` • `normal`

Attribute	Req/Opt	Default	Description
charset	Optional	JVM default file character set	The character encoding in which the file contents is encoded. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information character encodings, see www.w3.org/International/O-charset.html.
fixnewline	Optional	no	• yes: changes embedded line-ending characters in string variables to operating-system specific line endings. • no: does not change embedded line-ending characters in string variables.
mode	Optional		Applies only to UNIX and Linux. Permissions. Octal values of UNIX chmod command. Assigned to owner, group, and other, respectively; for example: • 644: assigns read/write permission to owner; read permission to group and other. • 777: assigns read/write/execute permission to all.

Example

This example creates a file with information a user entered in an HTML insert form:

```
<cffile action = "write"
    file = "c:\files\updates\#Form.UpdateTitle#.txt"
    output = "Created By: #Form.FullName#
    Date: #Form.Date#
    #Form.Content#">
```

If the user submitted a form with the following:

```
UpdateTitle = "FieldWork"
FullName = "World B. Frueh"
Date = "10/30/01"
Content = "We had a wonderful time in Cambridgeport."
```

ColdFusion would create a file named FieldWork.txt in the c:\files\updates\ directory and the file would contain the following text:

```
Created By: World B. Frueh
Date: 10/30/01
    We had a wonderful time in Cambridgeport.
```

This example shows the use of the mode attribute for UNIX. It creates the file /tmp/foo with permissions rw-r--r-- (owner = read/write, group = read, other = read):

```
<cffile action = "write"
    file = "/tmp/foo"
```

```
    mode = 644>
```

This example appends to the file and sets permissions to read/write (rw) for all:

```
<cffile action = "append"
    destination = "/home/tomj/testing.txt"
    mode = 666
    output = "Is this a test?">
```

This example uploads a file and gives it the permissions owner/group/other = read/write/execute):

```
cffile action = "upload"
    fileField = "fieldname"
    destination = "/tmp/program.exe"
    mode = 777>
```

This example uses the `fixnewline` attribute to changes embedded line-ending characters in `xmlString`, which is derived from `xmlData`, to operating-system specific line endings.

```
<cfxml variable="xmlData">
    <docroot>
        <payload type="string">This is some plain text</payload>
    </docroot>
</cfxml>
<cfset xmlString = toString(xmlData)>

<cfset key = createUUID()>
<cfset encString=encrypt(xmlString, key)>
<cffile action="write" addnewline="yes"
    file="C:\CFusionMX7\wwwroot\test\store.dat"
    output="#encString#" fixnewline="yes">
<cffile action="read" file="C:\CFusionMX7\wwwroot\test\store.dat"
    variable="retrievedString">
<cfset decString=decrypt(retrievedString, key)>
<cfdump var="#decString#">
<cfset newXML = xmlParse(decString)>
<cfdump var="#newXML#">
```

ColdFusion 8 supports using `cffile` to write an image, for example:

```
<!--- Create a new cfimage. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Draw a square on the image. --->
<cfset ImageDrawRect(myImage,10,10,100,100)>
<!--- Use cffile to write the cfimage to a JPG. --->
<cffile action="write" output="#myImage#" file="c:\cfpix\square.jpg">
```

cfflush

Description

Flushes currently available data to the client.

Category

Data output tags, Page processing tags

Syntax

```
<cfflush
    interval = "integer number of bytes">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcache`, `cfheader`, `cfinclude`, `cfsetting`, `cfsilent`

Attributes

Attribute	Req/Opt	Default	Description
`interval`	Optional		Integer. Flushes output each time this number of bytes becomes available. HTML headers, and data that is already available when the tag is executed, are omitted from the count.

Usage

The first occurrence of this tag on a page sends back the HTML headers and any other available HTML. Subsequent `cfflush` tags on the page send only the output that was generated after the previous flush.

When you flush data, ensure that enough information is available, as some browsers might not respond if you flush only a small amount. Similarly, set the `interval` attribute for a few hundred bytes or more, but not thousands of bytes.

Use the `interval` attribute only when a large amount of output will be sent to the client, such as in a `cfloop` or a `cfoutput` of a large query. Using this form globally (such as in the Application.cfm file) might cause unexpected errors when CFML tags that modify HTML headers are executed.

Because the `cfflush` tag sends data to the browser when it executes, it has several limitations, including the following:

- Using any of the following tags or functions on a page anywhere after the `cfflush` tag can cause errors or unexpected results: `cfcontent`, `cfcookie`, `cfform`, `cfheader`, `cfhtmlhead`, `cflocation`, and `SetLocale`. Similarly, do not use any tags that use AJAX features, including `cfdiv`, `cflayout`, `cflayoutarea`, `cfpod`, `cfsprydataset`, `cftooltip`, `cfwindow`, or HTML format `cfgrid`, `cftree`, `cftextarea`, or `cfinput` (using `autosuggest` or `datefield` attributes) tags. All of the preceding tags and functions normally modify the HTML header, but cannot do so after a `cfflush` tag, because the `cfflush` sends the header.
- Using the `cfset` tag to set a cookie anywhere on a page that has a `cfflush` tag does not set the cookie in the browser.
- Using the `cfflush` tag in the body of several tags, including `cfsavecontent`, `cfquery`, and custom tags, causes errors.
- If you save Client variables as cookies, any client variables that you set after a `cfflush` tag are not saved in the browser.

Note: *Normally, the* `cferror` *tag discards the current output buffer and replaces it with the contents of the error page. The* `cfflush` *tag discards the current buffer. As a result, the* `Error.GeneratedContent` *variable resulting from a* `cferror` *tag after a* `cfflush` *contains any contents of the output buffer that has not been flushed. This content is not sent to the client. The content of the error page displays to the client after the bytes that have been sent.*

Example

The following example uses `cfloop` tags and the `rand` random number generating function to delay data display. It simulates a page that is slow to generate data.

```
<h1>Your Magic numbers</h1>
<p>It will take us a little while to calculate your ten magic numbers. It takes a lot of
work to find numbers that truly fit your personality. So relax for a minute or so while we
do the hard work for you.</p>
<H2>We are sure you will agree it was worth the short wait!</H2>
<cfflush>

<cfflush interval=10>
<!--- Delay Loop to make it seem harder. --->
<cfloop index="randomindex" from="1" to="200000" step="1">
    <cfset random=rand()>
</cfloop>

<!--- Now slowly output 10 random numbers. --->
<cfloop index="Myindex" from="1" to="10" step="1">
    <cfloop index="randomindex" from="1" to="100000" step="1">
        <cfset random=rand()>
    </cfloop>
    <cfoutput>
        Magic number #Myindex# is:  #RandRange
        100000, 999999)#<br><br>
    </cfoutput>
</cfloop>
```

cfform

Description

Builds a form with CFML custom control tags; these provide more functionality than standard HTML form input elements. You can include the resulting form on the client page as HTML or Adobe Flash content, and generate the form by using XML and XSLT.

Category

Forms tags

Syntax

```
<cfform
    accessible = "yes|no"
    action = "form action"
    archive = "URL"
    codeBase = "URL"
    format = "HTML|Flash|XML"
    height = "pixels|percent"
    id = "HTML id"
    method = "POST|GET"
    name = "name"
    onError = "JavaScript function name or ActionScript code"
    onLoad = "load event script"
    onReset = "reset event script"
    onSubmit = "JavaScript"
    onSuccess = "JavaScript function name"
    preloader = "yes|no"
    preserveData = "yes|no"
    scriptSrc = "path"
    skin = "Flash skin|XSL skin"
    style = "style specification"
    timeout = "seconds"
    width = "pixels|percent"
    wMode = "window|transparent|opaque">

    ...

</cfform>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfajaximport`, `cfapplet`, `cfcalendar`, `cfformgroup`, `cfformitem`, `cfgrid`, `cfinput`, `Usage`, `cfselect`, `cfslider`, `cftextarea`, `cftree`; "Requesting and Presenting Information" on page 510 in the *ColdFusion Developer's Guide*

History

ColdFusion 8:

- Added support for adding interactive fields in PDF forms.
- Added the `onSuccess` attribute and support in AJAX controls for the `onError` attribute

ColdFusion MX 7:

- Added ability to set the default value of the `scriptSrc` attribute in the ColdFusion Administrator.

- Deprecated the `passthrough` attribute. The tag now supports all HTML `form` tag attributes directly.
- Added the `method` attribute and support for the GET method.
- Added support for Flash and XML output, including the `format`, `height`, `width`, `preloader`, `timeout`, `wMode`, `accessible`, and `skin` attributes.
- Added `cfformgroup`, `cfformitem`, and `cftextarea` child tags.
- Added the `onReset` attribute.

ColdFusion MX:

- Deprecated the `enableCAB` attribute. It might not work, and might cause an error, in later releases.
- Changed the `name` and `action` attributes to optional.
- Changed integer validation to require an integer value. In previous releases it would convert a floating point value to an integer.

Attributes

The following table lists attributes that ColdFusion uses directly. For HTML format forms, this tag also supports the standard HTML `form` tag attributes that are not on this list, and passes them directly to the browser. ColdFusion also includes all supported HTML attributes in the XML.

Attribute	Applies to	Req/Opt	Default	Description
`accessible`	Flash	Opt	`no`	Specifies whether to include support screen readers in the Flash form. Screen reader support adds approximately 80 KB to the SWF file sent to the client.
`action`	Flash HTML XML	Opt	See Description	Name of ColdFusion page to execute when the form is submitted for processing. If you omit this attribute, the form posts to the page identified by the CGI.SCRIPT_NAME variable, the requested page that resulted in displaying the form.
`archive`	applets in HTML and XML	Opt	/CFIDE/classes/cfapplets.jar	URL of downloadable Java classes for `cfgrid`, `cfslider`, and `cftree` applet controls.
`codeBase`	applets in HTML and XML	Opt	/CFIDE/classes/cf-j2re-win.cab	URL of downloadable JRE plug-in for Internet Explorer; used for `cfgrid`, `cfslider`, and `cftree` Java applet controls.
`format`	Flash HTML XML	Opt	`HTML`	• `HTML`: generates an HTML form and send it to the client. `cfgrid` and `cftree` child controls can be in Flash or applet format. • `Flash`: generates a Flash form and send it to the client. All controls are in Flash format. • `XML`: generates XForms-compliant XML and save the results in a variable specified by the `name` attribute. By default, ColdFusion also applies an XSL skin and displays the result. For more information, see the `skin` attribute.
`height`	Flash XML	Opt	100%	The height of the form. Use a number to specify pixels. In Flash, you can use a percentage value, such as `"height=60%"` to specify a percentage of the available width. The displayed height might be less than the specified size. **Note:** The `width` and `height` attributes are required for Flash forms, if they are used inside of a table.

Attribute	Applies to	Req/Opt	Default	Description
id			name attribute value	the HTML id of the form.
method	Flash HTML XML	Opt	post	The method the browser uses to send the form data to the server: • post: sends the data using the HTTP post method. This method sends the data in a separate message to the server. • get: sends the data using the HTTP get method, which puts the form field contents in the URL query string.
name	Flash HTML XML	Opt	CFForm_*n*	A name for the form. In HTML format, if you omit this attribute and specify an id attribute, ColdFusion does not include a name attribute in the HTML sent to the browser; this behavior lets you use the cfform tag to create XHTML-compliant forms. If you omit the name attribute and the id attribute, ColdFusion generates a name of the form CFForm_*n* where *n* is a number that is assigned serially to the forms on a page.
onError	Flash HTML	Opt		For Flash format forms: Applies only for onSubmit or onBlur validation; has no effect for onServer validation. An ActionScript expression or expressions to execute if the user submits a form with one or more validation errors. For HTML format forms: Applies only to forms inside cfdiv, cflayout, cfpod, or cfwindow controls. The name of a JavaScript function that runs if an asynchronous form submission fails. For more information, see Using forms in cfdiv, cflayout, cffpod, and cfwindow controls in the Usage section.
onLoad	HTML XML	Opt		JavaScript to execute when the form loads.
onReset	HTML XML	Opt		JavaScript to execute when the user clicks a reset button.
onSubmit	Flash HTML XML	Opt		JavaScript or ActionScript function to execute to preprocess data before form is submitted. If any child tags specify onSubmit field validation, ColdFusion does the validation before executing this JavaScript.
onSuccess	HTML	Opt		Applies only to forms inside cfdiv, cflayout, cfpod, or cfwindow controls. The name of a JavaScript function that will run when an asynchronous form submission succeeds. For more information see Using forms in cfdiv, cflayout, cffpod, and cfwindow controls in the Usage section.
preloader	Flash	Opt	yes	Specifies whether to display a progress bar when loading the Flash form.

Attribute	Applies to	Req/Opt	Default	Description
preserveData	HTML XML	Opt	no	When the cfform action attribute posts back to the page that contains the form, this attribute determines whether to override the control values with the submitted values. • no: uses values specified in the control tag attributes. • yes: uses corresponding submitted values. Applies to these controls: • cfinput, cfslider, cftextinput: overrides the value attribute value. • cfselect controls that are populated from queries: overrides the selected attribute. See cfselect. • cftree controls: overrides the cftreeitem expand attribute. If yes, expands previously-selected elements. The cftree completePath attribute must be set to yes. • cfgrid controls: has no effect. (This avoids confusion as to whether data has been resubmitted to the database by the control.)
scriptSrc	Flash HTML XML	Opt	See Description	Specifies the URL, relative to the web root, of the directory that contains ColdFusion JavaScript files, including the cfform.js file with the client-side JavaScript used by this tag and its child tags. For XML format forms, this directory is also the default directory for XSLT skins. When you use this attribute, the specified directory must have the same structure as the /CFIDE/scripts directory. For example, if you specify scriptsrc="/resources/myScripts", the JavaScript files used by ColdFusion AJAX features must be in the /resources/myScripts/ajax directory. This attribute is useful if the file is not in the default location. This attribute may be required in some hosting environments and configurations that block access to the /CFIDE directory. The location is set in the ColdFusion Administrator; by default, it is /CFIDE/scripts. **Notes:** If you specify this attribute, you must copy the CF_RunActiveContent.js file from the CFIDE/scripts directory to the specified directory. You can have only one scriptsrc attribute on a page, including any cfajaximport tag scriptsrc attribute. If you have multiple cfform tags, you can specify the scriptsrc attribute in a cfajaximport tag and it applies to all HTML format forms.

Attribute	Applies to	Req/Opt	Default	Description
`skin`	Flash XML	Opt	Flash: `haloGreen` XML: default.xsl	**Flash:** Use a halo color to stylize the output. The skin determines the color used for highlighted and selected elements. • `haloSilver` • `haloBlue` • `haloGreen` • `haloOrange` **XML:** Specfies whether to apply an XSL skin and display the resulting HTML to the client. Can be any of the following: • ColdFusion skin name: applies the specified skin. • XSL file name: applies the skin located in the specified path. • `none`: does not apply an XSL skin. Your CFML page must process the XML that ColdFusion saves in the variable specified by the `name` attribute, and display any results. • omitted or `default`: uses the ColdFusion default skin. You can specify the following ColdFusion skins (located in the *cf_webroot*\CFIDE\scripts\xsl directory): • basic • basiccss • beige • blue • bluegray • lightgray • red • silver A filename can be any of the following: • absolute URL • URL relative to the web root • absolute file path • name of a file in the scripts folder or a subdirectory of the *cf_webroot*\CFIDE\scripts directory. In this case, do not specify the .xsl suffix.
`style`	HTML, Flash, XML	Opt		Styles to apply to the form. In HTML or XML format, ColdFusion passes the style attribute to the browser or XML. In Flash format, must be a style specification in CSS format. For detailed information on specifying Flash styles, see "Creating Forms in Flash" on page 577 in the *ColdFusion Developer's Guide*.
`timeout`	Flash	Opt	0	Integer number of seconds for which to keep the form data in the Flash cache on the server. A value of 0 prevents the data from being cached. For more information, see "Caching data in Flash forms" on page 594 in the *ColdFusion Developer's Guide*.

Attribute	Applies to	Req/Opt	Default	Description
width	Flash XML	Opt	100%	The width of the form. Use a number to specify pixels. In Flash, you can use a percentage value, such as "width=60%" to specify a percentage of the available width. ***Note:*** The width and height attributes are required for Flash forms, if they are used inside of a table.
wMode	Flash	Opt	window	Specifies how the Flash form appears relative to other displayable content that occupies the same space on an HTML page. • window: the Flash form is the topmost layer on the page and obscures anything that would share the space, such as drop-down dynamic HTML lists. • transparent: the Flash form honors the z-index of dhtml so you can float items above it. If the Flash form is above any item, transparent regions in the form show the content that is below it. • opaque: the Flash form honors the z-index of dhtml so you can float items above it. If the Flash form is above any item, it blocks any content that is below it.

Note: *Attributes that are not marked as supported in XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML as html namespace attributes to the* `form` *tag.*

Usage

This tag requires an end tag.

You can use the following ColdFusion form control tags in the `cfform` tag:

- `cfapplet`: Used in HTML and XML format only; embeds a registered Java applet.
- `cfformgroup`: Used in Flash and XML format only; groups and arranges child controls.
- `cfformitem`: Used in Flash and XML format only; adds horizontal rules, vertical rules, and text to the form.
- `cfgrid`: Creates a grid control to display tabular data.
- `cfinput`: Creates and an input element.
- `cfselect`: Creates a drop-down list box.
- `cfslider`: Used in HTML and XML format only; creates a slider control.
- `cftextarea`: Creates a multiline text input box.
- `cftree`: Creates a tree control.

In HTML format, all tags, and in Flash format the `cftree` and `cfgrid` tags, require JavaScript support on the browser. The `cfapplet` tag and applet format `cfgrid`, `cfslider`, and `cftree` tags require the client to download a Java applet.

If you specify Flash format in the `cfform` tag, ColdFusion ignores any HTML in the form body. You must use ColdFusion tags, such as `cfinput`, for all form controls. You can include individual Flash format `cfgrid` and `cftree` controls in an HTML format `cfform` tag.

In Flash format, if your forms do not request sensitive data (such as credit card numbers), consider setting the `timeout` attribute. This can prevent users from getting "The form data has expired. Please reload this page in your browser" errors if they use the browser back button to return to the form. For more information, see “Caching data in Flash forms” on page 594 in “Caching data in Flash forms” on page 594 in the *ColdFusion Developer’s Guide*.

Note: *In Flash format, if you do not specify* `height` *and* `width` *attributes, Flash reserves browser space equal to the area of the browser window. If any other output follows the form, users must scroll to see it. Therefore, if you follow a Flash form with additional output, specify the* `height` *and* `width` *values. The* `width` *and* `height` *attributes are required for Flash forms, if they are used inside of a table.*

If attribute value text must include quotation marks, escape them by doubling them.

Using the onError attribute in Flash forms

If you use onSubmit or onBlur validation, the `onError` attribute lets you specify ActionScript code to execute if the user tries to submit a Flash form with validation errors, as follows:

- If you specify one or more valid Flash expressions, Flash executes the expressions.
- If you omit the attribute, Flash displays a dialog box with all applicable error messages.
- If you specify `onError=""` (an empty string) Flash does not display any message, but does not submit the form.

Your ActionScript can use the errors variable to determine the fields and errors. The errors object has the following fields:

Field	Contents
name	The `name` attribute of the control's CFML tag.
field	The internal name used by Flash for the field name (for example, _level0.field1).
value	The value in the field.
message	The `message` attribute of the control's CFML tag.

The following example shows `cfform` tags with an `onError` attribute that selects the tab in an accordion or tab navigator that contains a `lastName` field with an invalid entry:

```
<cfform name="form1" format="flash" width="800" height="500"
        onError="if (errors['lastName'] != undefined
            ){tabA.selectedIndex=0; _root.lastName.setFocus();}">
```

Incorporating HTML form tags and attributes

In HTML format, the `cfform` tag lets you incorporate the following standard HTML elements. They are not available in Flash format:

- Standard HTML `form` tag attributes and values. The attributes and values are included in the `form` tag that `cfform` outputs in the page. For example, you can use `form` tag attributes like `target` or `onMouseOver` with `cfform`.
- HTML tags that can ordinarily be put within the HTML `form` tag. For example, you can use the HTML `input` tag to create a submit button in a `cfform`, without the other features of `cfinput`:

```
<cfform>
    <input type = "Submit" value = " update... ">
</cfform>
```

Using forms in cfdiv, cflayout, cffpod, and cfwindow controls

The `cfdiv`, `cflayout`, `cffpod`, and `cfwindow` tags create AJAX-based controls that can serve as containers for interactive forms. When you use such a structure, you do not want submitting form information to cause a new page to be displayed; instead, you want dynamic code to modify the existing page without causing a complete reload. You can do this byusing the `onSuccess` and `onError` attributes.

The function specified by the onSuccess attribute gets called if the form data is submitted successfully. This function is responsible for updating the pod, layout, or window to reflect the results of the submission, for example, to display additional data or pop up a confirmation window. This function must not take any arguments

The function specified by the onError attribute gets called if an error occurs when the form data is submitted. This function is responsible for handling the error, such as displaying an error message. This function must take two arguments: an error number and an error message.

Incorporating interactive fields in PDF forms

ColdFusion 8 lets you use the cfform tag to create PDF forms that contain static and interactive form fields. The cfform tag must exist within a cfdocument tag (where format="pdf"). Only one cfform tag can exist within a cfdocument tag.

Completed forms can be posted to the server as an HTTP Post, or the entire PDF can be submitted as binary stream. If the PDF is submitted, you can use the cffile tag to save completed PDF form to a hard drive:

```
<cffile action="write" file="c:\savedpdf.pdf" output="#PDF.content#">
```

The output can be manipulated and extracted by using the tag.

Only the following cfform attributes are supported in generating PDF forms:

- action
- format
- method
- name
- onSubmit
- skin
- style

To embed an existing PDF form generated by LiveCycle Designer or Acrobat, use the tag.

Example

```
<h3>cfform Example</h3>
<!--- If Form.oncethrough exists, the form has been submitted. --->
<cfif IsDefined("Form.oncethrough")>
    <cfif IsDefined("Form.testVal1")>
        <h3>Results of Radio Button Test</h3>
        <cfif Form.testVal1>Your radio button answer was yes
        <cfelse>Your radio button answer was no
        </cfif>
    </cfif>
    <h3>Results of Checkbox Test</h3>
    <cfif IsDefined("Form.chkTest2")>
        Your checkbox answer was yes
    <cfelse>
        Your checkbox answer was no
    </cfif>
    <cfif IsDefined("Form.textSample") AND Form.textSample is not "">
        <h3>Results of Credit Card Input</h3>
        Your credit card number, <cfoutput>#Form.textSample#</cfoutput>,
        was valid under the MOD 10 algorithm.
    </cfif>
    <cfif IsDefined("Form.sampleSlider")>
        <cfoutput>
```

```
            <h3>You gave this page a rating of #Form.sampleSlider#</h3>
        </cfoutput>
    </cfif>
    <hr noshade="True">
</cfif>

<!--- Begin by calling the cfform tag. --->
<cfform name="cfformexample">
    <h4>This example displays radio button input type for cfinput.</h4>
    Yes <cfinput type = "Radio" name = "TestVal1" value = "Yes" checked>
    No <cfinput type = "Radio" name = "TestVal1" value = "No">
    <h4>This example displays checkbox input type for cfinput.</h4>
    <cfinput type = "Checkbox" name = "ChkTest2" value = "Yes">
    <h4>This shows client-side validation for cfinput text boxes.</h4>
    (<i>This item is optional</i>)<br>
    Please enter a credit card number:
    <cfinput type = "Text" name = "TextSample"
        message = "Please enter a Credit Card Number"
        validate = "creditcard" required = "No">
    <h4>This example shows the use of the cfslider tag.</h4>
    Rate your approval of this example from 1 to 10 by sliding control.<br>
    1 <cfslider name = "sampleSlider" width="100"
            label = "Page Value: " range = "1,10"
            message = "Please enter a value from 1 to 10"> 10
    <p><cfinput type = "submit" name = "submit" value = "show me the result">
    <cfinput type = "hidden" name = "oncethrough" value = "Yes"></p>
</cfform>
```

A simple XML form

The following example shows a simple XML-format form. It uses the default.xsl transform that is supplied with ColdFusion to generate the HTML output for display:

```
<cfform name="testXForm" format="XML" skin="basic">
<!--- Use cfformgroup to put the first and last names on a single line. --->
    <cfformgroup type="horizontal">
        <cfinput type="text" name="firstname" label="First Name:" value="Robert">
        <cfinput type="text" name="lastname" label="Last Name:" value="Smith">
</cfformgroup>
<cfinput type="password" name="password" label="Password:" value="">
<cfinput type="hidden" name="hidden" label="hidden:" value="">
<cfselect name="state" style="width:200" label="State">
    <option>California</option>
    <option selected>Utah</option>
    <option>Iowa</option>
    <option selected>New York</option>
 </cfselect>
 <cftextarea name="description" label="Description:" rows="5" cols="40">
    this is sample text.</cftextarea>
</cfform>
```

A simple PDF form

```
<cfdocument format="pdf">
    <cfdocumentsection ../>
    ...
    ...
    <cfform type="html/xform">
        <cfinput type="textbox" name="employeeName" value="#fullName#" readonly="true">
        <cfinput type="textbox" name="employeeID" value="#id#" readonly>
        <cfselect name="contributionPercentage" options="#optionsStruct#" required="true">
        <cfinput type="submit" name="SubmitAsHTTPPost">
```

```
        <cfinput type="submit" name="SubmitAsPDF" submitType="PDF">
    </cfform>
    ...
    ...
    <cfdocumentsection ../>
</cfdocument>
```

cfformgroup

Description

Creates a container control for multiple form controls. Used in the `cfform` tag body of Adobe Flash and XML forms. Ignored in HTML forms.

Category

Forms tags

Syntax

```
<cfformgroup
    type = "group type"
    label = "label"
    style = "style specification"
    selectedIndex = "page number">
    width = "pixels"
    height = "pixels"
    enabled = "yes|no"
    visible = "yes|no"
    onChange = "ActionScript expression"
    tooltip = "text"
    id = "unique identifier">
    ...ColdFusion forms controls...
</cfformgroup>

OR

<cfformgroup
    type = "repeater"
    query = "query object"
    maxrows = "integer">
    startrow = "row number"
    ...ColdFusion forms controls
</cfformgroup>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfapplet`, `cfcalendar`, `cfform`, `cfformitem`, `cfgrid`, `cfinput`, `cfselect`, `cfslider`, `cftextarea`, `cftree`, "Using the cfformgroup tag to structure forms" on page 582 and "Using cfformgroup tags" on page 598 in the *ColdFusion Developer's Guide*.

History

ColdFusion MX 7: Added this tag.

Attributes

The following table lists the attributes and their behavior in Flash forms. For XML, if not otherwise noted, the attribute is passed to the XML but is not interpreted by the basic XSL style sheet provided with ColdFusion.

Note: *Attributes that are not marked as supported in XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.*

Attribute	Req/Opt; formats	Default	Description
type	Required; Flash and XML		**XML**: Can be any XForms group type defined in the XSLT. The XSL skins provided with ColdFusion support the following types: • horizontal: align child tags horizontally within a form and put this tag's label attribute to the left of the children. • vertical: align child tags vertically within a form and put this tag's label attribute to the left of the children. • fieldset: corresponds to the HTML fieldset tag, which groups its children, typically by drawing a box around them and replacing part of the top line with legend text. To specify the legend, use the label attribute. To specify the box dimensions, use the style attribute with height and width values. You must explicitly use cfformgroup type="vertical" inside this formgroup to align its child tags vertically. **Flash**: Must be one of the following: • repeater: dynamically creates an instance of the cfformgroup's child tag or tags for each row of a query object, without requiring ColdFusion to recompile the Flash SWF file when the number of rows changes. • horizontal: aligns child tags horizontally within a form and put this tag's label attribute to the left of the children. Use this tag to arrange individual controls horizontally. • vertical: aligns child tags vertically within a form and puts this tag's label attribute to the left of the children. Use this tag to arrange individual controls vertically. • hbox: aligns children horizontally. Use this type to arrange groups of form controls horizontally. Do not use this attribute to align individual controls horizontally, because the child controls do not align properly; use the horizontal type instead. • *vbox*: aligns children vertically. Use this type to arrange groups of controls vertically. Do not use this attribute to align individual controls vertically, because the child controls do not align properly; use the vertical type instead. • hdividedbox: aligns children horizontally. Each child is in a box with a border, and there are dividers between the boxes that users can move to change the relative sizes of the children. Use a tag with this attribute to arrange groups of form controls horizontally. You cannot use this attribute to align individual controls horizontally. • vdividedbox: aligns children vertically. Each child is in a box with a border, and there are dividers between the boxes that users can move to change the relative sizes of the children. Use this type to group form controls, for example as a unit in an hbox form group. Do not use this attribute to align individual tags vertically. • panel: a container consisting of a title bar containing the label attribute text, a border, and a content area with vertically arranged children. • tile: places the children in a rectangular grid. • accordion: places each child in a pleat of an expanding and contracting accordion. Define each pleat using a cfformgroup type="page" tag. • tabnavigator: places the children in a tabbed dialog box. Define each tab by using a cfformgroup type="page" tag. • page: places the children tags, aligned vertically, in a single tab of a parent tabnavigator or pleat of an accordion container.

Attribute	Req/Opt; formats	Default	Description
`query`	Required for `type=repeater`, ignored otherwise; Flash		The query to use with the repeater. Flash creates an instance of each of the `cfformgroup` tag's child tags for each row in the query. You can use the `bind` attribute in the child tags to use data from the query row for the instance.
`enabled`	Optional; Flash	`yes`	Boolean value that specifies whether the controls in the form group are enabled. Disabled controls appear in light gray.
`height`	Optional; Flash		Height of the group container, in pixels. If you omit this attribute, Flash automatically sizes the container height. Ignored for Flash repeater type.
`id`	Optional; Flash		Unique identifier for the form group. When using the tabnavigator or accordion type, you must specify the `id` attribute to reference the controls through custom ActionScript.
`label`	Optional; Flash and XML		Label to apply to the form group. In Flash, does the following: • For a page or panel form group, determines the label to put on the corresponding accordion pleat, the tabnavigator tab, or the panel title bar. For a Flash horizontal or vertical form group, specifies the label to put to the left of the group. • Ignored in Flash for repeater, hbox, hdividedbox, vbox, vdividedbox, tile, accordion, and tabnavigator types.
`maxrows`	Optional; Flash		Used only for the `repeater` type; ignored otherwise. Specifies the maximum number of query rows to use in the Flash form repeater. If the query has more rows than the sum of the `startrow` attribute and this value, the repeater does not use the remaining rows.
`onChange`	Optional; Flash		Tabnavigator and accordion types only: ActionScript expression or expressions to execute when a new tab or accordion page is selected. **Note:** The `onChange` event occurs when the form first appears.
`selectedIndex`	Optional; Flash only		Used only for accordion and tabnavigator types; ignored otherwise. Specifies the page control to display as open, where 0 (not 1) specifies the first page control defined in the group.
`startrow`	Optional; Flash	0	Used only for the `repeater` type; ignored otherwise. Specifies the row number of the first row of the query to use in the Flash form repeater. This attribute is zero-based: the first row is row 0, not row 1 (as in most ColdFusion tags).
`style`	Optional; Flash and XML		**Flash**: a Flash style specification in CSS format. For detailed information on specifying Flash styles, see "Creating Forms in Flash" on page 577 in the *ColdFusion Developer's Guide*. **XML**: an inline CSS style specification.
`tooltip`	Optional; Flash		Text to display when the mouse pointer hovers in the form group area. If a control in the form group also specifies a tooltip, Flash displays the control's tooltip when the mouse pointer hovers over the control.
`visible`	Optional; Flash	`yes`	Boolean value specifying whether the controls in the form group are visible. If the controls are invisible, the space that would be occupied by visible controls is blank.
`width`	Optional; Flash and XML		Width of the group container, in pixels. If you omit this attribute, Flash automatically sizes the container width. Ignored for Flash repeater type.

Usage

This tag requires an end tag. This tag is ignored if the `cfform` type is HTML; any tag body's contents are interpreted as if the surrounding `cfformgroup` does not exist.

In Flash format forms, this tag organizes the contents of the form. It groups and arranges child tags. The body of this tag can contain the following tags; all other tags and text are ignored:

- `cfformgroup`
- `cfformitem`
- `cfcalendar`
- `cfgrid`
- `cfinput`
- `cfselect`
- `cftextarea`
- `cftree`

For more information on using this tag in Flash forms, see "Creating Forms in Flash" on page 577 in the *ColdFusion Developer's Guide*.

In XML format, ColdFusion passes the tag and its attributes to the XML; it is the responsibility of the skin XSLT to handle the XML. The ColdFusion basic skin supports the `horizontal`, `vertical`, and `dualselectlist` styles only. For more information on using this tag in XML forms, see "Creating Forms in Flash" on page 577 in the *ColdFusion Developer's Guide*.

Example

For a simple example of an XML form that uses a single `cfformgroup` tag, see `cfform`.

The following example shows how to use the `cfformgroup` tag to arrange elements on a Flash form. It creates an hdividedbox container that has a vbox container on each side. The left box has heading text and two radio buttons. The right box has heading text and three check boxes.

```
<h3>Simple cfformgroup Example</h3>
<cfform name="myform" height="450" width="500" format="Flash" >
    <cfformgroup type="hdividedbox" >
        <cfformgroup type="VBox">
            <cfformitem type="text" height="20">
                Pets:
            </cfformitem>
            <cfinput type="Radio" name="pets" label="Dogs" value="Dogs" checked>
            <cfinput type="Radio" name="pets" label="Cats" value="Cats">
        </cfformgroup>

        <cfformgroup type="VBox">
            <cfformitem type="text" height="20">
                Fruits:
            </cfformitem>
            <cfinput type = "Checkbox" name="chk1" Label="Apples" value="Apples">
            <cfinput type="Checkbox" name="chk2" Label="Bananas" value="Bananas">
            <cfinput type="Checkbox" name="chk3" Label="Pears" value="Pears">
        </cfformgroup>
    </cfformgroup>
</cfform>
```

The following more complex example shows more fully how you can use `cfformgroup` tags to arrange controls in a Flash form. It also shows many of the text formatting features that you can use in a text `cfformgroup` body. When you submit the form, the page dumps the contents of the Forms scope, to show you the submitted data.

```
<h2>cfformgroup Example</h2>
<cfif IsDefined("form.oncethrough")>
    <h3>The form submitted the following information to ColdFusion:</h3>
    <cfdump var="#form#"><br><br><br>
</cfif>

<h3>A Flash form using cfformgroup tags</h3>
<cfform name="myform" height="450" width="500" format="Flash">

<!--- The following formgroup shows how you can present formatted text. --->
    <cfformitem type="html">
        <b><font color="#FF0000" size="+4" face="serif">
        This form has two tabs, asking for the following:</font></b><br>
        <li>contact information</li>
        <li><i>preferences</i></li>
        <b>Try entering information on both tabs</b><br>
        Submit the form and see what ColdFusion gets in the Forms scope.</b><br>
        <a href="http://www..com/" target="_blank">
        <font color="#0000FF"><u>
        This link displays the home page in a new browser window
        </u></font></a><br>
         <br>
    </cfformitem>

<!--- Use a tabnavigator with two tabs for user input. --->
    <cfformgroup type="tabnavigator" height="220">
        <cfformgroup type="page" label="Contact Information">
            <!--- Align the first and last name fields horizontally --->
            <cfformgroup type="horizontal" label="Your Name">
                <cfinput type="text" required="Yes" name="firstName" label="First"
                    value="" width="100"/>
                <cfinput type="text" required="Yes" name="lastName" label="Last"
                    value="" width="100"/>
            </cfformgroup>
            <cfformitem type="html"><textformat indent="95"><font size="-2">
                Flash fills the email field in automatically.
                You can replace any of the text.
                </font></textformat>
            </cfformitem>
            <!--- The bind attribute gets the field contents from the firstname and lastName
                fields as they get filled in. --->
            <cfinput type="text" name="email" label="email"
                bind="{firstName.text}.{lastName.text}@mm.com">

            <cfinput type="text" name="phone" validate="telephone" required="Yes"
                label="Phone Number">
        </cfformgroup>

        <cfformgroup type="page" label="Preferences">
            <cfformitem type="text" height="30">
                <b>Tell us your preferences</b>
            </cfformitem>
            <!--- Put the pet selectors to the left of the fruit selectors. --->
            <cfformgroup type="hbox">
            <!--- Group the pet selector box contents, aligned vertically. --->
                <cfformgroup type="vbox">
                    <cfformitem type="text" height="20">
```

```
                    Pets:
                </cfformitem>
                <cfformgroup type="vertical">
                    <cfinput type="Radio" name="pets" label="Dogs" value="Dogs"
                        checked>
                    <cfinput type="Radio" name="pets" label="Cats" value="Cats">
                </cfformgroup>
            </cfformgroup>
        <!--- Group the fruit selector box contents, aligned vertically. --->
            <cfformgroup type="vbox">
                <cfformitem type="text" height="20">
                    Fruits:
                </cfformitem>
                <cfformgroup type="tile" width="200" label="Tile box">
                    <--- Flash requires unique names for all controls --->
                    <cfinput type = "Checkbox" name="chk1" Label="Apples"
                        value="Apples">
                    <cfinput type="Checkbox" name="chk2" Label="Bananas"
                        value="Bananas">
                    <cfinput type="Checkbox" name="chk3" Label="Pears"
                        value="Pears">
                    <cfinput type="Checkbox" name="chk4" Label="Oranges"
                        value="Oranges">
                    <cfinput type="Checkbox" name="chk5" Label="Grapes"
                        value="Grapes">
                    <cfinput type="Checkbox" name="chk6" Label="Cumquats"
                        value="Cumquats">
                </cfformgroup>
            </cfformgroup>
        </cfformgroup>
    </cfformgroup>

    <cfformgroup type="horizontal">
        <cfinput type = "submit" name="submit" width="100" value = "Show Results">
        <cfinput type = "reset" name="reset" width="100" value = "Reset Fields">
        <cfinput type = "hidden" name="oncethrough" value = "Yes">
    </cfformgroup>
</cfform>
```

cfformitem

Description

Inserts a horizontal line, a vertical line, a spacer, or text in a Flash form. Used in the `cfform` or `cfformgroup` tag body for Flash and XML forms. Ignored in HTML forms.

Category

Forms tags

Syntax

```
<cfformitem
    type = "hrule|vrule|spacer"
    height = "pixels"
    style = "style specification"
    visible = "yes|no"
    width = "pixels"/>

OR

<cfformitem
    type = "html|text|script"
    bind = "bind expression"
    enabled = "yes|no"
    height = "pixels"
    style = "style specification"
    tooltip = "text"
    visible = "yes|no"
    width = "pixels">
    ...text
</cfformitem>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfapplet`, `cfform`, `cfformgroup`, `cfgrid`, `cfinput`, `cfselect`, `cfslider`, `cftextarea`, `cftree`, "Adding text, images, rules, and space with the cfformitem tag" on page 579 in "Adding text, images, rules, and space with the cfformitem tag" on page 579 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7.01: Added the "`script`" value for `type` attribute.

ColdFusion MX 7: Added tag

Attributes

The following table lists the attributes and their behavior in Flash forms. For XML format, if not otherwise noted, the attribute is passed to the XML but is not interpreted by the basic XSL style sheet provided with ColdFusion.

Note: *Attributes that are marked as Flash only are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML in all controls except text and html types.*

Usage

Attribute	Req/Opt; formats	Default	Description
type	Required; Flash and XML		**Flash:** • html: place the text in the body of this tag on the form. For Flash forms, you can use the following text formatting tags, most of which correspond to HTML tags, in the text: a, b, br, font, i, img, li, p, textformat, and u. For details on using these formatting tags, see the Flash documentation. The style attribute has no effect on the format of the text in type. • text: place the text in the body of this tag on the form verbatim, without interpreting any markup. You can control the overall appearance of the text by using the style attribute. • spacer: places an invisible spacer of the specified height and width on the form. Used to place space between form controls. This tag must not have any children. • hrule: places a horizontal rule on the form. This tag must not have any children. • vrule: places a vertical rule on the form. This tag must not have any children. • script: lets you create functions in Flash forms, which reduces the possibility of reaching the 64 KB limit. **XML**: • html: puts the CFML tag's body text in a CDATA section in an XML xf:output element. • text: XML-formats (escapes characters such as <) the CFML tag's body text and puts it in a CDATA section in an XML xf:output element. • hrule: puts an hr tag in the output. Use the style attribute to specify all rule characteristics, including height and width. This tag must not have any children. Any other string: generates an XML xf:group element with the type name as the appearance attribute. The CFML tag body is put in a CDATA section in a cf:attribute name="body" element. The XSL transforms provided with ColdFusion ignore these elements.
bind	Optional; Flash		A Flash bind expression that populates the field with information from other form fields. If you use this attribute, ColdFusion ignores any text that you specify in the body of the cftextitem tag. This attribute can be useful if the cfformitem tag is in a cfformgroup type="repeater" tag. For more information, see Flash form data binding in the cfinput tag description.
enabled	Optional; Flash	yes	Boolean value that specifies whether the control is enabled. Disabled text appear in light gray. Has no effect on spacers and rules.
height	Optional; Flash		Height of the item, in pixels. If you omit this attribute, Flash automatically sizes the width. In ColdFusion XSL skins, use the style attribute, instead.
style	Optional; Flash and XML		**Flash:** • Must be a style specification in CSS format. • Ignored if the type attribute is html. For detailed information on specifying Flash styles, see "Creating Forms in Flash" on page 577 in the *ColdFusion Developer's Guide*. Not used with the spacer type. **XML:** • ColdFusion passes the style attribute to the XML. ColdFusion skins include the style attribute in the generated HTML.
tooltip	Optional; Flash		Text to display when the mouse pointer hovers over the control. Has no effect on spacers.

Attribute	Req/Opt; formats	Default	Description
visible	Optional; Flash	yes	Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank. Has no effect on spacers.
width	Optional; Flash		Width of the item, in pixels. If you omit this attribute, Flash automatically sizes the width. In ColdFusion XSL skins, use the style attribute, instead.

This tag requires an end tag or a slash before the closing end character of the opening tag, as the following example shows:

```
<cfformitem type="hrule" />
```

For more information on using this tag in Flash forms, see "Creating Forms in Flash" on page 577 in the *ColdFusion Developer's Guide*.

Example

The following example shows a simple Flash form by using horizontal rules and text:

```
<h3>cfformitem Example</h3>
<cfform name="myform" height="450" width="500" format="Flash" >
    <cfformitem type="hrule" />
    <cfformitem type="text">
        This simple form has two hrule cfformitem tags around the cfformitem tag that
            contains this text.
    </cfformitem>
    <cfformitem type="hrule" />
</cfform>
```

For a more complex form, see cfformgroup.

cfftp

Description

Lets users implement File Transfer Protocol (FTP) operations.

Category

File management tags, Internet protocol tags

Syntax

The tag syntax depends on the `action` attribute value. See the following sections:

- "cfftp: Opening and closing FTP server connections" on page 240
- "cfftp: Opening and closing secure FTP server connections" on page 243
- "cfftp: Connection: file and directory operations" on page 247
- "cfftp action = "listDir"" on page 252

See also

`cfhttp`, `cfldap`, `cfmail`, `cfpop`; "Performing file operations with cfftp" on page 1044 in "Interacting with Remote Servers" on page 1038 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `fingerprint`, `key`, `paraphrase`, and `secure` attributes to support secure FTP. Added the `values ="quote"`, `"site"`, `"allo"`, and `"acct"` to the `action` attribute.

ColdFusion MX 7: Added the `result` attribute for file and directory operations.

ColdFusion MX: Deprecated the `agentname` attribute. It might not work, and might cause an error, in later releases.

Usage

Use this tag to move files between a ColdFusion server and an FTP server.

This tag does not move files between a ColdFusion server and a client browser. You do this as follows:

- To transfer files from a client to a ColdFusion server: `cffile action = "upload"`
- To transfer files from a ColdFusion server to a client: the `cfcontent` tag

Security settings

ColdFusion security settings can prevent the `cfftp` tag from executing. If you run ColdFusion applications on a server that is used by multiple customers, consider the security of the files that the customer can move. For more information, see the "Administering Security" section of *Configuring and Administering ColdFusion*.

cfftp: Opening and closing FTP server connections

Description

To establish a connection with an FTP server, use the `open` action with a `connection` attribute.

Syntax

```
<cfftp
    action = "open|close|quote|site|allo|acct"
    actionparam = "command or account information"
    buffersize = "number"
    connection = "name"
    passive = "yes|no"
    password = "password"
    port = "port"
    proxyServer = "proxy server"
    retryCount = "number"
    server = "server"
    stopOnError = "yes|no"
    timeout = "time-sout in seconds"
    username = "name">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfhttp`, `cfldap`, `cfmail`, `cfpop`

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		FTP operation to perform. • `open`: creates an FTP connection. • `close`: terminates an FTP connection. • `quote`: sends a command verbatim to the FTP server. • `site`: executes a site-specific command. • `allo`: allocates memory for operations, such as putting large files, on the server. • `acct`: sends account information on systems that require it.
`actionparam`	Optional		Used only when `action` is `quote`, `site`, or `acct`. Specifies the command when `action` is `quote` or `site`; specifies account information when `action` is `acct`.
`buffersize`	Optional		Buffer size in bytes.
`connection`	Optional, but always used with `open` or `close`		Name of the FTP connection. If you specify the `username`, `password`, and `server` attributes, and if no connection exists for them, ColdFusion creates one. Calls to `cfftp` with the same connection name reuse the connection.
`passive`	Optional	no	• `yes`: enables passive mode. • `no`
`password`	Required if `action = "open"`		Password to log in the user.
`port`	Optional	21	Remote port to which to connect.

Attribute	Req/Opt	Default	Description
proxyServer	Optional		String. Name of proxy server (or servers) to use, if proxy access is specified.
retryCount	Optional	1	Number of retries until failure is reported.
server	Required if action = "open"		FTP server to which to connect; for example, ftp.myserver.com.
stopOnError	Optional	yes	• yes: halts processing, displays an appropriate error. • no: if secure="no", populates these variables: • cfftp.succeeded: yes or no. • cfftp.errorCode: error number. See the IETF Network Working Group RFC 959: File Transfer Protocol (FTP) at www.ietf.org/rfc/rfc0959.txt. • cfftp.errorText: Message text. For conditional operations, use cfftp.errorCode. Do not use cfftp.errorText for this purpose.
timeout	Optional	30	Value in seconds for the time-out of all operations, including individual data request operations.
username	Required if action = "open"		User name to pass in the FTP operation.

Usage

When you establish a connection with cfftp action="open" and specify a name in the connection attribute, ColdFusion caches the connection so that you can reuse it to perform additional FTP operations. When you use a cached connection for subsequent FTP operations, you do not have to specify the username, password, or server connection attributes. The FTP operations that use the same connection name automatically use the information stored in the cached connection. Using a cached connection helps save connection time and improves file transfer performance.

You do not need to open a connection for single, simple, FTP operations, such as GetFile or PutFile.

With any action except close, you can set the internal buffer size by specifying buffersize. If you specify quote, site, allo, or acct as the action and set secure="yes" an error is generated. You specify the command to send to the FTP server in the actionparam attribute when you specify site or quote as the action. When site is the action, you use the actionparam attribute to specify the site-specific information.

To keep a connection open throughout a session or longer, put the connection name in the Session or Application scope; for example, specify connection="Session.FTPConnection". However, if you do this, you must specify the full variable name in all FTP operations, and you must use the close action when you are finished. Keeping a connection open prevents others from using the FTP server; so close a connection as soon as possible. If you do not assign the connection name to Session or Application variable, the connection remains open for the current page only, and you do not have to close it manually.

Changes to a cached connection, such as changing retryCount or timeout values, might require reestablishing the connection.

Example

```
<p>cfftp lets users implement File Transfer Protocol operations. By default, cfftp caches
    an open connection to an FTP server.</p>
<p>cfftp operations are usually of two types:</p>
<ul>
    <li>Establishing a connection
```

```
    <li>Performing file and directory operations
</ul>
<p>This example opens and verifies a connection, lists the files in a directory, and closes
    the connection.</p>
<p>Open a connection</p>
<cfftp action = "open"
    username = "anonymous"
    connection = "My_query"
    password = "youremail@email.com"
    server = "ftp.tucows.com"
    stopOnError = "Yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
<p>List the files in a directory:
<cfftp action = "LISTDIR"
    stopOnError = "Yes"
    name = "ListFiles"
    directory = "/"
    connection = "my_query">
<cfoutput query = "ListFiles">
    #name#<br>
</cfoutput>

<p>Close the connection:</p>
<cfftp action = "close"
    connection = "My_query"
    stopOnError = "Yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
```

cfftp: Opening and closing secure FTP server connections

Description

To establish a connection with a secure FTP server, use the `open` action with a `connection` attribute, specify that `secure = "yes"`, and specify the key, passphrase, and fingerprint as appropriate.

```
<cfftp
    action = "open|close"
    connection = "name"
    fingerprint = "ssh-dss.ssh-rsa"
    key = "private key"
    passive = "yes|no">
    passphrase = "passphrase"
    password = "password"
    port = "port"
    proxyServer = "proxy server"
    retryCount = "number"
    secure = "yes|no"
    server = "server"
    stopOnError = "yes|no"
    timeout = "time-out in seconds"
    username = "name">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfhttp`, `cfldap`, `cfmail`, `cfpop`

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		FTP operation to perform. • `open`: creates an FTP connection. • `close`: terminates an FTP connection.
`connection`	Optional, but always used with `open` or `close`		Name of the FTP connection. If you specify the `username`, `password`, and `server` attributes, and if no connection exists for them, ColdFusion creates one. Calls to `cfftp` with the same connection name reuse the connection.
`fingerprint`	Optional. Used only when `server`, `username`, and `password` are supplied		Fingerprint of the host key in the form `ssh-dss.ssh-rsa`, which is a 16-byte unique identifier for the `server` attribute that you specify, The `fingerprint` consists of eight pairs of hexadecimal values in the form `hh:hh:hh:hh::hh:hh:hh:hh`. ColdFusion checks the `fingerprint` of the remote server only if the `fingerprint` value is specified.
`key`	Required if `action="open"` (When `secure="yes"`, either `password` or `key` is required.)		Public-key–based authentication. Refers to the absolute path to the private key of the user. Possession of a private key provides authentication by sending a signature created with a private key. The server must ensure that the key is a valid authentication for the user and that the signature is valid. Both must be valid to accept the authentication.
`passive`	Optional	no	Valid only if `secure="no"`. • `yes`: enables passive mode. • `no`

Attribute	Req/Opt	Default	Description
passphrase	Optional. Used when key is specified		Because private keys are stored in an encrypted form on the client host, the user must supply a passphrase to enable generating the signature.
password	Required if action="open" (When secure="yes", either password or key is required.)		Password to log in the user.
port	Optional	21	Remote port to which to connect.
proxyServer	Optional		String. Name of proxy server (or servers) to use, if proxy access is specified.
retryCount	Optional	1	Number of retries until failure is reported.
secure	Optional	no	• yes: enables secure FTP • no
server	Required if action="open"		FTP server to which to connect; for example, ftp.myserver.com.
stopOnError	Optional	no	• yes: halts processing, displays an appropriate error. • no: if secure="yes", populates the following variables: • If ColdFusion fails to connect to the secure FTP server, it halts processing and displays the appropriate error message • cfftp.succeeded: yes or no • cfftp.errorCode: error number • cfftp.errorText: message text • For all file operations, returns the following error codes: SSH-CONNECT 25 SSH_MSG_USERAUTH_FAILURE 51 SSH_MSG_USERAUTH_SUCCESS 52 SSH_MSG_REQUEST_SUCCESS 81 SSH_MSG_REQUEST_FAILURE 82 For conditional operations, use cfftp.errorCode. Do not use cfftp.errorText for this purpose.
timeout	Optional	30	Value in seconds for the time-out of all operations, including individual data request operations.
username	Required if action="open"		User name to pass in the FTP operation.

Usage

The cfftp tag lets you open a connection to a Secure Shell (SSH) server by using either symmetric or asymmetric encryption. To use symmetric encryption, you specify secure="yes", the user name, password, connection, and fingerprint. To use asymmetric encryption, you must first generate private-public key pairs for each user authorized to have access to the server. Each authorized user's public key is stored on the server; each user's private key is encrypted and stored on that user's computer. To open a connection to the SSH server, you specify secure="yes", the user name, the password or the private key and the passphrase that the server uses to decrypt the private key, connection, and fingerprint. After you open the connection to the SSH server, you can use that connection for any action supported by the cfftp tag.

To keep a connection open throughout a session or longer, put the connection name in the Session or Application scope; for example, specify `connection="Session.FTPConnection"`. However, if you do this, you must specify the full variable name in all FTP operations, and you must use the `close` action when you are finished. Keeping a connection open prevents others from using the FTP server; so close a connection as soon as possible. If you do not assign the connection name to Session or Application variable, the connection remains open for the current page only, and you do not have to close it manually.

Changes to a cached connection, such as changing `retryCount` or `timeout` values, might require reestablishing the connection.

Example

```
<!--- This example uses symmetric encryption. --->

<!--- Open the secure connection. --->
<cfftp action = "open"
    username = "myusername"
    connection = "My_query"
    password = "mypassword"
    fingerprint = "12:34:56:78:AB:CD:EF:FE:DC:BA:87:65:43:21"
    server = "ftp.tucows.com"
    secure = "yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
<cfdump var ="#My_query# label="connection">

<!--- Transfer files to the remote server. --->
<cfset absolutePathToLocalFile="C:\one\two\myfile.htm">
    <cfif FileExists(absolutePathToLocalFile)>
        <cfftp action = "putFile"
            connection="My_query"
            localFile="#variables.absolutePathToLocalFile#"
            remoteFile="/home/myname/sftptest/myfile.htm">
    <cfelse>
        <!--- Put error handling code here. --->
    </cfif>
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>

<!--- Close the connection. --->
<cfftp action="close" connection="My_query">
```

Example

```
<!--- This example uses asymmetric encryption. --->

<!--- Open the secure connection. --->
<cfftp action = "open"
    username = "myusername"
    connection = "My_query"
    key="C:\mykeys\myprivatekey"
    passphrase = "zHx628Fg"
    fingerprint = "12:34:56:78:AB:CD:EF:FE:DC:BA:87:65:43:21"
    server = "ftp.tucows.com"
    secure = "yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
<cfdump var ="#My_query# label="connection">

<!--- List files on the remote server. --->
<cftry>
    <!--- List the files in a directory. --->
        <cfftp action = "listDir"
            connection="My_query"
```

```
            stopOnError="yes"
            name="ListFiles"
            directory="/">
    <cfcatch>
        <!--- Close the connection. --->
        <cfftp action="close" connection="My_query" stopOnError="no">
    </cfcatch>
</cftry>
```

cfftp: Connection: file and directory operations

Description

To perform file and directory operations with `cfftp`, use this form of the `cfftp` tag.

Syntax

```
<cfftp
    action = "action"
    ASCIIExtensionList = "extensions"
    directory = "directory name"
    existing = "file or directory name"
    failIfExists = "yes|no"
    item = "directory or file"
    localFile = "filename"
    name = "query name"
    new = "file or directory name"
    passive = "yes|no"
    password = "password"
    proxyServer = "proxy server"
    remoteFile = "filename"
    result = "result name"
    server = "server"
    transferMode = "ASCII FTP|Binary FTP|Auto FTP"
    username = "name">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfhttp`, `cfldap`, `cfmail`, `cfpop`

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required if connection is not cached		FTP operation to perform: • changedir • createDir • listDir • removeDir • getFile • putFile • rename • remove • getCurrentDir • getCurrentURL • existsDir • existsFile • exists
`ASCIIExtensionList`	Optional	`txt;htm;html;cfm;cfml;shtm;shtml;css;asp;asa`	Delimited list of file extensions that force ASCII transfer mode, if `transferMode = "auto"`.
`directory`	Required if `action = "changedir", "createDir", "listDir", or "existsDir"`		Directory on which to perform an operation.
`existing`	Required if `action = "rename"`		Current name of the file or directory on the remote server.
`failIfExists`	Optional	`yes`	• `yes`: If a local file with same name exists, the `getFile` action fails. • `no`
`item`	Required if `action = "exists"` or `"remove"`		Object of these actions: file or directory.
`localFile`	Required if `action = "getFile"` or `"putFile"`		Name of the file on the local file system.
`name`	Required if `action = "listDir"`		Query name of directory listing.
`new`	Required if `action = "rename"`		New name of file or directory on the remote server.
`passive`	Optional	no	• `yes`: enables passive mode. • `no`
`password`	Required if `action = "open"`		Password to log in the user.
`proxyServer`	Optional		String. Name of the proxy servers to use, if proxy access is specified.

Attribute	Req/Opt	Default	Description
remoteFile	Required if action = "getFile", "putFile", or "existsFile"		Name of the file on the FTP server file system.
result	Optional		Specifies a name for the structure in which cfftp stores the returnValue variable. If set, this value replaces cfftp as the prefix to use when accessing returnVariable. For more information, see Usage.
server	Required if FTP connection is not cached		FTP server to which to connect; for example, ftp.myserver.com.
transferMode	Optional	auto	• ASCII FTP transfer mode • Binary FTP transfer mode • Auto FTP transfer mode
username	Required if connection is not cached		User name to pass in the FTP operation.

Usage

If you use connection caching to an active FTP connection, you do not have to respecify the username, password, or server connection attributes.

Changing a cached connection, such as changing retryCount or timeout values, might require reestablishing the connection.

If action = "listDir", the attributes column returns directory or normal. Other platform-specific values, such as hidden and system, are no longer supported.

If action = "listDir", a mode column is returned. The column contains an octal string representation of UNIX permissions; for example, "777."

The cfftp.returnValue variable provides the return value for these actions:

- getCurrentDir
- getCurrentURL
- existsDir
- existsFile
- exists

For more information, see the *ColdFusion Developer's Guide*.

Action (cfftp.ReturnValue variable)

The results of an action determine the value of the returnValue variable, as the following table shows:

cfftp action	Value of cfftp.returnValue
getCurrentDir	String. Current directory.
getCurrentURL	String. Current URL.

cfftp action	Value of cfftp.returnValue
existsDir	yes or no.
existsFile	yes or no.
exists	yes or no.

To access the `returnValue` variable, you must prefix it with either `cfftp` or the value specified by the `result` attribute, if it is set. The `result` attribute provides a way for `cfftp` calls from multiple pages, possibly at the same time, to avoid overwriting the results of one with another. If you set the `result` attribute to `myResult`, for example, you would access the `returnVariable` variable as `myResult.returnVariable`. Otherwise, you would access it as `cfftp.returnVariable`.

Example

The following example opens a connection and gets a file that lists file or directory name, path, URL, length, and modification date:

```
<p>Open a connection
<cfftp connection = "myConnection"
    username = "myUserName"
    password = "myUserName@allaire.com"
    server = "ftp.allaire.com"
    action = "open"
    stopOnError = "Yes">

<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
<cfftp connection = "myConnection"
    action = "LISTDIR"
    stopOnError = "Yes"
    name = "ListDirs"
    directory = "/">

<p>FTP Directory Listing:<br>
<cftable query = "ListDirs" HTMLTable = "Yes" colHeaders = "Yes">
    <cfcol header = "<b>Name</b>" text = "#name#">
    <cfcol header = "<b>Path</b>" text = "#path#">
    <cfcol header = "<b>URL</b>" text = "#url#">
    <cfcol header = "<b>Length</b>" text = "#length#">
    <cfcol header = "<b>LastModified</b>"
     text = "#DateFormat(lastmodified)#">
    <cfcol header = "<b>IsDirectory</b>" text = "#isdirectory#">
</cftable>

<p>Move Image File to Remote Server:<br></p>
<!--- The image will be put into the root directory of the FTP server unless
    otherwise noted, i.e., remoteFile = "somewhere_put.jpg" vs remoteFile =
"/support/somewhere_put.jpg"
--->
<cfftp
    connection = "myConnection"
    action = "putFile"
    name = "uploadFile"
    transferMode = "binary"
    localFile = "C:\files\upload\somewhere.jpg"
    remoteFile = "somewhere_put.jpg">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>

<p>Close the connection:
```

```
<cfftp connection = "myConnection"
    action = "close"
    stopOnError = "Yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
```

cfftp action = "listDir"

Description

To access the columns in a query object, use this tag with `action = "listDir"`.

Usage

When you use this action, you must specify a value for the `name` attribute. This value holds the results of the `listDir` action in a query object. The query object consists of columns that you can reference, in the form `queryname.columname[row]`, where `queryname` is the name of the query, specified in the `name` attribute; and `columnname` is a column returned in the query object. The value *row* is the row number of each file/directory entry returned by the `listDir` operation. A separate row is created for each entry:

cfftp query object column	Description
Name	Filename of the current element.
Path	File path (without drive designation) of the current element.
URL	Complete URL for the current element (file or directory).
Length	File size of the current element.
LastModified	Unformatted date/time value of the current element.
Attributes	String. Attributes of the current element: normal or Directory.
IsDirectory	Boolean. Whether object is a file or directory.
Mode	Applies only to UNIX and Linux. Permissions. Octal string.

Note: *Previously supported query column values that pertain to system-specific information are not supported; for example,* `hidden` *and* `system`*.*

cffunction

Description

Defines a function that you can call in CFML. Required to define ColdFusion component methods.

History

ColdFusion 8:

- Added `returnformat`, `secureJSON`, and `verifyClient` attributes
- Added `returnformat` attribute
- Added `component` as a valid value for the `ReturnType` attribute.

ColdFusion MX 7: Added the `description` attribute, and added the `XML` value to the `returntype` attribute.

ColdFusion MX: Added this tag.

Category

Extensibility tags

Syntax

```
<cffunction
    name = "method name"
    access = "method access"
    description = "function description"
    displayName = "name"
    hint = "hint text"
    output = "yes|no"
    returnFormat = "not specified|JSON|plain|WDDX"
    returnType = "data type"
    roles = "securityRoles"
    secureJSON = "yes|no"
    verifyClient = "no|yes">
```

See also

cfargument, cfcomponent, cfinterface, cfinvoke, cfinvokeargument, cfobject, cfproperty, cfreturn, SerializeJSON

Attributes

Attribute	Req/Opt	Default	Description
name	Required		A string; a component method that is used in the `cfcomponent` tag.
access	Optional	public	The client security context from which the method can be invoked. The following values are valid: • `private`: available only to the component that declares the method and any components that extend the component in which it is defined. • `package`: available only to the component that declares the method, components that extend the component, or any other components in the package. • `public`: available to a locally executing page or component method. • `remote`: available to a locally or remotely executing page or component method, or a remote client through a URL, Flash, or a web service. To publish the function as a web service, this option is required.
description	Optional		Supplies a short text description of the function.

Attribute	Req/Opt	Default	Description
displayname	Optional		Meaningful only for CFC method parameters. A value to be displayed in parentheses following the function name when using introspection to show information about the CFC.
hint	Optional		Meaningful only for CFC method parameters. Text to be displayed when using introspection to show information about the CFC. The hint attribute value follows the syntax line in the function description.
output	Optional	Function body is processed as standard CFML	Specifies under which conditions the function can generate HTML output. The following values are valid: • yes: the entire function body is processed as if it were in a cfoutput tag. Variables names surrounded by number signs (#) are automatically replaced with their values. • no: the function is processed as if it were within a cfsilent tag. If you do not specify this attribute, the function body is processed as standard CFML. Any variables must be in cfoutput tags.
returnformat		Return as WDDX or XML; see description.	The format in which to return values to a remote caller. This attribute has no effect on values returned to a local caller. The following values are valid: • json: serialize the return value into JSON format before returning it remotely. • wddx: serialize the return value into WDDX format before returning it remotely. • plain: ensure that the return value is a type that ColdFusion can convert directly to a string, and return the string value without serialization. Valid types include all simple types, such as numbers, and XML objects. If the return value is a complex type, such as an array, or a binary value, ColdFusion generates an error. If you specify a returntype attribute, its value must be any, boolean, date, guid, numeric, string, uuid, variablename, or XML; otherwise, ColdFusion generates an error. By default, ColdFusion serializes all return types (including simple return types), except XML, into WDDX format, and returns XML data as XML text. You can also use returnformat as an HTTP request parameter when calling a remote CFC function. This parameter has the same effect as the returnformat attribute and overrides any returnformat attribute value specified in the cffunction tag.

Attribute	Req/Opt	Default	Description
returnType	Required for a web service; Optional, otherwise.	any	String; a type name; data type of the function return value: • any • array • binary • boolean • component: the return value must be a ColdFusion component. • date • guid: the argument must be a UUID or GUID of the form *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* where each *x* is a character that represents a hexadecimal number (0-9A-F). • numeric • query • string • struct • uuid: the argument must be a ColdFusion UUID of the form *xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx* where each *x* is a character that represents a hexadecimal number (0-9A-F). • variableName: a string formatted according to ColdFusion variable naming conventions. • void: does not return a value. • xml: allows web service functions to return CFML XML objects and XML strings. • A component name: If the type attribute value is not one of the preceding items, ColdFusion treats it as the name of a ColdFusion component. When the function executes, it generates an error if the argument that is passed in is not a CFC with the specified name. **Note:** If a function does not return a value and the returnType value is numeric, ColdFusion generates an error; ColdFusion does not generate an error for other types.
roles	Optional	"" (empty)	A comma-delimited list of ColdFusion security roles that can invoke the method. Only users who are logged in with the specified roles can execute the function. If this attribute is omitted, all users can invoke the method.
secureJSON	Optional	See Description	A Boolean value that specifies whether to add a security prefix in front of any value that the function returns in JSON-format in response to a remote call. The default value is the value of any This.secureJSON variable in the Application.cfc file or the secureJSON attribute of the cfapplication tag, or if there is nosecureJSON application setting, the Prefix Serialized JSON setting in the Administrator Server Settings > Settings page, which defaults to false. For more information see "Improving security" on page 674 in the *ColdFusion Developer's Guide*.
verifyClient	Optional	no	A Boolean value that specifies whether to require remote function calls to include an encrypted security token. For use with ColdFusion AJAX applications only. For more information see "Improving security" on page 674 in the *ColdFusion Developer's Guide*.

Usage

The cffunction tag can define a function that you call in the same manner as a ColdFusion built-in function.

To define a ColdFusion component (CFC) method, you must use a cffunction tag.

The following example shows `cffunction` tag attributes for a simple CFC method that returns a ColdFusion Query object.

```
<cffunction
    name="getEmployees"
    access="remote"
    returnType="query"
    hint="This query returns all records in the employee database. It candrill-down or narrow
the search, based on optional input parameters.">
```

For detailed information on using the `cffunction` tag for ColdFusion components, see "Building and Using ColdFusion Components" on page 158 in the *ColdFusion Developer's Guide*.

If you specify `returnformat="json"` and the function returns a query, ColdFusion serializes the query into a JSON Object with two entries, and array of column names, and an array of column data arrays. For more information see `SerializeJSON`.

If you specify a `roles` attribute, the function executes only if a user is logged in and belongs to one of the specified roles.

If you specify `variableName` for the `returnType` attribute, the function must return a string that is in ColdFusion variable name format; that is, the function must return a string that starts with a letter, underscore, or Unicode currency symbol, and consist of letters, numbers, and underscores (_), periods, and Unicode currency symbols, only. ColdFusion does not check whether the value corresponds to an existing ColdFusion variable.

Example

```
<cfcomponent>
    <cffunction name="getEmp">
         <cfquery
                name="empQuery" datasource="ExampleApps" >
                SELECT FIRSTNAME, LASTNAME, EMAIL
                FROM tblEmployees
         </cfquery>
         <cfreturn empQuery>
    </cffunction>
    <cffunction name="getDept">
        <cfquery name="deptQuery" datasource="ExampleApps" >
            SELECT *
            FROM tblDepartments
         </cfquery>
         <cfreturn deptQuery>
    </cffunction>
</cfcomponent>
```

cfgraph

Description

This tag is deprecated. Use the `cfchart`, `cfchartdata`, and `cfchartseries` tags instead.

Displays data graphically.

History

ColdFusion MX: Deprecated this tag. It works differently than it did in ColdFusion 5, and it might not work in later releases.

The incompatibilities between the ColdFusion MX implementation and earlier implementations of this tag are as follows:

cfgraph tag attribute	ColdFusion MX functionality
Title	Ignored.
Titlefont	Ignored.
Barspacing	Ignored.
Bordercolor	Color used for border, gridlines, and text displays.
Colorlist	List of colorColdFusions to use for each data point for bar, pyramid, area, horizontalbar, cone, cylinder, step, and pie charts.
Valuelabelfont	Sets value label text font. If the `Valuelabelfont`, `Itemlabelfont`, and `Legendfont` values differ, ColdFusion uses the last value that you specify in the tag. Arial is not supported; it is mapped to `Dialog`.
Itemlabelfont	Sets item label text font. If the `Valuelabelfont`, `Itemlabelfont`, and `Legendfont` values differ, ColdFusion uses the last value that you specify in the tag. Arial is not supported; it is mapped to `Dialog`.
Legendfont	Sets legend text font. If the `Valuelabelfont`, `Itemlabelfont`, and `Legendfont` values differ, ColdFusion uses the last value that you specify in the tag Arial is not supported; it is mapped to `Dialog`.
ShowLegend	• above, below, left, right: these options cause the legend to display, but have no effect on its location. • none: prevents display of a legend.
Valuelabelsize	Sets value label text size. If the `Valuelabelsize` and `Itemlabelsize` values differ, ColdFusion uses the last value that you specify in the tag.
Itemlabelsize	Sets item label text size.
Itemlabelorientation	Ignored. ColdFusion calculates best orientation based on label and graph size.
Borderwidth	• a nonzero number: default-width border, regardless of number value. • 0: no border.
Depth	• 0: displays graph with two-dimensional appearance. • any other value: displays graph with threedimensional appearance.
Linewidth	Ignored.

cfgraph tag attribute	ColdFusion MX functionality
Showvaluelabel	• `yes`: displays values on mouse-click. • `no`: suppresses value displays. • `rollover`: displays values on mouse-over.
Valuelocation	Ignored.
url	URL of page to open if any item in the graph is clicked. The following variables may be used within the URL; they are substituted with real values before the URL is accessed: • `"$value$"`: selected row/column value or an empty string. • `"$itemlabel$"`: selected item (column) value or an empty string. • `"$serieslabel$"`: selected series (row) value or an empty string. • `"javascript:..."`: executes client side scripts.
Urlcolumn	Ignored.
Type="HorizontalBar"	The (0,0) coordinate is located at the lower-left.
ScaleFrom	If the smallest value in the data is less than `scaleFrom` or the largest value in the data is greater than `scaleTo`, the respective data value is used as the minimum or maximum on the Y scale. Therefore, regardless of the `scaleFrom` or `scaleTo` value, all data values display.

cfgraphdata

Description

This tag is deprecated. Use the cfchart, cfchartdata, and cfchartseries tags instead.

Displays a data point in a graph. Used within the cfgraph tag.

History

ColdFusion MX: Deprecated this tag. It works differently than in ColdFusion 5 and might not work in later releases.

cfgrid

Description

Used in the `cfform` tag. Puts a grid control (a table of data) in a ColdFusion form. To specify grid columns and row data, use the `cfgridcolumn` and `cfgridrow` tags, or use the `query` attribute, with or without `cfgridcolumn` tags.

Category

Forms tags

Syntax

```
<cfgrid
    name="name"
    align="value"
    appendKey="yes|no"
    autoWidth="yes|no"
    bgColor="web color"
    bind="bind expression"
    bindOnLoad="yes|no"
    bold="yes|no"
    colHeaderAlign="left|right|center"
    colHeaderBold="yes|no"
    colHeaderFont="font_name"
    colHeaderFontSize="size"
    colHeaderItalic="yes|no"
    colHeaders="yes|no"
    colHeaderTextColor="web color"
    delete="yes|no"
    deleteButton="text"
    enabled="yes|no"
    font="column_font"
    fontSize="size"
    format="applet|Flash|html|xml"
    gridDataAlign="left|right|center"
    gridLines="yes|no"
    height="integer"
    highlightHref="yes|no"
    href="URL"
    hrefKey="column_name"
    hSpace="integer"
    insert="yes|no"
    insertButton="text"
    italic="yes|no"
    maxRows="number"
    notSupported="text"
    onBlur="ActionScript"
    onChange="ActionScript or bind expression"
    onError="JavaScript function name"
    onFocus="ActionScript function"
    onValidate="JavaScript function name"
    pageSize="number of rows"
    pictureBar="yes|no"
    preservePageOnSort="yes|no"
    query="query name"
    rowHeaderAlign="left|right|center"
    rowHeaderBold="yes|no"
    rowHeaderFont="font name"
    rowHeaderFontSize="size"
    rowHeaderItalic="yes|no"
```

```
    rowHeaders="yes|no"
    rowHeaderTextColor="web color"
    rowHeight="pixels"
    selectColor="web color"
    selectMode="mode"
    selectOnLoad="yes|no"
    sort="yes|no"
    sortAscendingButton="text"
    sortDescendingButton="text"
    stripeRowColor="web color"
    stripeRows="yes|no"
    style= "style specification"
    target="URL_target"
    textColor="web color"
    tooltip="text"
    visible="yes|no"
    vSpace="integer"
    width="integer">

        zero or more cfgridcolumn and cfgridrow tags

</cfgrid>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfajaximport`, `cfapplet`, `cfcalendar`, `cfgridcolumn`, `cfgridrow`, `cfgridupdate`, `cfform`, `cfformgroup`, `cfformitem`, `cfinput`, `cfselect`, `cfslider`, `cftextarea`, `cftree`, "Using HTML format grids" on page 631 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added support for HTML format grids, including the `html` value of the `format` attribute and the following attributes: `bind`, `bindOnLoad`, `pageSize`, `preservePageOnSort`, `stripeRows`, `stripeRowColor`.

ColdFusion MX 7.01: Added support for the `onBlur` and `onFocus` events.

ColdFusion MX 7:

- Added the `format` attribute and support for Flash and XML output.
- Added `enabled`, `onChange`, `style`, `tooltip`, and `visible` attributes (Flash format only).

ColdFusion MX: Changed the `rowHeaderWidth` attribute: ColdFusion does not use the `rowHeaderWidth` attribute. You can omit it.

Attributes

Note: *In XML format, ColdFusion passes all attributes to the XML. The supplied XSLT skins do not handle or display XML format grids, but do display applet and Flash format grids.*

Attribute	Req/Opt; formats	Default	Description
name	Required; all		Name of the grid control.
align	Optional; applet		Alignment of the grid cell contents: • Top • Left • Bottom • Baseline • Texttop • Absbottom • Middle • Absmiddle • Right
appendKey	Optional; HTML, applet	yes	• yes: when used with href, appends "CFGRIDKEY=" and information about the selected items. For details, see "Using the href attribute" on page 268. • no
autoWidth	Optional; HTML, applet	no	• yes: sets column widths so that all columns display within the grid width. Widths are equal or the proportions are determined by the relative cfgridcolumn width attribute values. Horizontal scroll bars are not available. • no: sets columns to equal widths or the values specified in the cfgridcolumn width attributes.
bgColor	Optional; all		Background color of the control. For most formats, can be a hexidecimal format or a named color. For a hexadecimal value, use the form "##xxxxxx", where x = 0-9 or A-F; use two number signs or none. For a list of the supported named colors, see cfchart. • Limitations: for HTML format, must be a valid web color; for Flash format, must be a hexadecimal value. • Flash format only: to specify background colors for alternating rows, separate the two colors with a comma.
bind	Optional; HTML		A bind expression used to fill the contents of the grid. Cannot be used with the query attribute. For more information, see "Binding data to form fields" on page 650 in "Using Ajax Data and Development Features" on page 648 in the *ColdFusion Developer's Guide*.
bindOnLoad	Optional; HTML	yes	• yes: executes the bind attribute expression when first loading the form. • no: does not execute the bind attribute expression until the first bound event . Ignored if there is no bind attribute. For more information, see "Using the bindOnLoad attribute" on page 634 in the *ColdFusion Developer's Guide*.
bold	Optional; all	no	• yes: displays text in bold. • no

Attribute	Req/Opt; formats	Default	Description
colHeaderAlign	Optional; applet	left	• left: left-aligns the column header text. • right: right-aligns the column header text. • center: centers the column header text.
colHeaderBold	Optional; all	no	• yes: displays column headers in bold. • no
colHeaderFont	Optional; all		Font of column header.
colHeaderFontSize	Optional; all		Size of column header text, in points.
colHeaderItalic	Optional; all	no	• yes: displays column headers in italic. • no
colHeaders	Optional; Applet, Flash	yes	• yes: displays column headers. • no
colHeaderTextColor	Optional; all		Color of column headers. • Options: same as for textColor attribute.
delete	Optional; HTML, applet	no	• yes: users can delete row data from the grid; takes effect only if selectmode="edit". • no
deleteButton	Optional; HTML, applet	Delete	Text for the Delete button; takes effect only if selectmode="edit".
enabled	Optional; Flash	yes	Flash format only: Boolean value that specifies whether the control is enabled. A disabled control appears in light gray.
font	Optional; all		Font of text.
fontSize	Optional; all		Size of text, in points.
format	Optional; all	applet	• applet: generates a Java applet. • Flash: generates a Flash grid control. • html: generates an AJAX-based HTML grid control that supports data binding. • xml: generates an XML representation of the grid. In XML format forms, includes the generated XML in the form. In HTML format forms, puts the XML in a string variable with the name specified by the name attribute.
gridDataAlign	Optional; applet	left	• left: left-aligns data within the column. • right: right-aligns data within the column. • center: centers data within the column.
gridLines	Optional; applet, Flash	yes	• yes: enables row and column rules. • no
height	Optional; all	300 (applet only)	Height of the control, in pixels. If you omit the attribute in Flash format, the grid sizes automatically.
highlightHref	Optional; applet	yes	• yes: highlights links associated with an href attribute value. • no

Attribute	Req/Opt; formats	Default	Description
href	Optional; HTML,. applet		URL or name of a query column that contains URLs to hyperlink each grid cell with.
hrefKey	Optional; HTML,. applet		A query column to use for the value appended to the href URL of each cell, if appendKey="True". If you use cfgridcolumn tags, the column must be specified in one of these tags.
hSpace	Optional; applet		Horizontal space to the left and right of the control, in pixels.
insert	Optional; applet	no	• yes: users can insert row data in the grid; takes effect only if selectmode="edit". • no
insertButton	Optional; applet	Insert	Text for the Insert button; takes effect only if selectmode="edit".
italic	Optional; all	no	• yes: displays text in italic. • no
maxRows	Optional; all		Maximum number of rows to display in the grid.
notSupported	Optional; applet	See Description	Text to display if the browser does not support Java or has Java support disabled. Default: " Browser must support Java to view ColdFusion Java Applets"
onBlur	Optional, Flash		ActionScript that runs when the grid loses focus.
onChange	Optional; HTML, Flash		**Flash format:** ActionScript to run when the control changes due to user action in the control. **HTML format:** Required for HTML format grids that specify a bind attribute and a selectMode value of edit. A bind expression that calls a CFC method, JavaScript function, or URL to update the data source.
onError	Optional; HTML, applet		In HTML format grids, name of a JavaScript function to execute if an error occurs. In applet format grids, name of a JavaScript function to execute if validation fails.
onFocus	Optional, Flash		ActionScript that runs when the calendar gets focus.
onValidate	Optional; applet		A JavaScript function to validate user input. The form object, input object, and input object value are passed to the function, which must return true if validation succeeds; false otherwise.
pageSize	Optional; HTML	10	The number of rows to display per page for a dynamic grid. If the number of available rows exceeds the page size, the grid displays only the specified number of entries on a single page, and the user navigates between pages to show all data. The grid retrieves data for each page only when it is required for display. This attribute is ignored if you specify a query attribute.
pictureBar	Optional; applet	no	• yes: puts images (and no text) on the Insert, Delete, and Sort buttons. • no: puts text (and no images) on the Insert, Delete, and Sort buttons.
preservePageOnSort	Optional; HTML	no	Specifies whether to display the page with the current page number, or display page 1, after sorting (or resorting) the grid. If this attribute is yes, selections are preserved when the grid sorts.

Attribute	Req/Opt; formats	Default	Description
query	Optional; all		Name of the query associated with the control. Cannot be used with the bind attribute.
rowHeaderAlign	Optional; applet	left	• left: left-aligns the row header text. • right: right-aligns the row header text. • center: centers the row header text.
rowHeaderBold	Optional; applet	no	• yes: displays row label text in bold. • no
rowHeaderFont	Optional; applet		Font for the row labels.
rowHeaderFontSize	Optional; applet		Text size of the row labels, in points.
rowHeaderItalic	Optional; applet	no	• yes: displays row label text in italic. • no
rowHeaders	Optional; applet	yes	• yes: displays a column of numeric row labels. • no
rowHeaderTextColor	Optional; applet	black	Text color of grid control row headers. • Options: same as for the textColor attribute.
rowHeight	Optional; Applet, Flash, XML		Minimum row height, in pixels. Used with cfgridcolumn type = "Image"; defines space for graphics to display in row.
selectColor	Optional; all		Background color for a selected item. • Options: same as for textColor attribute
selectMode	Optional; all	Applet format: Browse; HTML, Flash format: Row	Selection mode for items in the control. • Edit: the user can edit grid data. Selecting a cell lets the user edit the cell. • Row: user selections automatically extend to the row that contains selected cell. The following are used in applet format only; HTML and Flash formats interpret these as Row: • Single: user selections are limited to the selected cell. • Column: user selections automatically extend to the column that contains selected cell. • Browse: the user can only browse grid data.
selectOnLoad	Optional; HTML	yes	• yes: selects the first row of the grid when the gird loads. • no: does not select any rows when the grid loads.
sort	Optional; applet	no	Adds sort buttons to perform simple text sorts on a user-selected column: • yes: put sort buttons on the grid control. • no Independent of this setting, users can sort columns by clicking the column head. If selectMode="browse", the table cannot be sorted.
sortAscendingButton	Optional; applet	A > Z	Text for the Sort button.

Attribute	Req/Opt; formats	Default	Description
sortDescendingButton	Optional; applet	Z > A	Text for the Sort button.
stripeRowColor	Optional; HTML		The color to use for one of the alternating stripes. The bgColor setting determines the other color.
stripeRows	Optional; HTML	no	Boolean value that indicates whether to make the rows stripes in alternating colors.
style	Optional; Flash		Must be a style specification in CSS format. Ignored for type="text".
target	Optional; HTML, applet		The target frame or window in which to display the href URL; for example, "_blank".
textColor	Optional Flash, applet		Color of text. Can be a hexadecimal value or a named color. For a hexadecimal value, use the form "##xxxxxx", where x = 0-9 or A-F; use two number signs or none. For a list of the supported named colors, see cfchart.
tooltip	Optional; Flash		Flash format only: text to display when the mouse pointer hovers over the control.
visible	Optional; Flash	yes	Flash format only: Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank.
vSpace	Optional; applet		Vertical space above and below the control, in pixels.
width	Optional; all	300 (applet only)	Width of the control. In Flash and applet format, must be a number of pixels. In HTML format, can be in any valid CSS measurement unit, and a numeric-only value specifies pixels. If you omit the attribute in Flash or HTML format; the grid sizes automatically.

Usage

Most of the following paragraphs describe grid features that apply to all, or at least two, grid formats. For information that is specific to Flash forms, see “Creating Forms in Flash” on page 577 in the *ColdFusion Developer’s Guide*. For information that is specific to HTML format grids, see “Using HTML format grids” on page 631 in the *ColdFusion Developer’s Guide*.

This tag must be in a cfform tag block.

An applet format grid requires the client to download a Java applet. Also, if the client does not have an up-to-date Java plug-in installed, the system might also have to download an updated Java plug-in to display the an applet format grid. A Flash format grid generates a Flash control, and can be embedded in an HTML format cfform tag. For this tag to work properly in either Flash or applet format, the browser must also be JavaScript-enabled.

Note: *If you specify Flash format for this tag in an HTML format form, and you do not specify* height *and* width *attributes, Flash takes up more than the remaining visible area on the screen. If any other output follows the grid, including any form controls, users must scroll to see it. Therefore, if you follow a Flash grid in an HTML form with additional output, specify* height *and* width *values.*

You can populate a cfgrid with data from a cfquery. If you do not specify any cfgridcolumn tags in the cfgrid body, ColdFusion generates a grid with the following:

- A column for each column in the query.

• A default header for each column, created by replacing hyphen or underscore characters in the table column name with spaces. The first character, and any character after a space, are changed to uppercase; all other characters are lowercase.

This tag requires an end tag.

Note: *Clicking the submit button while editing a grid cell occasionally causes the cell changes to be lost. To ensure that changes are submitted properly, recommends that after user updates data in a cell, they click another cell before submitting the form.*

Returning cfgrid data to the action page

The following information applies to all `cfgrid` formats. Also, HTML format grids can dynamically get data by using a bind expression. For more information, see "Using HTML format grids" on page 631 in the *ColdFusion Developer's Guide*.

When a user submits a form, the `cfgrid` tag sends information about user actions by setting form variables in the data submitted to the form's action page. Because the data can vary, depending on the tag's `SelectMode` attribute value, the form variables that are returned also vary depending on this value.

In general, the data returned falls into one of these categories:

- Simple data, returned from simple select operations
- Complex data, returned from insert, update and delete operations

Simple selection data (SelectMode = Single, Column, or Row)

The data that form variables return to the `cfform`'s action page contains information about which cells the user selected. In general, ColdFusion makes this data available in the action page, as ColdFusion variables in the Form scope, with the naming convention `form.#GridName#.#ColumnName#.`

Each `SelectMode` returns these form variables:

```
SelectMode="single"
form.#GridName#.#ColumnName# = "SelectedCellValue"
```

```
SelectMode="column"
form.#GridName#.#ColumnName# = "ValueOfCellRow1,
ValueOfCellRow2, ValueOfCellRowN"
```

```
SelectMode="row"
form.#GridName#.#Column1Name# = "ValueOfCellInSelectedRow"
form.#GridName#.#Column2Name# = "ValueOfCellInSelectedRow"
form.#GridName#.#ColumnNName# = "ValueOfCellInSelectedRow"
```

Complex update data (SelectMode = Edit)

The grid returns a large amount of data, to inform the action page of inserts, updates, or deletes that the user made to the grid. In most cases, you can use the `cfgridupdate` tag to automatically gather the data from the form variables; the tag collects data, writes SQL calls, and updates the data source.

If you cannot use `cfgridupdate` (if, for example, you must distribute the returned data to more than one data source), you must write code to read form variables. In this mode, ColdFusion creates the following array variables in the Form scope for each `cfgrid`:

```
form.#GridName#.#ColumnName#
form.#GridName#.original.#ColumnName#
form.#GridName#.RowStatus.Action
```

Each table row that contains an update, insert, or deletion has a parallel entry in each of these arrays. To view all the information for all the changes, you can traverse the arrays, as in this example. To make it work with a `cfgrid` on a submitted `cfform`, set the GridName variable to the name of the grid and the ColNameList to a list of the grid columns.

```
<cfloop index="ColName" list="#ColNameList#">
    <cfif IsDefined("form.#GridName#.#ColName#")>
        <cfoutput><br>form.#GridName#.#ColName#:<br></cfoutput>

        <cfset Array_New = form.[#GridName#][#ColName#]>
        <cfset Array_Orig = form[#GridName#]['original'][#ColName#]>
        <cfset Array_Action = form[#GridName#]RowStatus.Action>

        <cfif NOT IsArray(Array_New)>
            <b>The form variable is not an array!</b><br>
        <cfelse>
            <cfset size = ArrayLen(Array_New)>
            <cfoutput>
            Result Array Size is #size#.<br>
            Contents:<br>
            </cfoutput>

            <cfif size IS 0>
                <b>The array is empty.</b><br>
            <cfelse>
                <table BORDER="yes">
                    <tr>
                        <th>Loop Index</TH>
                        <th>Action</TH>
                        <th>Old Value</TH>
                        <th>New Value</TH>
                    </tr>
                    <cfloop index="LoopCount" from="1" to=#size#>
                    <cfset Val_Orig = Array_Orig[#LoopCount#]>
                    <cfset Val_New = Array_New[#LoopCount#]>
                    <cfset Val_Action = Array_Action[#LoopCount#]>
                    <cfoutput>
                    <tr>
                        <td>#LoopCount#</td>
                        <td>#Val_Action#</td>
                        <td>#Val_Orig#</td>
                        <td>#Val_New#</td>
                    </tr>
                    </cfoutput>
                    </cfloop>
                </table>
            </cfif>
        </cfif>

    <cfelse>
        <cfoutput>form.#GridName#.#ColName#: NotSet!</cfoutput><br>
    </cfif>
</cfloop>
```

Using the href attribute

When specifying a URL with grid items using the `href` attribute, the `selectMode` attribute value determines whether the appended key value is limited to one grid item or extends to a grid column or row. When a user clicks a linked grid item, a `cfgridkey` variable is appended to the URL, in this form:

```
http://myserver.com?cfgridkey=selection
```

If the appendKey attribute is set to no, no grid values are appended to the URL.

The value of *selection* is determined by the value of the selectMode and attribute:

- If you specify a hrefKey attribute, *selection* is the field value of the column specified by the attribute. Otherwise, it is one of the following:
- If selectMode="Single", *selection* is the value of the column clicked.
- If selectMode="Row", *selection* is a comma-delimited list of column values in the clicked row, beginning with the value of the first cell in the row.
- If selectMode="Column", *selection* is a comma-delimited list of row values in the clicked column, beginning with the value of the first cell in the column.

When you use an href attribute, you can also specify a target attribute with any of the standard HTML target specifiers, _blank, _parent, _self, and _top, or with a specific frame name.

Example

The following example creates a Flash form that displays a set of available courses from the CourseList table in the cfdocexamples database. For more complex examples that use the cfgrid tag, see cfgridcolumn, cfgridrow, and cfgridupdate.

```
<!--- Query the database to fill up the grid. --->
<cfquery name = "GetCourses" dataSource = "cfdocexamples">
    SELECT Course_ID, Dept_ID, CorNumber,
    CorName, CorLevel
    FROM CourseList
    ORDER by Dept_ID ASC, CorNumber ASC
</cfquery>

<h3>cfgrid Example</h3>
<i>Currently available courses</i>
<!--- cfgrid must be inside a cfform tag. --->
<cfform>
    <cfgrid name = "FirstGrid" format="Flash"
        height="320" width="580"
        font="Tahoma" fontsize="12"
        query = "GetCourses">
    </cfgrid>
</cfform>
```

cfgridcolumn

Description

Used with the cfgrid tag in a cfform. Formats a column and optionally populates the column from a query. The font and alignment attributes used in cfgridcolumn override global font or alignment settings defined in cfgrid.

Category

Forms tags

Syntax

```
<cfgridcolumn
    name = "column name"
    bgColor = "web color|expression"
    bold = "yes|no"
    dataAlign = "left|right|center"
    display = "yes|no"
    font = "column font"
    fontSize = "size"
    header = "header"
    headerAlign = "left|right|center"
    headerBold = "yes|no"
    headerFont = "font name"
    headerFontSize = "size"
    headerItalic = "yes|no"
    headerTextColor = "web color"
    href = "URL"
    hrefKey = "column name"
    italic = "yes|no"
    mask= "format mask"
    numberFormat = "format"
    select = "yes|no"
    target = "URL target"
    textColor = "web color|expression"
    type = "type"
    values = "comma-separated strings and/or numeric range"
    valuesDelimiter = "delimiter character"
    valuesDisplay = "comma-separated strings and/or numeric range"
    width = "column width">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfgrid, cfgridrow, cfgridupdate, cfform, cfapplet, cfinput, cfselect, cfslider, cftextarea, cftree

History

ColdFusion MX 7: Added the mask attribute, and the currency type attribute value.

ColdFusion MX: Changed behavior if select="no": a user cannot select and edit the cell data, regardless of the cfgrid selectmode attribute value. When clicked, the cell border (and, depending on the selectColor value, the cell background) changes color, but the cell data cannot be edited.

Attributes

Note: *In XML format, ColdFusion passes all attributes to the XML. The supplied XSLT skins do not handle or display XML format grids, but do display applet and Flash format grids.*

Attribute	Req/Opt; formats	Default	Description
`name`	Required; all		Name of the grid column element. If the grid uses a query, this attribute must be the name of the query column that populates the grid column.
`bgColor`	Optional; all		Color of background of grid column. • Options: same as for the `textColor` attribute.
`bold`	Optional; all	As specified by `cfgrid`	• `yes`: displays grid control text in bold. • `no`
`dataAlign`	Optional; applet, Flash	As specified by `cfgrid`	Column data alignment: • `left` • `right` • `center`
`display`	Optional; all	`yes`	• `yes` • `no`: hides the column.
`font`	Optional; all	As specified by `cfgrid`	Font of data in column.
`fontSize`	Optional; all	As specified by `cfgrid`	Size of text in column.
`header`	Optional; all	`yes`	Text for the column header. Used only if the `cfgrid colHeaders` attribute is `yes`. The default value is `yes`.
`headerAlign`	Optional; applet	As specified by `cfgrid`	Column header text alignment: • `left` • `right` • `center`
`headerBold`	Optional; HTML, applet	As specified by `cfgrid`	• `yes`: displays header in bold. • `no`
`headerFont`	Optional; HTML, applet	As specified by `cfgrid`	Font for the column header.
`headerFontSize`	Optional; HTML, applet	As specified by `cfgrid`	Size of text for the column header, in pixels.
`headerItalic`	Optional; HTML, applet	As specified by `cfgrid`	• `yes`: displays column header in italic. • `no`
`headerTextColor`	Optional; HTML, applet		Color of grid control column header text. • Options: same as for the `textColor` attribute.
`href`	Optional; HTML, applet		URL or query column name that contains a URL to hyperlink each grid column with.
`hrefKey`	Optional; HTML, applet		The query column to use for the value appended to the `href` URL of each column, instead of the column's value.

Attribute	Req/Opt; formats	Default	Description
`italic`	Optional; all	As specified by `cfgrid`	• `yes`: displays grid control text in italic. • `no`
`mask`	Optional; Flash		A mask pattern that controls the character pattern that the form displays or allows users to input and sends to ColdFusion. For columns with the currency `type` attribute, the `mask` specifies the currency symbol. ColdFusion automatically inserts the character before the numeric value. For columns with text or numeric values, `mask` specifies the format to display or allow users to input, as follows: • A = [A-Za-z] • X = [A-Za-z0-9] • 9 = [0-9] • ? = Any character • All other characters = ColdFusion inserts the literal character. If the column values are dates or timestamps, ColdFusion uses the mask pattern to format the selected date. For details of the date/time mask format, see date/time formats in mask attribute.
`numberFormat`	Optional; Applet		Format for displaying numeric data in the grid. See the preceding table of attributes.
`select`	Optional; all	`yes`	Determines selection behavior if the `cfgrid selectmode` attribute value is `column`, `edit`, or `single`; ignored for row or browse values. • `yes`: users can select the column or select or edit cells in the column, as specified by the `selectmode` attribute. • `no`: users cannot select the column or select or edit cells in the column.
`target`	Optional; HTML, Applet		Frame or standard HTML target in which to open link specified in `href`.

Attribute	Req/Opt; formats	Default	Description
`textColor`	Optional; Applet, Flash		Color of grid element text in column as a hexadecimal number or text name. To enter a hexadecimal value, use the form `"##`*xxxxxx*`"`, where x = 0-9 or A-F; use two number signs or none. Limitations: In HTML format, must specify a valid HTML color. In Applet format, must be one of the following: • Any color, in hexadecimal format • `Black` • `Red` • `Blue` • `Magenta` • `Cyan` • `Orange` • `Darkgray` • `Pink` • `Gray` • `White` • `Lightgray` • `Yellow`

Attribute	Req/Opt; formats	Default	Description
type	Optional; all		You can specify the following values in all formats: • boolean: column displays as check box; if cell is editable, user can change the check mark. • numeric: user can sort grid data numerically. • string_noCase: user can sort grid data as case-insensitive text. You can specify the following value in applet and Flash formats; it does not work in HTML format: • image: grid displays the image specified by the URL in the column. If you use a relative URL, the image must be in the CFIDE\classes directory or a subdirectory. If the image is larger than the column cell, it is clipped to fit. Flash images must be JPEG files. Applet images can be JPEG or GIF files. You can specify the following value in applet format; it does not work in Flash or HTML format. • image: you can use the following built-in ColdFusion image names, in addition to paths to image files, in the column values: - cd - computer - document - element - folder - floppy - fixed - remote You can specify the following value in Flash format; it does not work in applet or HTML format: • currency: formats the column data as currency, aligning it around the decimal point. If users sort the grid by using this column, it sorts correctly for the currency, Use the mask attribute to specify a currency symbol; the default value is the dollar sign ($).
values	Optional; HTML, applet		Formats cells in column as drop-down list boxes; specify items in drop-down list, for example: values = "arthur, scott, charles, 1-20, mabel"
valuesDelimiter	Optional; HTML, applet	, (comma)	Delimiter in values and valuesDisplay attributes.
valuesDisplay	Optional; HTML, applet		Maps elements in the values attribute to string to display in the drop-down list. Delimited strings and/or numeric ranges.
width	Optional; all	Column head width	Column width, in pixels.

In applet format only, you can use the following `numberFormat` attribute mask characters to format output in U.S. numeric and currency styles. For more information on using these mask characters, see "NumberFormat" on page 1094. (The `cfgridcolumn` tag does not support international number formatting.)

Character	Meaning
_	(Underscore) Digit placeholder.
9	Digit placeholder.
.	(Period) Location of mandatory decimal point.
0	Located to left or right of mandatory decimal point; pads with zeros.
( )	Puts parentheses around mask if number is less than 0.
+	Puts plus sign before positive numbers, minus sign before negative numbers.
-	Puts space before positive numbers, minus sign before negative numbers.
,	(Comma) Separates every third decimal-place with a comma.
L,C	Left-justify or center-justify number within width of mask column. First character of mask must be L or C. Default: right-justified.
$	Puts dollar sign before formatted number. Must be the first character of mask.
^	(Caret) Separates left from right formatting.

date/time formats in mask attribute

By default, Flash displays date/time values in grid columns with a format that shows values such as Oct 29 2004 11:03:21. Use the `mask` attribute to display the date or time in a different format, as described in the following table:

Pattern letter	Description
Y	Year. If the number of pattern letters is two, the year is truncated to two digits; otherwise, it appears as four digits. The year can be zero-padded, as the third example shows in the following set of examples: Examples: YY = 03 YYYY = 2003 YYYYY = 02003
M	Month in year. The format depends on the following criteria: • If the number of pattern letters is one, the format is interpreted as numeric in one or two digits. • If the number of pattern letters is two, the format is interpreted as numeric in two digits. • If the number of pattern letters is three, the format is interpreted as short text. • If the number of pattern letters is four, the format is interpreted as full text. Examples: M = 7 MM= 07 MMM=Jul MMMM= July
D	Day in month. Examples: D=4 DD=04 DD=10
E	Day in week. The format depends on the following criteria: • If the number of pattern letters is one, the format is interpreted as numeric in one or two digits. • If the number of pattern letters is two, the format is interpreted as numeric in two digits. • If the number of pattern letters is three, the format is interpreted as short text. • If the number of pattern letters is four, the format is interpreted as full text. Examples: E = 1 EE = 01 EEE = Mon EEEE = Monday
A	AM/PM indicator.
J	Hour in day (0-23).
H	Hour in day (1-24).
K	Hour in am/pm (0-11).
L	Hour in am/pm (1-12).

Pattern letter	Description
N	Minute in hour. Examples: N = 3 NN = 03
S	Second in minute.
Other text	You can add other text into the pattern string to further format the string. You can use punctuation, numbers, and all lowercase letters. You should avoid uppercase letters because they may be interpreted as pattern letters. Example: EEEE, MMM. D, YYYY at H:NN A = Tuesday, Sept. 8, 2003 at 1:26 PM

Example

The following example lets you update certain fields of the CourseList table in the cfdocexamples database. It uses `cfgridcolumn` tags to structure the table.

```
<!--- If the gridEntered field exists, the form has been submitted.
        Update the database. --->
<cfif IsDefined("form.gridEntered")>
    <cfgridupdate grid = "FirstGrid" dataSource = "cfdocexamples"
        tableName = "CourseList" keyOnly = "Yes">
</cfif>

<!--- Query the database to fill up the grid. --->
<cfquery name = "GetCourses" dataSource = "cfdocexamples">
    SELECT Course_ID, Dept_ID, CorNumber,CorName, CorLevel, CorDesc
    FROM CourseList
    ORDER by Dept_ID ASC, CorNumber ASC
</cfquery>

<html>
<head>
<title>cfgrid Example</title>
</head>
<body>
<h3>cfgrid Example</h3>
<I>You can update the Name, Level, and Description information for courses.</i>
<!--- The cfform tag must surround a cfgrid control. --->
<cfform action = "#CGI.SCRIPT_NAME#">
    <cfgrid name = "FirstGrid" width = "500"
            query = "GetCourses" colheaderbold="Yes"
            font = "Tahoma" rowHeaders = "No"
            selectColor = "Red" selectMode = "Edit" >
        <!--- cfgridcolumn tags arrange the table and control the display. --->
        <!--- Hide the primary key, required for update --->
        <cfgridcolumn name = "Course_ID" display = "No">
        <!--- select="No" does not seem to have any effect!!! --->
        <cfgridcolumn name = "Dept_ID" header = "Department" Select="No" width="75"
            textcolor="blue" bold="Yes">
        <cfgridcolumn name = "CorNumber" header = "Course ##" Select="No" width="65">
        <cfgridcolumn name = "CorName" header = "Name" width="125">
        <cfgridcolumn name = "CorLevel" header = "Level" width="85">
        <cfgridcolumn name = "CorDesc" header = "Description" width="125">
    </cfgrid>
    <br>
    <cfinput type="submit" name="gridEntered">
```

```
</cfform>
</body>
</html>
```

cfgridrow

Description

Lets you define a cfgrid control that does not use a query as source for row data. If a query attribute is specified in the cfgrid tag, the cfgridrow tags are ignored.

Category

Forms tags

Syntax

```
<cfgridrow
    data = "col1, col2, ...">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfgrid, cfgridcolumn, cfgridupdate, cfform, cfinput, cfselect, cfslider, cftextarea, cftree

Attributes

Attribute	Req/Opt	Default	Description
data	Required		Comma-delimited list of column values. If a value contains a comma, it must be escaped with another comma.

Example

The following example shows how you use the cfgridrow tag can populate a cfgrid tag from list data:

```
<!--- Set two lists, each with the data for a grid column. --->
<cfset cities = "Rome,Athens,Canberra,Brasilia,Paris">
<cfset countries = "Italy,Greece,Australia,Brazil,France">

<cfform name = "cities">
    <cfgrid name="GeoGrid" autowidth = "yes" vspace = "4" height = "120"
            font="tahoma" rowheaders="no">
        <cfgridcolumn name="City" header="City">
        <cfgridcolumn name="Country" header="Country">
        <!--- Loop through the lists using cfgridrow to poplulate the grid. --->
        <cfloop index="i" from="1" to="#ListLen(cities)#">
            <cfgridrow data ="#ListGetAt(cities, i)#,#ListGetAt(countries, i)#">
        </cfloop>
    </cfgrid><br><br>
</cfform>
```

cfgridupdate

Description

Used with a cfgrid tag. Updates data sources directly from edited grid data. This tag provides a direct interface with your data source.

This tag applies delete row actions first, then insert row actions, then update row actions. If it encounters an error, it stops processing rows.

Category

Forms tags

Syntax

```
<cfgridupdate
    grid = "grid name"
    dataSource = "data source name"
    tableName = "table name"
    keyOnly = "yes|no">
    password = "data source password"
    tableOwner = "table owner"
    tableQualifier = "qualifier"
    username = "data source user name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfgrid, cfgridcolumn, cfgridrow, cfform, cfapplet, cfinput, cfselect, cfslider, cftextinput, cftree

History

ColdFusion MX: Deprecated the connectString, dbName, dbServer, dbtype, provider, and providerDSN attributes. They do not work, and might cause an error, in releases later than ColdFusion 5.

Attributes

Attribute	Req/Opt	Default	Description
grid	Required		Name of the cfgrid form element that is the source for the update action.
dataSource	Required		Name of the data source for the update action.
tableName	Required		Name of the table to update. For ORACLE drivers, entry must be upper-case. For Sybase driver, entry is case-sensitive; must be same case as used when table was created.
keyOnly		no	Applies to the update action: • yes: the WHERE criteria are limited to the key values. • no: the WHERE criteria include key values and the original values of changed fields.
password	Optional		Overrides password value specified in ODBC setup.
tableOwner	Optional		Table owner, if supported.

Attribute	Req/Opt	Default	Description
tableQualifier	Optional		Table qualifier, if supported. Purpose: • SQL Server and Oracle driver: name of database that contains the table. • Intersolv dBASE driver: directory of DBF files.
username	Optional		Overrides username value specified in ODBC setup.

Example

The following example lets you update a database by using a cfgrid tag to add and delete entire records or to update the data in individual cells. The cfgridupdate tag processes the data from the submitted form and updates the database.

```
<!--- If the gridEntered form field exists, the form was submitted. Perform gridupdate. --->
<cfif IsDefined("form.gridEntered") is True>
    <cfgridupdate grid = "FirstGrid" dataSource = "cfdocexamples" Keyonly="true"
        tableName = "CourseList">
 </cfif>

<!--- Query the database to fill up the grid. --->
<cfquery name = "GetCourses" dataSource = "cfdocexamples">
    SELECT Course_ID, Dept_ID, CorNumber, CorName, CorLevel, CorDesc
    FROM CourseList
    ORDER by Dept_ID ASC, CorNumber ASC
</cfquery>

<h3>cfgrid Example</h3>
<I>Try adding a course to the database, and then deleting it.</i>
<cfform>
<cfgrid name = "FirstGrid" width = "450"
    query = "GetCourses" insert = "Yes" delete = "Yes"
    font = "Tahoma" rowHeaders = "No"
    colHeaderBold = "Yes"
    selectMode = "EDIT"
    insertButton = "Insert a Row" deleteButton = "Delete selected row" >
</cfgrid><br>
<cfinput type="submit" name="gridEntered">
</cfform>...
```

cfheader

Description

Generates custom HTTP response headers to return to the client.

Category

Data output tags, Page processing tags

Syntax

```
<cfheader
    charset="character set"
    name = "header name"
    value = "header value">

OR

<cfheader
    statusCode = "status code"
    statusText = "status text">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcache`, `cfflush`, `cfhtmlhead`, `cfinclude`, `cfsetting`, `cfsilent`, `cfcontent`

History

ColdFusion MX 6.1: Changed behavior for the `name` attribute: `cfheader name="Content-Disposition"` uses the default file character encoding to encode this header's value, so the name of a file can include characters in the character encoding used in the file.

Attributes

Attribute	Req/Opt	Default	Description
charset	Optional	UTF-8	The character encoding in which to encode the header value. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information about character encodings, see www.w3.org/International/O-charset.html.
name	Required if statusCode not specified		Header name.
statusCode	Required if name not specified		Number. HTTP status code.
statusText	Optional		Explains the status code.
value	Optional		HTTP header value.

Usage

If you use this tag after the cfflush tag on a page, an error is thrown.

Example

```
<h3>cfheader Example</h3>

<p>cfheader generates custom HTTP response headers to return to the client.
<p>This example forces browser client to purge its cache of requested file.
<cfheader name="Expires" value="#GetHttpTimeString(Now())#">
```

cfhtmlhead

Description

Writes text to the `head` section of a generated HTML page.

Category

Page processing tags

Syntax

```
<cfhtmlhead
    text = "text">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfcache`, `cfflush`, `cfheader`, `cfinclude`, `cfsetting`, `cfsilent`

Attributes

Attribute	Req/Opt	Default	Description
`text`	Required		Text to add to the `<head>` area of an HTML page.

Usage

Use this tag for embedding JavaScript code, or putting other HTML tags, such as `meta`, `link`, `title`, or `base` in an HTML page header.

If you use this tag after the `cfflush` tag on a page, an error is thrown.

Example

```
<!--- This example displays the information provided by the Designer & Developer Center XML
    feed, http://www..com/devnet/resources/_resources.rdf
    See http://www..com/desdev/articles/xml_resource_feed.html for more information on this
    feed. --->

<!--- Set the URL address. --->
<cfset urlAddress="http://www..com/devnet/resources/_resources.rdf">

<!--- Use the CFHTTP tag to get the file content represented by urladdress.
    Note that />, not an end tag, terminates this tag. --->
<cfhttp url="#urladdress#" method="GET" resolveurl="Yes" throwOnError="Yes"/>

<!--- Parse the XML and output a list of resources. --->
<cfset xmlDoc = XmlParse(CFHTTP.FileContent)>

<!--- Get the array of resource elements, the xmlChildren of the xmlroot. --->
<cfset resources=xmlDoc.xmlroot.item>
<cfset numresources=ArrayLen(xmlDoc.xmlRoot.xmlChildren)-1>
<cfloop index="i" from="1" to="#numresources#">
<cfset item=resources[i]>
<cfoutput>
<strong><a href=#item.link.xmltext#>#item.title.xmltext#</strong></a><br>
<strong>Author</strong>  #item.creator.xmltext#<br>
<strong>Description</strong> #item.description.xmlText#<br>
<strong>Applies to these products</strong><br>
<cfloop index="i" from="1" to="#arrayLen(item.subject)#" step="1">
```

```
#item.subject[i].xmltext#<br>
</cfloop>
<br>
</cfoutput>
</cfloop>
```

cfhttp

Description

Generates an HTTP request and handles the response from the server.

Category

Internet protocol tags

Syntax

```
<cfhttp
    url = "server URL"
    charset = "character encoding"
    clientCert = "filename"
    clientCertPassword = "password"
    columns = "query columns"
    delimiter = "character"
    file = "filename"
    firstrowasheaders = "yes|no"
    getAsBinary = "auto|yes|no|never"
    method = "method name"
    multipart = "yes|no"
    name = "query name"
    password = "password"
    path = "path"
    port = "port number"
    proxyServer = "host name"
    proxyPort = "port number"
    proxyUser = "username"
    proxyPassword = "password"
    redirect = "yes|no"
    resolveURL = "yes|no"
    result = "result name"
    textQualifier = "character"
    throwOnError = "yes|no"
    timeout = "time-out period in seconds"
    username = "username"
    userAgent = "user agent">

    cfhttpparam tags [optional for some methods]

</cfhttp>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfhttpparam`, `GetHttpRequestData`, `cfftp`, `cfldap`, `cfmail`, `cfpop`, `SetEncoding`

History

ColdFusion 8: Added the `clientCert` and `clientCertPassword` attributes.

ColdFusion MX 7.01: Added the "never" value of the `getAsBinary` attribute.

ColdFusion MX 7: Added the `result` attribute, which allows you to specify an alternate variable in which to receive a result.

ColdFusion MX 6.1:

- Added support for the following methods: HEAD, PUT, DELETE, OPTIONS, TRACE.
- Added `multipart`, `getAsBinary`, `proxyUser`, and `proxyPassword` attributes.
- Changed `httpparam` behavior: all operations can have `httpparam` tags.
- Added the `cfhttp.errorDetail` return variable.
- Modified response body content types considered to be text.
- Changed behavior for multiple headers: multiple headers of the same type are returned in an array.
- Added support for HTTPS proxy tunneling.
- Fixed bugs in code and documentation.

ColdFusion MX:

- Added the `charset` and `firstrowasheaders` attributes.
- Changed Secure Sockets Layer (SSL) support: ColdFusion uses the Sun JSSE library, which supports 128-bit encryption, to support SSL.

Attributes

The following attributes control the HTTP transaction and can be used for all HTTP methods:

Attribute	Req/Opt	Default	Description
`url`	Required	Uses the http protocol	Address of the resource on the server that handles the request. The URL must include the hostname or IP address. If you do not specify the transaction protocol (http:// or https://), ColdFusion uses the default protocol, http. If you specify a port number in this attribute, it overrides any `port` attribute value. The `cfhttpparam` tag `URL` attribute appends query string attribute-value pairs to the URL.
`charset`	Optional	For request: `UTF-8` For response: `charset` specified by response Content- Type header, or `UTF-8` if response does not specify charset.	The character encoding of the request, including the URL query string and form or file data, and the response. The following list includes commonly used values: • `utf-8` • `iso-8859-1` • `windows-1252` • `us-ascii` • `shift_jis` • `iso-2022-jp` • `euc-jp` • `euc-kr` • `big5` • `euc-cn` • `utf-16` For more information character encodings, see www.w3.org/International/O-charset.html.
`clientCert`	Optional		The full path to a PKCS12 format file that contains the client certificate for the request.

Attribute	Req/Opt	Default	Description
clientCertPassword	Optional		Password used to decrypt the client certificate.
getAsBinary	Optional	no	• no: if ColdFusion does not recognize the response body type as text, converts it to a ColdFusion object. • auto: if ColdFusion does not recognize the response body type as text, converts it to ColdFusion Binary type data. • yes: always converts the response body content into ColdFusion Binary type data, even if ColdFusion recognizes the response body type as text. • never: prevents the automatic conversion of certain MIME types to the ColdFusion Binary type data; treats the returned content as text. ColdFusion recognizes the response body as text if: • the header does not specify a content type. • the content type starts with "text". • the content type starts with "message". • the content type is "application/octet-stream". If ColdFusion does not recognize the body as text and converts it to an object, but the body consists of text, the cfoutput tag can display it. The cfoutput tag cannot display Binary type data. (To convert binary data to text, use the ToString function.)
method	Optional	GET	• GET: requests information from the server. Any data that the server requires to identify the requested information must be in the URL or in cfhttp type="URL" tags. • POST: sends information to the server for processing. Requires one or more cfhttpparam tags. Often used for submitting form-like data. • PUT: requests the server to store the message body at the specified URL. Use this method to send files to the server. • DELETE: requests the server to delete the specified URL. • HEAD: identical to the GET method, but the server does not send a message body in the response. Use this method for testing hypertext links for validity and accessibility, determining the type or modification time of a document, or determining the type of server. • TRACE: requests that the server echo the received HTTP headers back to the sender in the response body. Trace requests cannot have bodies. This method enables the ColdFusion application to see what is being received at the server, and use that data for testing or diagnostic information. • OPTIONS: a request for information about the communication options available for the server or the specified URL. This method enables the ColdFusion application to determine the options and requirements associated with a URL, or the capabilities of a server, without requesting any additional activity by the server.
password	Optional		Use to pass a password to the target URL for Basic Authentication. Combined with username to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
port	Optional	80 for http 443 for https	Port number on the server to which to send the request. A port value in the url attribute overrides this value.
proxyServer	Optional		Host name or IP address of a proxy server to which to send the request.
proxyPort	Optional	80	Port number to use on the proxy server.
proxyUser	Optional		User name to provide to the proxy server.

Attribute	Req/Opt	Default	Description
proxyPassword	Optional		Password to provide to the proxy server.
redirect	Optional	yes	If the response header includes a Location field AND ColdFusion receives a 300-series (redirection) status code, specifies whether to redirect execution to the URL specified in the field: • yes: redirects execution to the specified page. • no: stops execution and returns the response information in the cfhttp variable, or throws an error if the throwOnError attribute is True. The cfhttp.responseHeader.Location variable contains the redirection path. ColdFusion follows a maximum of four redirects on a request. If there are more, ColdFusion functions as if redirect = "no". **Note:** The cflocation tag generates an HTTP 302 response with the url attribute as the Location header value.
resolveURL	Optional	no	• no: does not resolve URLs in the response body. As a result, any relative URL links in the response body do not work. • yes: resolves URLs in the response body to absolute URLs, including the port number, so that links in a retrieved page remain functional. Applies to these HTML tags: • img • src • a href • form action • applet code • script src • embed src • embed pluginspace • body background • frame src • bgsound src • object data • object classid • object codebase • object usemap Does not resolve URLs if the file and path attributes are used.
throwOnError	Optional	no	• yes: if the server returns an error response code, throws an exception that can be caught using the cftry and cfcatch or ColdFusion error pages. • no: does not throw an exception if an error response is returned. In this case, your application can use the cfhttp.StatusCode variable to determine if there was an error and its cause.

Attribute	Req/Opt	Default	Description
timeout	Optional		Value, in seconds, that is the maximum time the request can take. If the time-out passes without a response, ColdFusion considers the request to have failed. If the client specifies a time-out in the URL search parameter (for example, ?RequestTime=120) ColdFusion uses the lesser of the URL time-out and the timeout attribute value; this ensures that the request times out before, or at the same time as, the page. If the URL does not specify a time-out, ColdFusion uses the lesser of the Administrator time-out and the timeout attribute value. If the time-out is not set in any of these, ColdFusion waits indefinitely for the cfhttp request to process.
userAgent	Optional	ColdFusion	Text to put in the user agent request header. Used to identify the request client software. Can make the ColdFusion application appear to be a browser.
username	Optional		Use to pass a user name to the target URL for Basic Authentication. Combined with password to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.

The following attribute is used with the PUT method to determine how to send data specified with `httpparam type="formField"`:

Attribute	Req/Opt	Default	Description
multipart	Optional	no (Sends as multipart only if request includes File type data.)	Tells ColdFusion to send all data specified by cfhttpparam type="formField" tags as multipart form data, with a Content-Type of multipart/form-data. By default, ColdFusion sends cfhttp requests that contain only formField data with a Content Type of application/x-www-form-urlencoded. (If the request also includes File type data, ColdFusion uses the multipart/form-data content type for all parts.) If yes, ColdFusion also sends the request's charset in each Content-Type description. All form field data must be encoded in this character encoding, and ColdFusion does not URLEncode the data. (The field name must be in ISO-88591-1 or ASCII.) Some http parsers, including the one used by previous versions of ColdFusion, ignore the multipart form field character encoding description.

The following attribute allows you to specify the name of the variable in which you would like the results of the operation returned. The name you specify replaces `cfhttp` as the prefix by which you access the returned variables. For example, if you set the `result` attribute to `myResult`, you would access `FileContent` as `#myResult.FileContent#`.

The `result` attribute allows functions or CFCs that are called from multiple pages at the same time to avoid overwriting the results of one call with another. For information about the variables returned by a `cfhttp` get operation, see Variables returned by a cfhttp get operation in the Usage section.

Attribute	Req/Opt	Default	Description
result	Optional		Specifies the name of the variable in which you want the result returned.

The following attributes tell ColdFusion to put the HTTP response body in a file. You can put the response body in a file for GET, POST, PUT, DELETE, OPTIONS, and TRACE methods, but it is generally not useful with the DELETE or OPTIONS method.

Attribute	Req/Opt	Default	Description
file	Required if path is specified and not a GET method	See Description	Name of the file in which to store the response body. For a GET operation, the default is the file requested in the URL, if there is one. For example, if the URL in a GET method is http:www.myco.com/test.htm, the default file is test.htm. Do not specify the path to the directory in this attribute; use the path attribute.
path	Required if file is specified.		Tells ColdFusion to save the HTTP response body in a file. Contains the absolute path to the directory in which to store the file.

The following attributes tell ColdFusion to convert the HTTP response body into a ColdFusion query object. They can be used with the GET and POST methods only.

Attribute	Req/Opt	Default	Description
columns	Optional	First row of response contains column names.	The column names for the query, separated by commas, with no spaces. Column names must start with a letter. The remaining characters can be letters, numbers, or underscore characters (_). If there are no column name headers in the response, specify this attribute to identify the column names. If you specify this attribute, and the firstrowasHeader attribute is True (the default), the column names specified by this attribute replace the first line of the response. You can use this behavior to replace the column names retrieved by the request with your own names. If a duplicate column heading is encountered in either this attribute or in the column names from the response, ColdFusion appends an underscore to the name to make it unique. If the number of columns specified by this attribute does not equal the number of columns in the HTTP response body, ColdFusion generates an error.
delimiter	Optional	, (comma)	A character that separates query columns. The response body must use this character to separate the query columns.
firstrowasheaders	Optional	yes	Determines how ColdFusion processes the first row of the query record set: • yes: processes the first row as column heads. If you specify a columns attribute, ColdFusion ignores the first row of the file. • no: processes the first row as data. If you do not specify a columns attribute, ColdFusion generates column names by appending numbers to the word "column"; for example, "column_1".
name	Optional		Tells ColdFusion to create a query object with the given name from the returned HTTP response body.
textQualifier	Optional	" [double-quotation mark]	A character that, optionally, specifies the start and end of a text column. This character must surround any text fields in the response body that contain the delimiter character as part of the field value. To include this character in column text, escape it by using two characters in place of one. For example, if the qualifier is a double-quotation mark, escape it as " ".

Usage

The cfhttp tag is a general-purpose tool for creating HTTP requests and handling the returned results. It enables you to generate most standard HTTP request types. You use embedded cfhttpparam tags to specify request headers and body content.

When ColdFusion receives a response to a cfhttp request, it can put the response body (if any) in a file or the cfhttp.FileContent string variable. If the body text is structured as a result set, ColdFusion can put the body text in query object. You can also access the values of all returned headers and specify how to handle error status and redirections, and specify a time-out to prevent requests from hanging.

The HTTP protocol is the backbone of the World Wide Web and is used for every web transaction. Because the cfhttp tag can generate most types of requests, it provides significant flexibility. Possible uses include:

- Interacting with dynamic web sites and services that are not available as web services. (Use the cfinvoke tag to access SOAP web services.)
- Getting the contents of an HTML page or other file such as an image on a web server for use in your CFML page or storage in a file.
- Sending a secure request to a server by specifying the https protocol in the url attribute.
- Using the POST method to send a multipart/form-data style post to any URL that can handle such data and return results, including CGI executables or even other ColdFusion pages.
- Using the PUT method to upload files to a server that does not accept FTP requests.

This tag can, and for PUT and POST requests must, have a body that contains cfhttpparam tags. If this tag has cfhttpparam tags, it must have a </cfhttp> end tag.

To use HTTPS with the cfhttp tag, you might need to manually import the certificate for each web server into the keystore for the JRE that ColdFusion uses. This procedure should not be necessary if the certificate is signed (issued) by an authority that the JSSE (Java Secure Sockets Extension) recognizes (for example, Verisign); that is, if the signing authority is in the cacerts already. However, you might need to use the procedure if you are issuing SSL (secure sockets layer) certificates yourself.

Manually import a certificate

1. Go to a page on the SSL server in question.
2. Double-click the lock icon.
3. Click the Details tab.
4. Click Copy To File.
5. Select the base64 option and save the file.
6. Copy the CER file into C:\CFusionMX7\runtime\jre\lib\security (or whichever JRE ColdFusion is using).
7. Run the following command in the same directory (keytool.exe is located in C:\CFusionMX7\runtime\jre\bin):

```
keytool -import -keystore cacerts -alias giveUniqueName -file filename.cer
```

Variables returned by a cfhttp get operation

The cfhttp tag returns the following variables. If you set the result attribute, the name you assign replaces cfhttp as the prefix. For additional information, see the result attribute.

Name	Description
cfhttp.charSet	Response character character set (character encoding) specified by the response Content-Type header.
cfhttp.errorDetail	If the connection to the HTTP server fails, contains details about the failure. For instance: "Unknown host: my.co.com"; otherwise, the empty string. recommends that you check this variable for an error condition before checking other variables.
cfhttp.fileContent	Response body; for example, the contents of a html page retrieved by a GET operation. Empty if you save the response in a file.
cfhttp.header	Raw response header containing all header information in a single string. Contains the same information as the cfhttp.responseHeader variable.
cfhttp.mimeType	MIME type specified by the response Content-Type header; for example, text/html.
cfhttp.responseHeader	The response headers formatted into a structure. Each element key is the header name, such as Content-Type or Status_Code. If there is more than one instance of a header type, the type values are put in an array. One common technique is to dynamically access the cfhttp.responseHeader structure as a dynamic array; for example, #cfhttp.resonseHeader[fieldVariable]#.
cfhttp.statusCode	The HTTP status_code header value followed by the HTTP Explanation header value; for example, "200 OK".
cfhttp.text	Boolean; true if the response body content type is text. ColdFusion recognizes the response body as text in the following situations: • if the header does not specify a content type • if the content type starts with "text" • if the content type starts with "message" • if the content type is "application/octet-stream"

Building a query from a delimited text file

The cfhttp tag can create a ColdFusion query object form the response body. To do so, the response body must consist of lines of text, with each line having fields that are delimited by a character that identifies the column breaks. The default delimiter is a comma (,). The response data can also use a text qualifier; the default is a double-quotation mark ("). If you surround a string field in the text qualifier, the field can contain the delimiter character. To include the text qualifier in field text, escape it by using a double character. The following line shows a two-line request body that is converted into a query. It has three comma-delimited fields:

```
Field1,Field2,Field3
"A comma, in text","A quote: ""Oh My!""",Plain text
```

Run the following code to show how ColdFusion treats this data:

```
<cfhttp method="Get"
    url="127.0.0.1:8500/tests/escapetest.txt"
    name="onerow">
<cfdump var="#onerow#"><br>
```

Column names can be specified in three ways:

- By default, ColdFusion uses the first row of the response as the column names.
- If you specify a comma-delimited columns attribute, ColdFusion uses the names specified in the attribute as the column names. Set firstRowAsHeaders="no" if the first row of the response contains data. Otherwise, ColdFusion ignores the first row.

• If you do not specify a `columns` attribute and set `firstrowasheaders="no"`, ColdFusion generates column names of the form Column_1, Column2, etc.

The `cfhttp` tag checks to ensure that column names in the data returned by the tag start with a letter and contain only letters, numbers, and underscore characters (_).

ColdFusion checks for invalid column names. Column names must start with a letter. The remaining characters can be letters, numbers, or underscores (_). If a column name is not valid, ColdFusion generates an error.

Notes

• For the ColdFusion Administrator time-out and the URL time-out to take effect, you must enable the time-out in the ColdFusion Administrator, Server Settings page. For more information, see *Configuring and Administering ColdFusion*.

• The `cfhttp` tag supports Basic Authentication for all operations.

• The `cfhttp` tag uses SSL to negotiate secure transactions.

• If you put the HTTP response body in a file, ColdFusion does not put it in the CFHTTP.FileContent variable or generate a query object. If you do not put the response body in a file, ColdFusion puts it in the CFHTTP.FileContent variable; if you specify a `name` attribute ColdFusion generates a query object.

• The `cfhttp` tag does not support NTLM or Digest Authentication.

• If you are using Microsoft IIS, there is no HTTP header size limit. To specify an HTTP header size limit, you must set it in IIS.

Example

```
<!--- This example displays the information provided by
    the Designer & Developer Center XML feed,
    http://www..com/desdev/resources/_resources.xml
    See http://www..com/desdev/articles/xml_resource_feed.html
    for more information on this feed. --->

<!--- Set the URL address. --->
<cfset urlAddress="http://www..com/desdev/resources/_resources.xml">

<!--- Use the CFHTTP tag to get the file content represented by urladdress.
        Note that />, not an end tag, terminates this tag. --->
<cfhttp url="#urladdress#" method="GET" resolveurl="Yes" throwOnError="Yes"/>

<!--- Parse the XML and output a list of resources. --->
<cfset xmlDoc = XmlParse(CFHTTP.FileContent)>
<!--- Get the array of resource elements, the xmlChildren of the xmlroot. --->
<cfset resources=xmlDoc.xmlroot.xmlChildren>
<cfset numresources=ArrayLen(resources)>

<cfloop index="i" from="1" to="#numresources#">
    <cfset item=resources[i]>
    <cfoutput>
        <strong><a href=#item.url.xmltext#>#item.title.xmltext#</strong></a><br>
        <strong>Author</strong>  #item.author.xmltext#<br>
        <strong>Applies to these products</strong><br>
        <cfloop index="i" from="4" to="#arraylen(item.xmlChildren)#">
            #item.xmlChildren[i].xmlAttributes.Name#<br>
        </cfloop>
        <br>
    </cfoutput>
</cfloop>
```

cfhttpparam

Description

Allowed inside cfhttp tag bodies only. Required for cfhttp POST operations. Optional for all others. Specifies parameters to build an HTTP request.

Category

Internet protocol tags

Syntax

```
<cfhttpparam
    type = "transaction type"
    encoded = "yes|no"
    file = "filename"
    mimeType = "MIME type designator"
    name = "data name"
    value = "data value">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfhttp, GetHttpRequestData, cfftp, cfldap, cfmail, cfmailparam, cfpop

History

ColdFusion MX 6.1:

- Added the header and body types.
- Added the encoded and mimeType attributes.
- Changed HTTP method behavior: all HTTP methods can have httpparam tags.
- Changed the name attribute requirements: it is not required for all types.

Attributes

Attribute	Req/Opt	Default	Description
type	Required		Information type: • header: specifies an HTTP header. ColdFusion does not URL encode the header. • CGI: specifies an HTTP header. ColdFusion URL encodes the header by default. • body: specifies the body of the HTTP request. ColdFusion does not automatically set a content-type header or URL encode the body contents. To specify the content-type, use a separate cfhttp tag with type=header. • XML: identifies the request as having a content-type of text/xml. Specifies that the value attribute contains the body of the HTTP request. Used to send XML to the destination URL. ColdFusion does not URL encode the XML data. • file: tells ColdFusion to send the contents of the specified file. ColdFusion does not URL encode the file contents. • URL: specifies a URL query string name-value pair to append to the cfhttp url attribute. ColdFusion URL encodes the query string. • formField: specifies a form field to send. ColdFusion URL encodes the Form field by default. • cookie: specifies a cookie to send as an HTTP header. ColdFusion URL encodes the cookie.
encoded	Optional	yes	Applies to FormField and CGI types; ignored for all other types. Specifies whether to URL encode the form field or header.
file	Required only if type="File"		Applies to File type; ignored for all other types. The absolute path to the file that is sent in the request body.
mimeType	Optional		Applies to File type; invalid for all other types. Specifies the MIME media type of the file contents. The content type can include an identifier for the character encoding of the file; for example, text/html; charset=ISO-8859-1 indicates that the file is HTML text in the ISO Latin-1 character encoding.
name	Required. Optional (and ignored) for Body and XML types		Variable name for data that is passed. Ignored for Body and XML types. For File type, specifies the filename to send in the request.
value	Required. Optional (and ignored) for File type		Value of the data that is sent. Ignored for File type. The value must contain string data or data that ColdFusion can convert to a string for all type attributes except Body. Body types can have string or binary values.

Usage

Specifies header or body data to send in the HTTP request. The type attribute identifies the information that the parameter specifies. A cfhttp tag can have multiple cfhttpparam tags, subject to the following limitations:

- An XML type attribute cannot be used with additional XML type attributes, or with body, file, or formField type attributes.
- A body type attribute cannot be used with additional body type attributes, or with XML, file, or formField type attributes.
- The XML and body type attributes cannot be used with the cfhttp tag TRACE method.
- The file type attribute is only meaningful with the cfhttp tag POST and PUT methods.
- The formField type attribute is only meaningful with the cfhttp tag POST and GET methods.

If you send an HTTP request to a ColdFusion page, all HTTP headers, not just those sent using the CGI type, are available as CGI scope variables, However, any custom variables (such as "myVar") do not appear in a dump of the CGI scope.

When you send a file using the `type="file"` attribute, the file content is sent in the body of a multipart/form-data request. If you send the file to a ColdFusion page, the Form scope of the receiving page contains an entry with the name you specified in the `cfhttpparam` tag `name` attribute as the key. The value of this variable is the path to a temporary file containing the file that you sent. If you also send Form field data, the location of the filename in the form.fieldnames key list depends on the position of the `cfhttpparam` tag with the file relative to the `cfhttp` tags with the form data.

URL-encoding preserves special characters (such as the ampersand) when they are passed to the server. For more information, see the function "URLEncodedFormat" on page 1215.

To send arbitrary data in a "raw" HTTP message, use a `cfhttpparam` tag with a `type="body"` attribute to specify the body content and use `cfhttpparam` tags with a `type="header"` attributes to specify the headers.

Example

```
<!--- This example consists of two CFML pages.
        The first page posts to the second. --->

<!--- The first, posting page.
        This page posts variables to another page and displays the body of the response from
        the second page. Change the URL and port as necessary for your environment. --->

<cfhttp
        method="post"
        url="http://127.0.0.1/tests/http/cfhttpparamexample.cfm"
        port="8500"
        throwonerror="Yes">
    <cfhttpparam name="form_test" type="FormField" value="This is a form variable">
    <cfhttpparam name="url_test" type="URL" value="This is a URL variable">
    <cfhttpparam name="cgi_test" type="CGI" value="This is a CGI variable">
    <cfhttpparam name="cookie_test" type="Cookie" value="This is a cookie">
</cfhttp>

<!--- Output the results returned by the posted-to page. --->
<cfoutput>
    #cfhttp.fileContent#
</cfoutput>


<!--- This is the cfhttpparamexample.cfm page that receives and processes the Post request.
Its response body is the generated HTML output. --->

<h3>Output the passed variables</h3>
<cfoutput>
    Form variable: #form.form_test#
    <br>URL variable: #URL.url_test#
    <br>Cookie variable: #Cookie.cookie_test#
    <br>CGI variable: #CGI.cgi_test#<br>
    <br>Note that the CGI variable is URL encoded.
</cfoutput>
```

cfif

Description

Creates simple and compound conditional statements in CFML. Tests an expression, variable, function return value, or string. Used, optionally, with the cfelse and cfelseif tags.

Category

Flow-control tags

Syntax

```
<cfif expression>
    HTML and CFML tags
    <cfelseif expression>
    HTML and CFML tags
    <cfelse>
    HTML and CFML tags
</cfif>
```

See also

cfelse, cfelseif, cfabort, cfbreak, cfexecute, cfexit, cflocation, cfloop, cfswitch, cfthrow, cftry

Usage

If the value of the expression in the cfif tag is true, ColdFusion processes all the code that follows, up to any cfelseif or cfelse tag, and then skips to the cfif end tag. Otherwise, ColdFusion does not process the code that immediately follows the cfif tag, and continues processing at any cfelseif or cfelse tag, or with the code that follows the cfif end tag.

When testing the return value of a function that returns a Boolean, you do not have to define the True condition explicitly. This example uses the IsArray function:

```
<cfif IsArray(myarray)>
```

If successful, IsArray evaluates to yes, the string equivalent of the Boolean True. This is preferred over explicitly defining the True condition this way:

```
<cfif IsArray(myarray) IS True>
```

This tag requires an end tag.

Example

In this example, variables are shown within number signs. This is not required.

```
<!--- This example shows the interaction of cfif, cfelse, and cfelseif. --->
<!----- First, perform a query to get some data. ----->
<cfquery name="getCenters" datasource="cfdocexamples">
    SELECT Center_ID, Name, Address1, Address2, City, State, Country, PostalCode,
        Phone, Contact
    FROM Centers
    ORDER by City, State, Name
</cfquery>
<p>CFIF gives us the ability to perform conditional logic based on a condition
    or set of conditions.</p>
<p>For example, we can output the list of Centers from the snippets datasource
    by group and only display them <b>IF</b> City = San Diego.</p>
<hr>
<!---- Use CFIF to test a condition when outputting a query. ----->
<p>The following centers are in San Diego:</p>
```

```
<cfoutput query="getCenters">
        <cfif Trim(City) is "San Diego">
            <br><b>Name/Address:</b>#Name#, #Address1#, #City#, #State#
            <br><b>Contact:</b> #Contact#
            <br>
        </cfif>
</cfoutput>
<hr>
<p>If we would like more than one condition to be the case, we can ask for a list of the
centers in San Diego <b>OR</b> Santa Ana. If the center does not follow this condition, we
can use CFELSE to show only the names and cities of the other centers.</p>
<p>Notice how a nested CFIF is used to specify the location of the
    featured site (Santa Ana or San Diego).</p>
<!----- Use CFIF to specify a conditional choice for multiple options;
    also note the nested CFIF. --->
<p>Complete information is shown for centers in San Diego or Santa Ana.
    All other centers are listed in italic:</p>
<cfoutput query="getCenters">
    <cfif Trim(City) is "San Diego" OR Trim(City) is "Santa Ana">
        <h4>Featured Center in
            <cfif Trim(City) is "San Diego">
                San Diego
            <cfelse>
                Santa Ana
            </cfif>
            </h4> <b>Name/Address:</b>#Name#, #Address1#, #City#, #State#
            <br><b>Contact:</b> #Contact#<br>
    <cfelse>
        <br><i>#Name#, #City#</i>
    </cfif>
</cfoutput>
<hr>
<p>Finally, we can use CFELSEIF to cycle through a number of conditions and
produce varying output. Note that you can use CFCASE and CFSWITCH for a more
elegant representation of this behavior.
<!--- Use CFIF in conjunction with CFELSEIF to specify more than one
    branch in a conditional situation. --->
<cfoutput query="getCenters">
    <cfif Trim(City) is "San Diego" OR Trim(City) is "Santa Ana">
        <br><i>#Name#, #City#</i> (this one is in
                <cfif Trim(City) is "San Diego">San Diego
                <cfelse>Santa Ana
                </cfif>)
            <cfelseif Trim(City) is "San Francisco">
                <br><i>#Name#, #City#</i> (this one is in San Francisco)
            <cfelseif Trim(City) is "Suisun">
                <br><i>#Name#, #City#</i> (this one is in Suisun)
            <cfelse> <br><i>#Name#</i>
                <b>Not in a city we track</b>
            </cfif>
</cfoutput>
```

cfimage

Description

Creates a ColdFusion image. You can use the `cfimage` tag to perform common image manipulation operations as a shortcut to `Image` functions. You can use the `cfimage` tag independently or in conjunction with `Image` functions.

History

ColdFusion 8: Added this tag.

Category

Other tag

Syntax

```
Add a border to an image
<cfimage
    required
    action = "border"
    source = "absolute pathname|pathname relative to the web root|URL|#cfimage variable#"
    optional
    color = "hexadecimal value|web color"
    destination = "absolute pathname|pathname relative to the web root"
    isBase64 = "yes|no"
    name = "cfimage variable"
    overwrite = "yes|no"
    thickness = "number of pixels">

Create a CAPTCHA image
<cfimage
    required
    action = "captcha"
    height = "number of pixels"
    text = "text string"
    width = "number of pixels"
    optional
    destination = "absolute pathname|pathname relative to the web root"
    difficulty = "high|medium|low"
    overwrite = "yes|no"
    fonts = "comma-separated list of font names"
    fontSize = "point size">

Convert an image file format
<cfimage
    required
    action = "convert"
    destination = "absolute pathname|pathname relative to the web root"
    source = "absolute pathname|pathname relative to the web root"|URL|#cfimage variable#
    optional
    isBase64 = "yes|no"
    name = "cfimage variable"
    overwrite = "yes|no">

Retrieve information about an image
<cfimage
    required
    action = "info"
    source = "absolute pathname|pathname relative to the web root|URL|#cfimage variable#"
    structname = "structure name"
    optional
```

```
    isBase64 = "yes|no">

Read an image into memory
<cfimage
    required
    name = "cfimage variable"
    source = "absolute pathname|pathname relative to the web root|URL|#cfimage variable#"
    optional
    action = "read"
    isBase64 = "yes|no">

Resize an image
<cfimage
    required
    action = "resize"
    height = "number of pixels|percent%"
    source = "absolute pathname|pathname relative to the web root|URL|#cfimage variable#"
    width = "number of pixels|percent%"
    optional
    destination = "absolute pathname|pathname relative to the web root"
    isBase64 = "yes|no"
    name = "cfimage variable"
    overwrite = "yes|no">

Rotate an image
<cfimage
    required
    action = "rotate"
    angle = "angle in degrees"
    source = "absolute pathname|pathname relative to the web root|URL|#cfimage variable#"
    optional
    destination = "absolute pathname|pathname relative to the web root"
    isBase64= "yes|no"
    name = "cfimage variable"
    overwrite = "yes|no">

Write an image to a file
<cfimage
    required
    action = "write"
    destination = "absolute pathname|pathname relative to the web root"
    source = "absolute or relative pathname|URL|#cfimage variable#"
    optional
    isBase64= "yes|no"
    overwrite = "yes|no"
    quality = "JPEG image quality">

Write an image to the browser
<cfimage
    required
    action = "writeToBrowser"
    source = "absolute pathname|pathname relative to the web root|URL|#cfimage variable#"
    optional
    format = "png|jpg|jpeg"
    isBase64= "yes|no">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

ImageAddBorder, ImageInfo, ImageNew, ImageRead, ImageReadBase64, ImageResize, ImageRotate, ImageWrite, ImageWriteBase64 , "Creating and Manipulating ColdFusion Images" on page 765 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Action	Req/Opt	Default	Description
action	N/A	Optional	read	Action to take. Must be one of the following: • border • captcha • convert • info • read • resize • rotate • write • writeToBrowser The default action is read, which you do not need to specify explicitly.
angle	rotate	Required		Angle in degrees to rotate the image. You must specify an integer for the value.
color	border	Optional	black	Border color. Hexadecimal value or supported named color; see the name list in "Valid HTML named colors" on page 305. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxx", where x = 0-9 or A-F; use two number signs or none.
destination	border captcha convert *resize* rotate write	Optional (see Description)		Absolute or relative pathname where the image output is written. The image format is determined by the file extension. The convert and write actions require a destination attribute. The border, captcha, resize, and rotate actions require a name attribute or a destination attribute. You can specify both. ColdFusion 8 supports only CAPTCHA images in PNG format. If you do not enter a destination, the CAPTCHA image is placed inline in the HTML output and displayed in the web browser.
difficulty	captcha	Optional	low	Level of complexity of the CAPTCHA text. Specify one of the following levels of text distortion: • low • medium • high
fonts	captcha	Optional		One or more valid fonts to use for the CAPTCHA text. Separate multiple fonts with commas. ColdFusion supports only the system fonts that the JDK can recognize. For example, TTF fonts in the Windows directory are supported on Windows.
fontSize	captcha	Optional	24	Font size of the text in the CAPTCHA image. The value must be an integer.

Attribute	Action	Req/Opt	Default	Description
format	writeToBrowser	Optional	PNG	Format of the image displayed in the browser. If you do not specify a format, the image is displayed in PNG format. You cannot display a GIF image in a browser. GIF images are displayed in PNG format.
height	captcha resize	Required		Height in pixels of the image. For the resize attribute, you also can specify the height as a percentage (an integer followed by the percent (%) symbol). When you resize an image, if you specify a value for the width, you can let ColdFusion calculate the aspect ratio by specifying "" as the height. If specified, the value must be an integer.
isBase64	border convert info read resize rotate write writeToBrowser	Optional	no	Specifies whether the source is a Base64 string: • yes: the source is a Base64 string. • no: the source is not a Base64 string.
name	border convert read resize rotate	Optional (see Description)		Name of the ColdFusion image variable to create. The read action requires a name attribute. The border, resize, and rotate actions require a name attribute or a destination attribute. You can specify both.
overwrite	border captcha convert read resize rotate write	Optional	no	Valid only if the destination attribute is specified. The overwrite values are: • yes: overwrites the destination file. • no: does not overwrite the destination file. If the destination file already exists, ColdFusion generates an error if the overwrite action is not set to yes.
quality	write	Optional	0.75	Quality of the JPEG destination file. Applies only to files with an extension of JPG or JPEG. Valid values are fractions that range from 0 through 1 (the lower the number, the lower the quality).
source	border convert info read resize rotate write writeToBrowser	Required		• URL of the source image; for example, "http://www.google.com/images/logo.gif" • Absolute pathname or a pathname relative to the web root; for example: "c:\images\logo.jpg" • ColdFusion image variable containing another image, BLOB, or byte array; for example, "#myImage#" • Base64 string; for example, "....."
structName	info	Required		Name of the ColdFusion structure to be created.
text	captcha	Required		Text string displayed in the CAPTCHA image. Use capital letters for better readability. Do not include spaces because users cannot detect them in the resulting CAPTCHA image.

Attribute	Action	Req/Opt	Default	Description
thickness	border	Optional	1	Border thickness in pixels. The border is added to the outside edge of the source image, increasing the image area accordingly. The value must be an integer.
width	captcha resize	Required		Width in pixels of the image. For resize, you also can specify the width as a percentage (an integer followed by the % symbol). When you resize an image, if you specify a value for the height, you can let ColdFusion calculate the aspect ratio by specifying "" as the width. If specified, the value must be an integer.

Usage

ColdFusion 8 introduced the cfimage tag and the ColdFusion image, a construct native to ColdFusion that contains image data. You can manipulate ColdFusion images in memory and write them to a file, a database, or directly to a browser. You use the cfimage tag to create ColdFusion images from existing image files and perform simple image actions, such as rotating or resizing. Alternatively, you can use the ImageNew function to create a ColdFusion image from the beginning or from an existing image. You can use the Image functions to perform complex image manipulation operations on ColdFusion images that you create with the cfimage tag or with the ImageNew function.

You can perform the following tasks with ColdFusion images:

- Convert an image from one file format to another. For example, you can convert a BMP file to a JPEG file or a Base64 string to a GIF.
- Enforce consistent sizes on files uploaded to the server.
- Enforce size limits on JPEG images (by changing the quality of the image).
- Save a ColdFusion image to a file or write the image directly to a browser.
- Use the ImageGetBlob function within the cfquery tag to insert a ColdFusion image as a Binary Large Object Bitmap (BLOB) in a database. Also, you can extract a BLOB from a database and generate a ColdFusion image from it.
- Create watermark images.
- Create thumbnail images.
- Create a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) image, a distorted text image that is human-readable, but not machine-readable, used in a challenge-response test for preventing spam.

For more detailed examples, see "Creating and Manipulating ColdFusion Images" on page 765 in the *ColdFusion Developer's Guide*.

Supported image file formats

The cfimage tag operates on a number of different file formats. To list the formats that are supported on the server where the ColdFusion application is deployed, use the GetReadableImageFormats function and the GetWriteableImageFormats function.

Scorpio supports the following default image formats on Macintosh, Windows, and Unix operating systems:

- JPEG
- GIF

- TIFF
- PNG
- BMP

Scorpio does not support the following image formats:

- Animated GIF
- Multipage TIFF
- PSD
- AI

CMYK support

The `cfimage` tag supports reading and writing CMYK images, but does not support actions that require converting the images. For example, you can use CMYK images with the `read`, `write`, `writeToBrowser`, `resize`, `rotate`, and `info` actions. You cannot use CMYK images with the `convert`, `captcha`, and `border` actions. The same rule applies to image functions. For example, the `ImageNew`, `ImageRead`, and `ImageWrite` functions support CMYK images, but the `ImageAddBorder` function does not.

Valid HTML named colors

The following table lists the W3C HTML 4 named color value or hexadecimal values that the `color` attribute accepts:

Color name	RGB value
Black	##000000
Blue	##0000FF
Red	##FF0000
Gray	##808080
LightGray	##D3D3D3
DarkGray	##A9A9A9
Green	##008000
Pink	##FFC0CB
Cyan	##00FFFF
Magenta	##FF00FF
Orange	##FFA500
White	##FFFFFF
Yellow	##FFFF00

For all other color values, you must enter the hexadecimal value. Enter a six-digit value, which specifies the RGB value. Values between 00 and FF are allowed.

Image quality

By default, the `cfimage` tag generates images with antialiasing turned on (to remove the appearance of jagged edges). The interpolation method is set to `highestQuality`: this produces a high-quality image, but decreases processing speed. To turn off antialiasing, use the `ImageSetAntialiasing` function. To change the interpolation method or for more control over image attributes, use the following functions:

- ImageResize
- ImageRotate
- ImageScaleToFit
- ImageShear
- ImageTranslate

border action Use the `border` action to create a rectangular border around the outer edge of an image. You can control the thickness of the border and its color. For more control, use the `ImageAddBorder` function. The following example shows how to set the thickness and color of a border:

```
<!--- This example shows how to create a ColdFusion image from an existing JPEG file, add a
    five-pixel-wide red border to the image, and save it to a new JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" action="border" thickness="5"
    destination="jeff05.jpg" color="red" overwrite="yes">
```

captcha action Use the `captcha` action to create a distorted text image that is human-readable but not machine-readable. When you create a CAPTCHA image, you specify the text that is displayed in the CAPTCHA image; ColdFusion randomly distorts the text. You can specify the height and width of the text area, which affects the spacing between letters, the font size, the fonts used for the CAPTCHA text, and the level of difficulty, which affects readability. The following example shows how to write a CAPTCHA image directly to the browser:

```
<cfimage action="captcha" fontSize="25" width="400" height="150" text="rEadMe"
    fonts="Arial,Verdana,Courier New">
```

Note: *For the CAPTCHA image to display, the* `width` *value must be greater than:* `fontSize` *times the number of characters specified in* `text` *times 1.08. In this example, the minimum* `width` *is 162.*

ColdFusion 8 supports CAPTCHA images in PNG format only.

Note: *Use unique names for the CAPTCHA image files so that when multiple users access the CAPTCHA images, the files are not overwritten.*

The following example shows how to create CAPTCHA images with a medium level of difficulty that are written to files:

```
<!--- Use the GetTickCount function to generate unique names for the CAPTCHA files. --->
<cfset tc = GetTickCount()>
<cfimage action="captcha" fontSize="15" width="180" height="50" text="rEadMe"
    destination="images/rEadMe#tc#.png" difficulty="medium">
```

For a detailed example, see the "Creating and Manipulating ColdFusion Images" on page 765 in the *ColdFusion Developer's Guide*.

convert action Use the `convert` action to convert an image from one file format to another. For more information on file formats, see "Supported image file formats" on page 304. The following example shows how to convert a JPEG file to a PNG file:

```
<!--- This example shows how to convert a JPEG image to a PNG image. --->
<cfimage source="../cfdocs/images/artgallery/aiden02.jpg" action="convert"
    destination="aiden02.png">
```

Note: *Converting images between one file format to another is time-consuming. Also, image quality can degrade; for example, PNG images support 24-bit color, but GIF images support only 256 colors. Converting transparent images (images with alpha) can degrade image quality.*

info action Use the `info` action to create a ColdFusion structure that contains information about the image, including the color model, height, width, and source of the image. The structure is the same as returned by the `ImageInfo` function. The following example shows how to retrieve all of the information about an image:

```
<!--- This example shows how to retrieve and display image information. --->
<cfimage source="../cfdocs/images/artgallery/viata03.jpg" action="info"
structName="viatoInfo">
<cfdump var="#viatoInfo#">

<!--- Alternatively, you can use the cfoutput tag to display specific image information, as
    shown in the following example. --->
<cfoutput>
    <p>height: #viatoInfo.height# pixels</p>
    <p>width: #viatoInfo.width# pixels</p>
    <p>source: #viatoInfo.source#</p>
    <p>transparency: #viatoInfo.colormodel.transparency#</p>
    <p>pixel size: #viatoInfo.colormodel.pixel_size#</p>
    <p>color model: #viatoInfo.colormodel.colormodel_type#</p>
    <p>alpha channel support: #viatoInfo.colormodel.alpha_channel_support#</p>
    <p>color space: #viatoInfo.colormodel.colorspace#</p>
</cfoutput>
```

read action Use the `read` action to read an image from the specified local file pathname or URL, and create a ColdFusion image in memory. You can use the ColdFusion image variable as the source for another `cfimage` tag or for `Image` functions. The `read` action performs the same operation as the `ImageRead` function. The following example shows how to create a ColdFusion image from a JPEG file and manipulate it using the `ImageGrayscale` function:

```
<!--- This code shows how to create a ColdFusion image from a JPEG file.
    --->
<cfimage source="../cfdocs/images/artgallery/jeff01.jpg" name="myLogo">
<!--- This code shows how to convert the image to grayscale. --->
<cfset ImageGrayscale(myImage)>
<!--- This code shows how to write the grayscale image to a JPEG file. --->
<cfimage source="#myImage#" action="write" destination="myGrayscaleImage.jpg"
overwrite="yes">
```

resize action Use the `resize` action to resize an image to the specified height and width. You can specify the height and width in pixels or as a percentage:

```
<!--- This example shows how to specify the height and width of an image in pixels. --->
<cfimage source="../cfdocs/images/artgallery/jeff01.jpg" action="resize" width="100"
height="100" destination="jeff01_thumbnail.jpg" overwrite="yes">
<!--- This example shows how to specify the height and width of an image as percentages. --->
<cfimage source="../cfdocs/images/artgallery/jeff02.jpg" action="resize"
    width="50%" height="50%" destination="jeff02_thumbnail.jpg" overwrite="yes">
<!--- This example shows how to specify the height of an image in pixels and its width as a
    percentage. --->
<cfimage source="../cfdocs/images/artgallery/jeff03.jpg" action="resize"
    width="50%" height="100" destination="jeff03_thumbnail.jpg" overwrite="yes">
```

For more control of resize attributes, use the `ImageResize` function.

rotate action Use the `rotate` action to rotate an image by degrees:

```
<!--- This example shows how to rotate an image by 30 degrees. --->
<cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" action="rotate" angle="30"
    name="maxwellAngle">
<!--- Display the rotated image in a browser. --->
<cfimage source="#maxwellAngle#" action="writeToBrowser">
```

For more control of the rotate attributes, use the `ImageRotate` function.

write action Use the `write` action to write an image to the specified path. The new image is converted to the file type specified in the destination attribute. The `write` action performs the same operation as the `ImageWrite` function. When you write an image to a JPEG file, the image quality is set to 75% of the original image by default. To control the image size, use the `quality` attribute of the `write` action.

You can use the write action to change the quality of a JPEG image to reduce file size. The following example shows how to change image quality to .5:

```
<!--- This example shows how to create a PNG file from a JPEG file by using the write action.
--->
<cfimage source="../cfdocs/images/artgallery/aiden01.jpg" action="write"
    destination="aiden01.png">
<!--- This example shows how to create a low-quality JPEG image. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" action="write"
    destination="jeff05_lq.jpg" quality=".5">
<!--- This example shows how to write a JPEG file to a new location. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" action="write"
    destination="jeff05.jpg">
```

writeToBrowser action Use the `writeToBrowser` action to display one or more ColdFusion images directly to the browser without writing them to files. Images are displayed in PNG format. The following example shows how to reduce the size of an image and display it in the browser:

```
<!--- This example shows how to create a ColdFusion image from a JPEG file, resize it, and
    then display it in the browser as a PNG image. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" action="resize"
    width="50%" height="50%" name="smLogo">
<cfimage source="#smLogo#" action="writeToBrowser">
```

Example

This example shows how to create a ColdFusion image and manipulate it by using `Image` functions:

```
<!--- Create the ColdFusion image variable "myImage" from a JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Pass the ColdFusion image to the Image functions to blur the image by a radius of 5,
    flip the image 90 degrees, and convert the image to grayscale. --->
<cfset ImageBlur(myImage,5)>
<cfset ImageFlip(myImage,"90")>
<cfset ImageGrayscale(myImage)>
<!--- Write the transformed image to a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

cfimpersonate

Description

This tag is obsolete. Use the newer security tools; see “Conversion functions” on page 641 and “Securing Applications” on page 312 in the *ColdFusion Developer’s Guide*.

History

ColdFusion MX: This tag is obsolete. It does not work in ColdFusion MX and later releases.

cfimport

Description

You can use the cfimport tag to import either of the following:

- All ColdFusion pages in a directory, as a tag custom tag library.
- A Java Server Page (JSP) tag library. A JSP tag library is a packaged set of tag handlers that conform to the JSP 1.1 tag extension API.

Category

Application framework tags

Syntax

```
<cfimport
    prefix = "custom"
    taglib = "tag library location">
```

See also

cfapplication

History

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
taglib	Required		Tag library URI. The path must be relative to the web root (and start with /), the current page location, or a directory specified in the Administrator ColdFusion mappings page. • A directory in which custom ColdFusion tags are stored. In this case, all the cfm pages in this directory are treated as custom tags in a tag library. • A path to a JAR in a web-application, for example, "/WEB-INF/lib/sometags.jar" • A path to a tag library descriptor, for example, "/sometags.tld" **Note:** You must put JSP custom tag libraries in the /WEB-INF/lib directory. This limitation does not apply to ColdFusion pages.
prefix	Required		Prefix by which to access the imported custom CFML tags JSP tags. If you import a CFML custom tag directory and specify an empty value, " ", for this attribute, you can call the custom tags without using a prefix. You must specify and use a prefix for a JSP tag library.

Usage

The following example imports the tags from the directory myCustomTags:

```
<cfimport
    prefix="mytags"
    taglib="myCustomTags">
```

You can import multiple tag libraries using one prefix. If there are duplicate tags in a library, the first one takes precedence.

JSP tags have fixed attributes; however, if the tag supports runtime attribute expressions, most tag libraries support the use of the syntax #expressions#.

To reference a JSP tag in a CFML page, use the syntax <prefix:tagname>. Set the prefix value in the prefix attribute.

Use JSP custom tags in a ColdFusion page

1 Put a JSP tag library JAR file (for example, myjsptags.jar) into the ColdFusion server directory wwwroot/WEB-INF/lib. If the tag library has a separate TLD file, put it in the same directory as the JAR file.

2 At the top of a CFML page, insert code such as the following:

```
<cfimport
    prefix="mytags"
    taglib="/WEB-INF/lib/myjsptags.jar">
```

To reference a JSP tag from a JAR file, use the following syntax:

```
<cfoutput>
    <mytags:helloTag message="#mymessage#" />
<cfoutput>
```

The cfimport tag must be on the page that uses the imported tags. For example, if you use a cfimport tag on a page that you include with the cfinclude call, you cannot use the imported tags on the page that has the cfinclude tag. Similarly, if you have a cfimport tag on your Application.cfm page, the imported tags are available on the Application.cfm page only, not on the other pages in the application. ColdFusion does not throw an error in these situations, but the imported tags do not run.

You cannot use the cfimport tag to suppress output from a tag library.

For more information, see the Java Server Page 1.1 specification.

Example

```
<h3>cfimport example</h3>
<p>This example uses the random JSP tag library that is available from the
    Jakarta Taglibs project, at http://jakarta.apache.org/taglibs/</p>

<cfimport taglib="/WEB-INF/lib/taglibs-random.jar" prefix="randomnum">

<randomnum:number id="randPass" range="000000-999999" algorithm="SHA1PRNG" provider="SUN"/>
<cfset myPassword = randPass.random>
<cfoutput>
    Your password is #myPassword#<br>
</cfoutput>
```

cfinclude

Description

Embeds references to ColdFusion pages in CFML. You can embed `cfinclude` tags recursively. For another way to encapsulate CFML, see "cfmodule" on page 403. (A ColdFusion page was formerly sometimes called a ColdFusion template or a template.)

Category

Flow-control tags, Page processing tags

Syntax

```
<cfinclude
    template = "template name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcache`, `cfflush`, `cfheader`, `cfhtmlhead`, `cfsetting`, `cfsilent`

History

ColdFusion MX: Changed error behavior: if you use this tag to include a CFML page whose length is zero bytes, you do not get an error.

Attributes

Attribute	Req/Opt	Default	Description
`template`	Required		A logical path to a ColdFusion page.

Usage

ColdFusion searches for included files in the following locations:

1. In the directory of the current page or a directory relative to the current page
2. In directories mapped in the ColdFusion Administrator

You cannot specify an absolute URL or file system path for the file to include. You can only use paths relative to the directory of the including page or a directory that is registered in the ColdFusion Administrator Mappings. The following `cfinclude` statements work, assuming that the `myinclude.cfm` file exists in the specified directory:

```
<cfinclude template="myinclude.cfm">
<cfinclude template="../myinclude.cfm">
<cfinclude template="/CFIDE/debug/myinclude.cfm">
```

But these do not work:

```
<cfinclude template="C:\CFusion\wwwroot\doccomments\myinclude.cfm">
<cfinclude template="http://localhost:8500/doccomments/myinclude.cfm">
```

The included file must be a syntactically correct and complete CFML page. For example, to output data from within the included page, you must have a `cfoutput` tag, including the end tag, on the included page, not the referring page. Similarly, you cannot span a `cfif` tag across the referring page and the included page; it must be complete within the included page.

You can specify a variable for the `template` attribute, as the following example shows:

```
<cfset templatetouse="../header/header.cfm">
<cfinclude template="#templatetouse#">
```

Example

```
<!--- This example shows the use of cfinclude to paste CFML or HTML code into another page
    dynamically. --->

<h4>This example includes the dochome.htm page from the CFDOCS directory. The images do not
    display, because they are located in a separate directory. However, the page appears
    fully rendered within the contents of this page.</h4>
<cfinclude template = "../cfdocs/dochome.htm">
```

cfindex

Description

Populates a Verity collection with metadata and creates indexes for searching it. Verity is a search engine that you can integrate in a ColdFusion application to search physical files of various types or a database query. Indexing database columns that result from a query lets users search the query data much faster than they could if you used multiple SQL queries to return the same data.

You must define a Verity collection using the ColdFusion Administrator or the `cfcollection` tag before creating indexes for the collection.

You also can index a Verity collection using the ColdFusion Administrator or by using a native Verity indexing tool, such as Vspider or MKVDK. These options, however, limit you to indexing a collection of files in a directory path.

For more information on creating, indexing, and searching a Verity collection, see "Building a Search Interface" on page 460 in the *ColdFusion Developer's Guide*.

Category

Extensibility tags

Syntax

```
<cfindex
    action = "update|delete|purge|refresh"
    collection = "collection name"
    body = "body"
    category = "category name"
    categoryTree = "category tree"
    custom1 = "custom value"
    custom2 = "custom value"
    custom3 = "custom value"
    custom4 = "custom value"
    extensions = "file extensions"
    key = "ID"
    language = "language"
    prefix = "location of documents"
    query = "query name"
    recurse = "yes|no"
    status = "status"
    title = "title"
    type = "type"
    URLpath = "URL">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfcollection`, `cfexecute`, `cfobject`, `cfreport`, `cfsearch`, `cfwddx`

History

ColdFusion MX 7.0.1: Added the `prefix` attribute.

ColdFusion MX 7:

- Added the `category`, `categoryTree`, `custom3`, and `custom4` attributes for the `update` and `refresh` actions.
- Added the `status` attribute for the `update`, `refresh`, `delete`, and `purge` actions.

- Removed reference to external collections.
- Removed suggested `cflock` usage.

ColdFusion MX:

- The `action` attribute value `optimize` is obsolete. It does not work, and might cause an error, in ColdFusion MX.
- Changed the `external` attribute behavior: it is not necessary to specify the `external` attribute. (ColdFusion automatically detects whether a collection is internal or external.)
- Changed Verity operations behavior: ColdFusion supports Verity operations on Acrobat PDF files.
- Changed thrown exceptions: this tag can throw the SEARCHENGINE exception.
- Changed acceptable collection naming: this tag accepts collection names that include spaces.
- Changed query result behavior: the `cfindex` tag can index the query results from a `cfsearch` tag.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		• update: updates a collection and adds `key` to the index. • delete: removes collection documents as specified by the `key` attribute. • purge: deletes all of the documents in a collection. Causes the collection to be taken offline, preventing searches. • refresh: deletes all of the documents in a collection, and then performs an update.
`collection`	Required		Name of a collection that is registered by ColdFusion; for example, `"personnel"`.
`body`	Required if `type=custom`		• ASCII text to index. • Query column names, if name is specified in query. You can specify columns in a delimited list, for example: `"emp_name, dept_name, location"`. This attribute is ignored if `type` is `file` or `path`, and is invalid if `action` is `delete`.
`category`	Optional		A string value that specifies one or more search categories for which to index the data. You can define multiple categories, separated by commas, for a single index.
`categoryTree`	Optional		A string value that specifies a hierarchical category or category tree for searching. It is a series of categories separated by forward slashes ("/"). You can specify only one category tree.
`custom1`	Optional		Use to index discrete values in collection records, which lets you search for specific records using the Verity MATCHES operator. By contrast, values specified in the `body` attribute are concatenated and searched as a body of text using the specified criteria. If `type` = `query`, a query column name. If `type` = `custom`, a data field to be indexed.
`custom2`	Optional		Usage is the same as for `custom1`.
`custom3`	Optional		Usage is the same as for `custom1`.
`custom4`	Optional		Usage is the same as for `custom1`.
`extensions`	Optional	htm, html, cfm, cfml, dbm, dbml	Delimited list of file extensions that ColdFusion uses to index files, if `type` = `"Path"`. `"*."` returns files with no extension. `".*"` returns all files. For example, the following code returns files with a listed extension or no extension: `extensions == ".htm, .html, .cfm, .cfml, "*."`

Attribute	Req/Opt	Default	Description
key	Required	(empty string)	The value specified for key depends on the type attribute: • If type = "file", the directory path and filename for the file, • If type = "path", the directory path for the location of the files. • If type = "custom", a unique identifier that specifies the location of the data, For a query, the name of the column that holds the primary key, for example. If not a query, an identifier such as the URL for a web page, for example.
language	Optional	English	For options, see cfcollection. Requires the appropriate Verity Locales language pack (Western Europe, Asia, Multilanguage, Eastern Europe/Middle Eastern).
prefix	Optional		Specifies the location of files to index when the computer that contains the K2 Search Service is not the computer on which you installed ColdFusion, and when you index files with the type attribute set to path.
query	Optional.		The name of the query against which the collection is generated.
recurse	Optional	no	• yes: if type = "path", indexes qualified files in directories below the path specified in the key attribute. • no
status	Optional		The name of the structure into which ColdFusion MX returns status information.
title	Optional		Provides a title for the document if one cannot be extracted from the document.
type	Optional	custom, if query attribute is specified. Otherwise, file.	• file: applies action value to filename, including path. Expects a filename in the key attribute. • path: applies action to files in a directory path that pass the extensions filter. Expects a directory name in the key attribute. • *custom*: applies action to custom data; for example, to data from a query.
URLpath	Optional		If type is file or path, specifies the URL path. During indexing, this pathname is prefixed to filenames and returned from a search as the url.

Usage

The attributes settings that the cfindex tag requires depend on whether you set the query attribute. If you set the query attribute to a valid query name, it specifies that cfindex is to index the data in the query rather than indexing documents on a disk. If you do *not* set the query attribute, cfindex assumes it is indexing a file (type = file), a set of files in a directory path (type = path), or text that you provide in the body attribute (type = custom).

If you set the query attribute to a valid query name, the cfindex tag creates indexes as specified by the following attributes and their values:

Type	Attribute values
File	The key attribute is the name of a column in the query that contains a full filename (including path).
Path	The key attribute is the name of a column in the query that contains a directory pathname.
	The extensions and recurse attributes, if specified, elaborate on which files are included. If the action is delete, cfindex deletes keys for the collection.
Custom	The key attribute specifies a column name that contains anything you want; for example, the primary key value in the database. It must be unique because this is the primary key in the collection. If the action is delete, the key attribute is the name of a column in the query that contains the keys to delete.
	The body attribute is required and is a comma-delimited list of the names of the columns that contain the text data to be indexed.

If you do *not* set the `query` attribute, the `cfindex` tag creates indexes as specified by the following attributes and their values:

Type	Attribute values
File	The `key` attribute is required and is a full pathname to a file.
Path	The `key` attribute is required and it is a directory pathname.
	The `extensions` and `recurse` attributes, if specified, designate which types of files are included. If the action is `delete`, both the keys and the document files are deleted.
Custom	The `key` attribute is an identifier that specifies the key. If the action is `delete`, the `key` attribute is the document key to delete.
	The `body` attribute is required and is the text to be indexed.

If `type` is not specified but `query` is set, ColdFusion sets the `type` to the default value of `custom`.

If neither `type` nor `query` is set, ColdFusion sets `type` to the default value of `file`.

If `type` equals `custom`, all attributes except for `key` and `body` can specify a literal value, not only a column name. This allows you to change a field to empty in the collection.

Status attribute

The `status` attribute provides the following information and diagnostics about the result of a `cfindex` operation:

Key	Type	Description
BADKEYS	Struct	A structure of keys with diagnostic messages about the indexing of these keys. If there are no bad keys, this key does not exist.
DELETED	Number	The number of keys deleted.
MESSAGES	Array	An array of diagnostic messages, including nonfatal errors and warnings, returned from the Verity K2 Index server. If there are no messages, this key does not exist.
INSERTED	Number	The number of keys inserted into the collection.
UPDATED	Number	The number of keys updated in the collection.

Example

```
<!--- EXAMPLE #1 Index a file, type = "file". ---------------------------->
<!--- Example dumps content of status variable (info). ------------------->
<cfindex collection="CodeColl"
    action="refresh"
    type="file"
    key="C:\blackstone\wwwroot\vw_files\cfindex.htm"
    urlpath="http://localhost:8500/vw_files/"
    language="English"
    title="Cfindex Reference page"
    status="info">

<!--- Search for Attributes. --->
<cfsearch
    name = "mySearch"
    collection = "CodeColl"
    criteria = "Attributes"
    contextpassages = "1"
    maxrows = "100">
<cfoutput>
    key=#mySearch.key#<br />
```

```
    title=#mySearch.title#<br />
    context=#mySearch.context#<br />
    url=#mySearch.url#<br />
</cfoutput>

<cfdump var="#info#">

<!--- EXAMPLE #2 Index a path (type = "path"). ------------------------------>
<cfindex collection="CodeColl"
    action="refresh"
    type="path"
    key="C:\inetpub\wwwroot\vw_files\newspaper\sports"
    urlpath="http://localhost/vw_files/newspaper/sports"
    extensions = ".htm, .html"
    recurse="no"
    language="English"
    categoryTree="vw_files/newspaper/sports"
    category="Giants">

<!--- Search for any references to criteria. --->
<cfsearch
    name = "mySearch"
    collection = "CodeColl"
    categoryTree="vw_files/newspaper/sports"
    category="Giants"
    criteria = "Williams"
    contextpassages = "1"
    maxrows = "100">
<cfoutput>
    key=#mySearch.key#<br />
    title=#mySearch.title#<br />
    context=#mySearch.context#<br />
    url=#mySearch.url#<br />
</cfoutput>

<!---EXAMPLE #3: Index a QUERY (type = "custom") using custom1. ------------>
<!--- Retrieve data from the table. --->
<cfquery name="getCourses" datasource="cfdocexamples">
    SELECT * FROM COURSES
</cfquery>

<!--- Update the collection with the above query results. --->
<!--- key is Course_ID in the Courses table. ---->
<!--- body specifies the columns to be indexed for searching. --->
<!--- custom1 specifies the value of the Course_Number column. --->

<cfindex
    query="getCourses"
    collection="CodeColl"
    action="Update"
    type="Custom"
    key="Course_ID"
    title="Courses"
    body="Course_ID,Descript"
    custom1="Course_Number"
>
<h2>Indexing Complete</h2>
<!--- cno supplies value for searching custom1; could be form input instead. --->
<cfset cno = "540">
<cfsearch
    name = "mySearch"
```

```
    collection = "CodeColl"
    criteria = "CF_CUSTOM1 <MATCHES> #cno#"
    contextpassages = "1"
    maxrows = "100">
<!--- Returns indexed values (Course_ID and Descript) for
            Course_Number 540. --->
<cfoutput>
    key=#mySearch.key#<br />
    title=#mySearch.title#<br />
    context=#mySearch.context#<br />
    url=#mySearch.url#<br />
</cfoutput>

<!--- EXAMPLE #4 Index a FILE within a QUERY (type= "file"). --------------->
<!--- Retrieve row with a column that contains a filename (Contract_File). --->
<cfquery name="getEmps" datasource="cfdocexamples">
    SELECT * FROM EMPLOYEE WHERE EMP_ID = 1
</cfquery>

<!--- Update the collection with the above query results. --->
<!--- key specifies the column that contains a complete filename. --->
<!--- file is indexed in same way as if no query involved. --->
<cfindex
    query="getEmps"
    collection="CodeColl"
    action="Update"
    type="file"
    key="Contract_File"
    title="Contract_File"
    body="Emp_ID,FirstName,LastName,Contract_File">

<h2>Indexing Complete</h2>
<cfsearch
    name = "mySearch"
    collection = "CodeColl"
    criteria = "vacation"
    contextpassages = "1"
    maxrows = "100">
<cfoutput>
    key=#mySearch.key#<br />
    title=#mySearch.title#<br />
    context=#mySearch.context#<br />
    url=#mySearch.url#<br />
</cfoutput>

<!--- EXAMPLE # 5 Index a PATH within a QUERY. ---------------------------->
<!--- Retrieve a row with a column that contains a path (Project_Docs). --->
<cfquery name="getEmps" datasource="cfdocexamples">
    SELECT * FROM EMPLOYEE WHERE Emp_ID = 15
</cfquery>

<!--- Update the collection with the above query results. --->
<!--- key specifies a column that contains a directory path. --->
<!--- path is indexed in same way as if no query involved. --->
<cfindex
    query="getEmps"
    collection="CodeColl"
    action="update"
    type="path"
    key="Project_Docs"
    title="Project_Docs"
```

```
    body="Emp_ID,FirstName,LastName,Project_Docs">

<h2>Indexing Complete</h2>

<cfsearch
    name = "getEmps"
    collection = "CodeColl"
    criteria = "cfsetting"
    contextpassages = "1"
    maxrows = "100">
<cfoutput>
    key=#getEmps.key#<br />
    title=#getEmps.title#<br />
    context=#getEmps.context#<br />
    url=#getEmps.url#<br />
</cfoutput>

<!--- EXAMPLE #6 Deletes keys in the CodeColl collection for html files --->
<!--- in the specified directory (but not in subdirectories). ------------->

<cfindex collection="CodeColl"
    action="delete"
    type="path"
    key="C:\CFusion\wwwroot\vw_files\newspaper"
    urlpath="http://localhost:8500/vw_files/newspaper"
    extensions = ".htm, .html"
    recurse="no">

<!--- EXAMPLE #7 Purges all keys in the CodeColl collection --->
<!--- with recursion. ---------------------------------------------------------->

<cfindex collection="CodeColl"
    action="purge"
    type="path"
    key="C:\CFusion\wwwroot\vw_files\newspaper">
```

cfinput

Description

Used within the cfform tag, to place input controls that support input validation on a form.

Category

Forms tags

Syntax

```
<cfinput
    name = "name"
    autosuggest = "list or bind expression"
    autosuggestBindDelay = "integer number if seconds"
    autosuggestMinLength = "integer"
    bind = "bind expression"
    bindAttribute = "attribute name"
    bindOnLoad = "no|yes"
    checked = "yes|no"
    dayNames = "day of week labels separated by commas"
    delimiter = "character"
    disabled = "disabled"
    enabled = "yes|no"
    firstDayOfWeek = "day name"
    height = "number of pixels"
    id = "HTML id"
    label = "text"
    mask = "masking pattern"
    maxLength = "number"
    maxResultsDisplayed = "number"
    message = "text"
    monthNames = "month labels"
    onBindError = "JavaScript function name"
    onChange = "JavaScript or ActionScript"
    onClick = "JavaScript or ActionScript"
    onError = "script name"
    onKeyDown = "JavaScript or ActionScript"
    onKeyUp = "JavaScript or ActionScript"
    onMouseDown = "JavaScript or ActionScript"
    onMouseUp = "JavaScript or ActionScript"
    onValidate = "script name"
    pattern = "regular expression"
    range = "minimum value, maximum value"
    required = "yes|no"
    showAutosuggestLoadingIcon = "yes|no"
    size = "integer"
    sourceForToolTip = "URL"
    src = "image URL"
    style = "style specification"
    tooltip = "text"
    type = "input type"
    typeahead = "no|yes"
    validate = "data type"
    validateAt = "onBlur|onServer|onSubmit"
    value = "initial value"
    visible = "yes|no"
    width = "integer number of pixels">
```

Some attributes apply to only specific display formats. For details, see the Attributes table.

In HTML format forms, standard HTML input control attributes not listed above are passed to the HTML and have their normal effect.

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfajaximport`, `cfapplet`, `cfcalendar`, `cfform`, `cfformgroup`, `cfformitem`, `cfgrid`, `cfselect`, `cfslider`, `cftextarea`, `cftree`, "Using Ajax form controls and features" on page 627 in "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*

History

ColdFusion 8

- Added `autosuggest`, `autosuggestBindDelay`, `autosuggestMinLength`, `delimiter`, `maxResultsDisplayed`, `showAutosuggestLoadingIcon`, and `typeahead` attributes.
- Added support for the `bind` attribute in HTML forms and the `bindAttribute` and `bindOnload`, and `onBindError` attributes.
- Added the `sourceForTooltip` attribute
- Added support for `datefield` value of the `type` attribute in HTML forms.

ColdFusion MX 7:

- Added support for button, file, hidden, image, reset, and submit controls.
- Added support for generating Flash and XML controls (specified in the `cfform` tag).
- Added `datefield` type (Flash forms only) and the supporting `dayNames` and `monthNames` attributes.
- Added `bind`, `enabled`, `height`, `label`, `tooltip`, `visible`, and `width` attributes for use in Flash forms.
- Added support for onBlur and onServer validation, including the `validateAt` attribute.
- Added USdate, range, boolean, email, URL, uuid, guid, maxlength, noblanks, and submitOnce `validation` attribute values.
- Added support for preventing multiple submissions.
- Added the `mask` attribute.
- Deprecated the `passthrough` attribute. The tag now supports all HTML `input` tag attributes directly.

ColdFusion MX: Changed the `cfform` tag `preserveData` attribute behavior: if it is set to True, ColdFusion checks radio and check box values only if their value matches the posted value for the control. (In earlier releases, if the posted value did not match any of the `cfinput` check boxes or radio buttons for the control, the `checked` attribute was used.

Attributes

The following table lists attributes that ColdFusion uses directly. The tag also supports all HTML `form` tag attributes that are not on this list, and passes them directly to the browser.

Note: *Attributes that are not marked as supported in All or XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.*

Attribute	Req/Opt; formats	Default	Description
`name`	Required; all		Name for form input element.
`autosuggest`	Optional, HTML		Specifies entry completion suggestions to display as the user types into a text input. The user can select a suggestion to complete the text entry. The valid value can be either of the following: • A string that consists of the suggestion values separated by the delimiter specified by the `delimiter` attribute. • A bind expression that gets the suggestion values based on the current input text. The bind expression must pass a `cfautosuggestvalue` bind parameter to represent the user input. Valid only for `cfinput type="text"`. For more information, see "Using autosuggest text input fields" on page 645 in the *ColdFusion Developer's Guide*
`autosuggestBindDelay`	Optional, HTML	0.5 seconds	A nonzero integer that specifies the minimum time between `autosuggest` bind expression invocations, in seconds. This value also specifies the delay from when the user first enters a minimum-length entry in the field until the suggestion box appears. Use this attribute to limit the number of requests that are sent to the server when a user types. Valid only for `cfinput type="text"`. **Note:** The only way to get the default behavior is to omit the attribute. Otherwise, the delay must be a nonzero integer value.
`autosuggestMinLength`	Optional, HTML	1	The minimum number of characters required in the text box before invoking a bind expression to return items for suggestion. Valid only for `cfinput type="text"`.
`bind`	Optional; HTML, Flash		A bind expression that dynamically sets an attribute of the control. For details, see Usage.
`bindAttribute`	Optional; HTML	`value`	Specifies the HTML tag attribute whose value is set by the `bind` attribute. You can only specify attributes in the browser's HTML DOM tree, not ColdFusion-specific attributes. Ignored if there is no `bind` attribute. Valid only for `cfinput type="text"`.
`bindOnLoad`	Optional; HTML	`no`	A Boolean value that specifies whether to execute the `bind` attribute expression when first loading the form. Ignored if there is no `bind` attribute. Valid only for `cfinput type="text"`.
`checked`	Optional; all	`no`	Selects a radio button or check box control: • `yes` • `no` For HTML format, you can indicate that the item is selected by specifying the value as `checked` or specifying the attribute with no value.
`dayNames`	Optional; all	`S, M, T, W, T, F, S`	Applies to `datefield` type only. A comma-delimited list that sets the names of the weekdays displayed in the calendar. Sunday is the first day; the rest of the weekday names follow in the normal order.

Attribute	Req/Opt; formats	Default	Description
delimiter	Optional, HTML	comma (,)	The delimiter to use to separate entries in a static autosuggest list. This attribute is meaningful only if the autosuggest attribute is a string of delimited values.
disabled	Optional; all	not disabled	Disables user input, making the control read-only. The attribute behavior depends on the format of the form as follows: • HTML format forms: ColdFusion passes this attribute directly to the HTML. To disable input, specify disabled without a value (HTML format) or with the value disabled (XHTML compliant). To enable input, omit this attribute. • Flash format forms: To disable input, specify disabled without an attribute, or disabled="yes" (or any ColdFusion positive Boolean value, such as true). To enable input, omit the attribute or specify disabled="no" (or any ColdFusion negative Boolean value, such as false).
enabled	Optional; Flash	yes	Boolean value that specifies whether the control is enabled. A disabled control appears in light gray. The inverse of the disabled attribute.
firstDayOfWeek	Optional; all	0	Applies to datefield type only. Integer in the range 0-6 that specifies the first day of the week in the calendar: 0 indicates Sunday; 6 indicates Saturday.
height	Optional; see Description		Applies to most Flash types, HTML image type on some browsers. The height of the control, in pixels. The displayed height might be less than the specified size.
id	Optional; HTML	name attribute value	The HTML ID of the form.
label	Optional; Flash, XML		Label to put next to the control on a Flash form. Not used for button, hidden, image, reset, or submit types.
mask	Optional; Flash, HTML		For tags with type="text". A mask pattern that controls the character pattern that users can enter, or that the form sends to ColdFusion. Mask characters and the corresponding valid input characters are: • A = [A-Za-z] • X = [A-Za-z0-9] • 9 = [0-9] • ? = Any character • All other characters = insert the literal character For tags with type="datefield", a pattern that controls the format of dates that the user selects in the calendar. Mask characters are: • D = day; can use 0-2 mask characters. • M = month; can use 0-4 mask characters. • Y = year; can use 0, 2, or 4 characters. • E = day in week; can use 0-4 characters. For more information, see Masking input data in Usage.
maxLength	Optional; all		Maximum length of text entered, if type = "Text" or "password". For complete length validation, specify maxLength validation in a validate attribute; otherwise, this attribute prevents users from typing beyond the specified length, but does not prevent them from pasting in a longer value.

Attribute	Req/Opt; formats	Default	Description
maxResultsDisplayed	Optional; HTML	10	The maximum number suggestions to display in the autosuggest list. Valid only for `cfinput type="text"`.
message	Optional; all		Message text to display if validation fails.
monthNames	Optional; all	`January, February, March, April, May, June, July, August, September, October, November, December`	Applies to `datefield` type only. A comma-delimited list of the month names that display at the top of the calendar.
onBindError	Optional; HTML	See Description	The name of a JavaScript function to execute if evaluating a bind expression, including an autosuggest bind expression, results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and have specified a global error handler (by using the `ColdFusion.setGlobalErrorHandler` function), it displays the error message; otherwise a default error pop-up displays.
onChange	Optional; all		JavaScript (HTML/XML) or ActionScript (Flash) to run when the control changes due to user action. In Flash, applies to datefield, password, and text types only.
onClick	Optional; all		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user clicks the control. In Flash, applies to button, checkbox, image, radio, reset, and submit types only.
onError	Optional; HTML, XML		Name of a custom JavaScript function to execute if validation fails.
onKeyDown	Optional; all		JavaScript (HTML/XML) or ActionScript (Flash) ActionScript to run when the user presses a keyboard key in the control.
onKeyUp	Optional; all		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a keyboard key in the control.
onMouseDown	Optional; all		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a mouse button in the control.
onMouseUp	Optional; all		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user presses a mouse button in the control.
onValidate	Optional; HTML, XML		Name of a custom JavaScript function to validate user input. The form object, input object, and input object values are passed to the routine, which should return `true` if validation succeeds, and `false` otherwise. If used, the `validate` attribute is ignored.
pattern	Required if `validate="regex"`; HTML, XML		JavaScript regular expression pattern to validate input. ColdFusion uses this attribute only if you specify *regex* in the `validate` attribute. Omit leading and trailing slashes. For examples and syntax, see "Building Dynamic Forms with cfform Tags" on page 531 in the *ColdFusion Developer's Guide*.

Attribute	Req/Opt; formats	Default	Description
range	Optional; all		Minimum and maximum allowed numeric values. ColdFusion uses this attribute only if you specify *range* in the validate attribute. If you specify a single number or a single number followed by a comma, it is treated as a minimum, with no maximum. If you specify a comma followed by a number, the maximum is set to the specified number, with no minimum.
required	Optional; all	no	• yes: the field must contain data. • no: allows an empty field.
showAutosuggestLoadingIcon	Optional; HTML	true	A Boolean value that specifies whether to display an animated icon when loading an autosuggest value for a text input.
size	Optional; all		Size of input control. Ignored, if type = "radio" or "checkbox". If specified in a Flash form, ColdFusion sets the control width pixel value to 10 times the specified size and ignores the width attribute.
sourceForTooltip	Optional; HTML		The URL of a page to display as a tool tip. The page can include HTML markup to control the format, and the tip can include images. If you specify this attribute, an animated icon appears with the text "Loading..." while the tip is being loaded.
src	Optional; Flash, HTML		Applies to Flash button, reset, submit, and image types, and the HTML image type. URL of an image to use on the button.
style	Optional; all		In HTML or XML format, ColdFusion passes the style attribute to the browser or XML. In Flash format, must be a style specification in CSS format. For detailed information on specifying Flash styles, see "Creating Forms in Flash" on page 577 in the *ColdFusion Developer's Guide*. In XML format, ColdFusion passes the style attribute to the XML.
tooltip	Optional; Flash, HTML		Text to display when the mouse pointer hovers over the control. Ignored if you specify a sourceForTooltip attribute.
type	Optional; all	text	The input control type to create: • button: push button. • checkbox: check box. • datefield: HTML and Flash only; date entry field with an expanding calendar from which users select the date or dates. In HTML format only, users can also enter the date by typing in the field. • file: file selector; not supported in Flash. • hidden: invisible control. • image: clickable button with an image. • password: password entry control; hides input values. • radio: radio button. • reset: form reset button. • submit: form submission button. • text: text entry box.

Attribute	Req/Opt; formats	Default	Description
typeahead	Optional; HTML	no	A Boolean value that specifies whether the autosuggest feature should automatically complete a user's entry with the first result in the suggestion list. Valid only for cfinput type="text".
validate	Optional; all		The type or types of validation to do. Available validation types and algorithms depend on the format. For details, see Usage.
validateAt	Optional; all	onSubmit	How to do the validation; one or more of the following values: • onSubmit • onServer • onBlur The onBlur and onSubmit values are identical in Flash forms. For multiple values, use a comma-delimited list. For details, see Usage.
value	depends on type setting; all		HTML: corresponds to the HTML value attribute. Its use depends on control type. Flash: optional; specifies text for button type inputs: button, submit, and image.
visible	Optional; Flash	yes	Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank.
width	Optional; see Description		Applies to most Flash types, and HTML image type on some browsers. The width of the control, in pixels. For Flash forms, ColdFusion ignores this attribute if you also specify a size attribute value.

Note: *Attributes that are marked as not supported in XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.*

Usage

For this tag to work properly. the browser must be JavaScript-enabled.

If the cfform preserveData attribute is true and the form posts back to the same page, the posted value of the cfinput control is used, instead of its Value or Checked attribute.

You can use the keyboard to access and select dates from a datefield Flash input: press Tab to go to the field and press the Spacebar to open the menu. Use the Up, Down, Left, and Right Arrow keys to change the selected date. Use the Home and End keys to reach the first and last enabled date in a month, respectively. Use the Page Up and Page Down keys to reach the previous and next month, respectively.

Note: *To clear a datefield entry in Flash format forms, select the field to open the menu, and click the selected date.*

For more information, see cfform. For information on using JavaScript regular expressions with this tag, see "Building Dynamic Forms with cfform Tags" on page 531 in the *ColdFusion Developer's Guide*.

Validation

The following sections describe how to do validation in cfinput tags.

Validation methods ColdFusion provides four methods of validation of cfinput text and password fields.

You can specify one or a combination of the following in the validateAt attribute:

• `onSubmit` The form page on the browser includes JavaScript functions that perform validation before the form is submitted to the server. In Flash format forms, this option is identical to `onBlur`.

• `onBlur` In HTML format the form page on the browser includes JavaScript functions that perform validation when the field loses the focus. In Flash format, the attribute is equivalent to onSubmit. OnBlur validation uses the same algorithms as onSubmit validation. OnBlur validation was added in ColdFusion MX 7.

• `onServer` ColdFusion performs the validation on the server. Some onServer algorithms vary from the onSubmit algorithms. OnServer Date and Time validation allow more patterns than onSubmit validation. OnServer validation was added in ColdFusion MX 7, and automatically generates hidden fields to support the validation.

You can also omit a `validate` attribute and specify the type of validation for the field in a separate hidden form field. This form of validation is equivalent to onServer validation, but it lets you specify separate messages for each validation that you do on the field. It is backward compatible with previous ColdFusion releases. For more information on hidden form field validation, see `cfform` and "Validating form data using hidden fields" on page 566 in "Validating form data using hidden fields" on page 566 in the *ColdFusion Developer's Guide*.

Validation types Use the following values in the `validate` attribute to specify input validation for all validation methods. Most attributes apply only to password or text fields. You can specify multiple validation types in a comma-delimited list, but only some combinations are meaningful.

Type	Description
`date`	If `validateAt="onServer"`, allows any date format that returns `true` in the `IsDate` function; otherwise, same as USdate.
`USdate`	A US date of the format mm/dd/yy mm-dd-yy or mm.dd.yy, with 1-2 digit days and months, 1-4 digit years.
`eurodate`	A date of the format dd/mm/yy, with 1-2 digit days and months, 1-4 digit years. The format can use /, -, or . characters as delimiters.
`time`	Time format hh:mm:ss
`float` *Or* `numeric`	A number; allows integers.
`integer`	An integer.
`range`	A numeric range.
`boolean`	A value that can be converted to a Boolean value: Yes, No, True, False, or a number.
`telephone`	Standard U.S. telephone formats. Allows an initial 1 long-distance designator and up to 5-digit extensions, optionally starting with x.
`zipcode`	U.S. 5- or 9-digit ZIP code format #####-####. The separator can be a hyphen (-) or a space.
`creditcard`	Strips blanks and dashes; verifies number using mod10 algorithm. Number must have 13-16 digits.
`ssn` *Or* `social_security_number`	US. Social Security number format, ###-##-####. The separator can be a hyphen (-) or a space.
`email`	A valid e-mail address of the form name@server.domain. ColdFusion validates the format only; it does not check that entry is a valid active e-mail address.
`URL`	A valid URL pattern; supports http, https, ftp file, mailto, and news URLs.
`guid`	A unique identifier that follows the Microsoft/DCE format, xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, where x is a hexadecimal number.
`uuid`	A universally unique identifier (UUID) that follows the ColdFusion format, xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx, where x is a hexadecimal number.
`maxlength`	Limits the input to a maximum number of characters.
`noblanks`	Does not allow fields that consist only of blanks.

Type	Description
`regex` *or* `regular_expression`	Matches input against the `pattern` attribute. Valid in HTML and XML format only; ignored in Flash format.
`SubmitOnce`	Used only with submit and image types; prevents the user from submitting the same form multiple times before until the next page loads (for example, submitting an order a second time before getting the first order confirmation). Valid in HTML and XML format only; ignored in Flash format.

Validation differences The preceding table describes the general validation behavior. The underlying validation code must differ depending on the validation method and the form type. As a result, the algorithms used vary in some instances, including the following:

- The validation algorithms used for date/time values varies between onSubmit/OnBlur and OnServer.
- The algorithms used for onSubmit/OnBlur validation in Flash vary from those used for HTML/XML format, and generally follow simpler rules.

The table describes the onSubmit/OnBlur behavior in HTML format. For detailed information on the OnServer validation algorithms, see "Data validation types" on page 558 in "Data validation types" on page 558 in the *ColdFusion Developer's Guide*.

For more information on validation, including discussions of the advantages and disadvantages of different validation types, see "Validating Data" on page 554 in the *ColdFusion Developer's Guide*.

Masking input data

In HTML and Flash forms, the `mask` attribute controls the format of data that can be entered into a text field or that is selected in a datefield input control calendar. In HTML format, it does *not* prevent users from typing a date that does not follow the mask into a datefield input control. You can combine masking and validation on a field.

In text fields, ColdFusion automatically inserts any literal mask characters, such as hyphen (-) characters in telephone numbers. Users type only the variable part of the field.

The following pattern enforces entry of a part number of the format EB-1234-c1-098765, where the user starts the entry by typing the first numeric character, such as 3. ColdFusion fills in the preceding EB prefix and all - characters. The user must enters four numbers, followed by two alphanumeric characters, followed by six numbers.

```
<cfinput type="text" name="newPart" mask="EB-9999-XX-999999"/>
```

Note: *To force a pattern to be all-uppercase or all-lowercase, use the ColdFusion* `UCase` *or* `LCase` *functions in the action page.*

For tags with `type="datefield"` (and `cfcalendar` tags), the number of pattern characters determines the format of the output when the user selects a date in the calendar, as follows:

Mask	Pattern
D	Single- or double-digit day of month, such as 1 or 28
DD	Double-digit day of month, such as 01 or 28
M	Single- or double-digit month, such as 1 or 12
MM	Double-digit month, such as 01 or 12
MMM	Abbreviated month name, such as Jan or Dec
MMMM	Full month name, such as January or December
YY	Two-character year, such as 05
YYYY	Four-character year, such as 2005

Mask	Pattern
E	Single-digit day of week, such as 1 or 7
EEE	Abbreviated day of week name, such as Mon or Sun
EEEE	Full month day of week name, such as Monday or Sunday

The following pattern specifies that the Flash forms sends the date selected by using a `datefield` input control to ColdFusion as text in the format 04/29/2004:

```
<cfinput name="stDate" type="datefield" label="date:" mask="mm/dd/yyyy"/>
```

Flash form data binding

The `bind` attribute lets you populate form fields by using the contents of other form fields. To specify text from another field in a Flash format `cfinput` tag `bind` attribute, use the following format:

{*sourceTagName*.text}

For example, the following line uses the values from the firstName and lastName fields to construct an e-mail address. (The user can change or replace this value with a typed entry.)

```
<cfinput type="text" name="email" label="email"
    bind="{firstName.text}.{lastName.text}@mm.com">
```

HTML form data binding

The `bind` attribute lets you set `cfinput` attributes dynamically. For example, you can automatically fill an email field text-input value based on name and domain field values.

In HTML format, the `bind` attribute specifies a *bind expression*, which can have any for the following forms:

- A *Bind parameter* or string that contains one or more bind parameters. A bind parameter specifies a form control value or other attribute and, optionally, an event. In its most basic form, a bind parameter consists of the `name` or `id` attribute of the control to which you are binding in braces ({ }) The value of the control attributes specified in the bind parameters determine the value of the `cfinput` control attribute.
- A CFC or JavaScript function, or URL, typically using one or more bind parameters as function parameters. The data returned by the function or URL sets the `cfinput` attribute value.

For details of using HTML form data binding, see "Binding data to form fields" on page 650 in the *ColdFusion Developer's Guide*.

Note: *To bind to a `cfinput` control with `type` attribute of `button`, you must specify a bind event setting such as `click` in the bind expression of the control that binds to the button. The default event, `onChange`, has no effect.*

Example: without binding

```
<!--- This example shows the use of cfinput within a cfform to ensure simple
    validation of text items. --->
<cfform action = "cfinput.cfm">
<!--- Phone number validation. --->
Phone Number Validation (enter a properly formatted phone number): <br>
<cfinput
    type = "Text" name = "MyPhone"
    message = "Enter telephone number, formatted xxx-xxx-xxxx (e.g. 617-761-2000)"
    validate = "telephone" required = "yes">
    <font size = -1 color = red>Required</font>
<!--- Zip code validation. --->
<p>Zip Code Validation (enter a properly formatted zip code):<br>
<cfinput
    type = "Text" name = "MyZip"
```

```
    message = "Enter zip code, formatted xxxxx or xxxxx-xxxx"
    validate = "zipcode" required = "yes">
    <font size = -1 color = red>Required</font>
<!--- Range validation. --->
<p>Range Validation (enter an integer from 1 to 5): <br>
<cfinput
    type = "Text" name = "MyRange" range = "1,5"
    message = "You must enter an integer from 1 to 5"
    validate = "integer" required = "no">
<!--- Date validation. --->
<p>Date Validation (enter a properly formatted date):<br>
<cfinput
    type = "Text" name = "MyDate"
    message = "Enter a correctly formatted date (dd/mm/yy)"
    validate = "date" required = "no">
<input
    type = "Submit" name = ""
    value = "send my information">
</cfform>
```

Example: with binding

The following example uses binding to generate a default e-mail address based on input controls with a first, last and domain names, and fills in the e-mail text-input field with the result.

The CFML page contains the following code:

```
<h3>CFC bind using onkeyup() event</h3>
<ul>
    <li>When you type in the First Name field, the first initial appears in
            the Email control.</li>
    <li>Each key you enter in the Last Name and Domain fields appears in the
            Email control.</li>
</ul>

<cfform name="mycfform">
    First Name<font color="#FF0000">*</font>:   
        <cfinput type="text" name="firstname" required="yes"><br>
    Last Name<font color="#FF0000">*</font>:   
        <cfinput type="text" name="lastname" required="yes"><br>
    Domain<font color="#FF0000">*
        </font>:        
        <cfinput type="text" name="domain" required="yes"><br>
    <br>
        <cfinput type="text" name="email1"
            bind="cfc:bindFcns.getEmailId({firstname@keyup},{lastname@keyup},
                {domain@keyup})">
    <br><br>
<font color="#FF0000">*</font> indicates the field is required.
</cfform>
```

The bindFcns CFC contains the following code:

```
<cfcomponent>
    <cffunction name="getEmailId" access="remote">
        <cfargument name="firstname">
        <cfargument name="lastname">
        <cfargument name="domain">
        <cfreturn "#left(firstname,1)#.#lcase(arguments.lastname)#@#
    lcase(domain)#">
    </cffunction>
</cfcomponent>
```

cfinsert

Description

Inserts records in data sources from data in a ColdFusion form or form Scope.

Category

Database manipulation tags

Syntax

```
<cfinsert
    dataSource = "data source name"
    tableName = "table name"
    formFields = "formfield1, formfield2, ..."
    password = "password"
    tableOwner = "owner"
    tableQualifier = "table qualifier"
    username = "user name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfprocparam`, `cfprocresult`, `cfquery`, `cfqueryparam`, `cfstoredproc`, `cftransaction`, `cfupdate`

History

ColdFusion MX: Deprecated the `connectString`, `dbName`, `dbServer`, `dbtype`, `provider`, and `providerDSN` attributes. They do not work, and might cause an error, in releases later than ColdFusion 5.

Attributes

Attribute	Req/Opt	Default	Description
`dataSource`	Required		Data source; contains table.
`tableName`	Required		Table in which to insert form fields.
			ORACLE drivers: must be uppercase.
			Sybase driver: case-sensitive. Must be the same case used when table was created
`formFields`	Optional	(all on form, except keys)	Comma-delimited list of form fields to insert. If not specified, all fields in the form are included.
			If a form field is not matched by a column name in the database, ColdFusion throws an error.
			The database table key field must be present in the form. It may be hidden.
`password`	Optional		Overrides password specified in ODBC setup.
`tableOwner`	Optional		For data sources that support table ownership (such as SQL Server, Oracle, and Sybase SQL Anywhere), use this field to specify the owner of the table.
`tableQualifier`	Optional		For data sources that support table qualifiers, use this field to specify qualifier for table. The purpose of table qualifiers varies among drivers. For SQL Server and Oracle, qualifier refers to name of database that contains table. For Intersolv dBASE driver, qualifier refers to directory where DBF files are located.
`username`	Optional		Overrides username specified in ODBC setup.

Example

```
<!--- This example shows how to use cfinsert instead of cfquery to put data in a
```

```
    datasource. --->
<!--- If form.POSTED exists, we insert new record, so begin cfinsert tag. --->
<cfif IsDefined ("form.posted")>
    <cfinsert dataSource = "cfdocexamples"
        tableName = "COMMENTS"
        formFields = "Email,FromUser,Subject,MessText,Posted">
    <h3><I>Your record was added to the database.</i></h3>
</cfif>

<cfif IsDefined ("form.posted")>
    <cfif Server.OS.Name IS "Windows NT">
        <cfinsert datasource="cfdocexamples" tablename="COMMENTS"
            formfields="EMail,FromUser,Subject,MessText,Posted">
    <cfelse>
        <cfinsert datasource="cfdocexamples" tablename="COMMENTS"
            formfields="CommentID,EMail,FromUser,Subject,MessText,Posted">
    </cfif>
    <h3><i>Your record was added to the database.</i></h3> </cfif>

<!--- Use a query to show the existing state of the database. --->
<cfquery name = "GetComments" dataSource = "cfdocexamples">
    SELECT
        CommentID, EMail, FromUser, Subject, CommtType, MessText, Posted, Processed
    FROM
        COMMENTS
</cfquery>

<html>
<head></head>
<h3>cfinsert Example</h3>
<p>First, show a list of the comments in the cfdocexamples datasource.
<!--- Show all the comments in the db. --->
<table>
    <tr>
        <td>From User</td><td>Subject</td><td>Comment Type</td>
        <td>Message</td><td>Date Posted</td>
    </tr>
    <cfoutput query = "GetComments">
        <tr>
            <td valign = top><a href = "mailto:#Email#">#FromUser#</A></td>
            <td valign = top>#Subject#</td>
            <td valign = top>#CommtType#</td>
            <td valign = top><font size = "-2">#Left(MessText, 125)#</font></td>
            <td valign = top>#Posted#</td>
        </tr>
    </cfoutput>
</table>
<p>Next, we'll offer the opportunity to enter a comment:
<!--- Make a form for input. --->
<form action = "cfinsert.cfm" method = "post">
    <pre>
    Email:   <input type = "Text" name = "email">
    From:    <input type = "Text" name = "fromUser">
    Subject: <input type = "Text" name = "subject">
    Message: <textarea name = "MessText" COLS = "40" ROWS = "6"></textarea>
    Date Posted:  <cfoutput>#DateFormat(Now())#</cfoutput>
    <!--- Dynamically determine today's date. --->
    <input type = "hidden"
        name = "posted" value = "<cfoutput>#Now()#</cfoutput>">
    </pre>
    <input type = "Submit"
```

```
        name = "" value = "insert my comment">
</form>
```

cfinterface

Description

Defines an interface that consists of a set of signatures for functions. The interface does not include the full function definitions; instead, you implement the functions in a ColdFusion component (CFC). The interfaces that you define by using this tag can make up the structure of a reusable application framework.

History

ColdFusion 8: Added this tag.

Category

Application framework tags, Extensibility tags

Syntax

```
<cfinterface
    displayName = "descriptive name"
    extends = "interfaceName1[,interfaceName2]..."
    Hint = "hint text">
        <cffunction ...>
            <cfargument ... >
            <cfargument ... >
            ...
        </cffunction>
        <cffunction ...>
            ...
        </cffunction>
        ...
</cfinterface>
```

See also

cfargument, cfcomponent, cffunction, GetComponentMetaData, IsInstanceOf

Attributes

Attribute	Req/Opt	Default	Description
displayName	Optional		A value to be displayed when using introspection to show a descriptive name for the interface.
extends	Optional		A comma-delimited list of one or more interfaces that this interface extends. Any CFC that implements an interface must also implement all the functions in the interfaces specified by this property. If an interface extends another interface, and the child interface specifies a function with the same name as one in the parent interface, both functions must have the same attributes; otherwise ColdFusion generates an error.
hint	Optional		Text to be displayed when using introspection to show information about the interface. The hint attribute value follows the syntax line in the function description.

Usage

The cfinterface tag declares a set of related functions that any ColdFusion component (CFC) that implements the interface must define. The interface specifies function signatures, but does not implement the functions; instead, the CFC that implements the interface must contain the full function definitions.

For example, you could create a create, read, update, and delete (CRUD) interface that defines the basic signatures of the four operations. All components that implement the interface must then conform to the interface signatures. You can then implement the interface in different components to manage different types of data sources. Because all the components implement the same interface, you can ensure that you can easily replace one component with another, depending on the specific data source that an individual application requires.

You define an interface by creating a ColdFusion file with a .cfc extension and specifying the `cfinterface` tag as the first and only top-level tag in the file. The filename determines the interface name, so myInterface.cfc defines the myInterface interface. You can specify any attributes in the `cfinterface` tag; however, only the names listed in the Attributes table are meaningful to ColdFusion. The filename must not contain commas, or any periods except for the separator before the .cfc extension.

Inside the `cfinterface` tag body, you specify the interface by declaring the functions of the interface. The interface definition must follow these basic rules:

- The `cfinterface` tag body can contain only `cffunction` tags and comments.
- The `cffunction` tag bodies can contain only `cfargument` tags, which declare the function arguments, and comments.
- The `cffunction` tag body is optional.

The following example shows the general format of an interface definition:

```
<cfinterface extends="IBasicInterface">
    <cffunction name="hello" description="Should print a greeting containing the input
         argument or 'world'.">
        <cfargument name="whom" type="string" default="world">
    </cffunction>
    <cffunction name="calculateTwo" returnType="numeric" output="no"
            description="calculates a result using two numbers and returns the result">
        <cfargument name="first" type="numeric" required="yes"/>
        <cfargument name="second" type="numeric" required="no" default="0"/>
    </cffunction>
    <cffunction name="disclaimer"/>
</cfinterface>
```

This interface extends the IBasicInterface interface, so any component that implements this interface must also implement the methods of the IBasicInterface interface. This interface requires the component to implement the following three functions:

- A `hello` function that can optionally take a single string argument, which has a default value of `"world"`.
- A `calculateTwo` function that takes one required numeric argument, has an optional numeric argument with a default value of 0, and must return a number.
- A `disclaimer` function that takes no arguments and returns any type.

The CFC that implements an interface specifies the interface name in the `cfcomponent` tag's `implements` attribute. It must implement all of the interface's methods as specified in the interface `cffunction` tags. The order of function arguments in the interface definition and the component definition must be identical.

The following table lists the attributes that you can use in the `cffunction` and `cfargument` tags, and describes the requirements and limitations on how you can use them in the interface definition and the component implementation:

Attribute	Interface requirements	Implementation requirements
cffunction		
access	Optional; only public is allowed	Optional; can be public or remote.
description	Optional	Can differ from value in interface.
displayName	Optional	Can differ from value in interface.
hint	Optional	Can differ from value in interface.
name	Required	Must be identical to value in interface.
output	Optional	Must be identical to value in interface. If you omit this attribute in the interface, you must omit it in the implementation.
returnType	Optional	Must be identical to value in interface; however, an omitted type option and an option value of any are equivalent and ColdFusion treats them as a match.
roles	Not allowed	Can be any valid value.
cfargument		
default	Optional	Must be identical to value in interface. If you omit this attribute in the interface, you can specify any value in the implementation.
displayName	Optional	Can differ from value in interface
hint	Optional	Can differ from value in interface
name	Required	Must be identical to value in interface.
required	Optional	If the interface specifies yes, this attribute must also specify yes. If the interface specifies no or omits this attribute, you can specify no or omit the attribute.
type	Optional	Must be identical to value in interface; however, an omitted type option and an option value of any are equivalent and ColdFusion treats them as a match.

A CFC can implement multiple interfaces.

Note: *If a CFC implements multiple interfaces and two or more of the interfaces define functions with identical names, the signatures of these functions must be the same in all the interfaces; ColdFusion does not support function overloading.*

ColdFusion uses the same rules to locate interfaces as it does to locate components. You can use the GetComponentMetaData function to get information about an interface.

Adobe recommends that you use a consistent technique for identifying interface names, for example, by always starting the file (and therefore interface) name with a capital I. Any component that implements only that single interface could have a similar name, for example the same root prefixed by a capital C. You could have an IresourceInfo.cfc interface file and a corresponding CresourceInfo.cfc component file, for example.

Example

The following example defines an IBasicMath interface with add, subtract, multiply, and divide operations. The integerMath CFC implements this interface by defining integer arithmetic versions of the operations. The testMath.cfm application uses the integerMath functions to do arithmetic calculations on two decimal numbers (using the values of pi and e).

As an exercise, consider modifying the interface definition to take and return values of any type, and then implement a second CFC that uses the `PrecisionEvaluate` function to calculate arbitrary precision arithmetic and return the results. (We have omitted these versions for brevity.)

The IBasicMath.cfc file defines the interface as follows:

```
<cfinterface>
    <cffunction name = add returntype = "numeric" output = "no"
            description = "Add two values">
        <cfargument name = "first" type="numeric" required = "no" default ="0">
        <cfargument name = "second" type = "numeric" required = "no" default = "0">
    </cffunction>
    <cffunction name = subtract returntype = "numeric" output = "no"
            description = "Subtract two values">
        <cfargument name = "first" type = "numeric" required = "no" default = "0">
        <cfargument name ="second" type = "numeric" required = "no" default = "0">
    </cffunction>
    <cffunction name = multiply returntype = "numeric" outpu = "no"
            description = "Multiply two values">
        <cfargument name = "first" type = "numeric" required = "no" default = "0">
        <cfargument name = "second" type = "numeric" required = "no" default = "0">
    </cffunction>
    <cffunction name = divide returntype = "numeric" output = "no"
            description = "Divide two values">
        <cfargument name = "first" type = "numeric" required = "no" default="0">
        <cfargument name = "second" type="numeric" required = "no" default="1">
    </cffunction>
</cfinterface>
```

The integerMath.cfc file defines the integerMath component, which implements the IBasicMath interface, as follows:

```
<cfcomponent implements = "IBasicMath" >
    <cffunction name = add returntype = "numeric" output = "no"
            description = "Add two values">
        <cfargument name = "first" type = "numeric" required = "no" default = "0">
        <cfargument name = "second" type = "numeric" required = "no" default = "0">
        <cfreturn Round(first + second)>
    </cffunction>
    <cffunction name = subtract returntype = "numeric" output = "no"
            description = "Subtract two values">
        <cfargument name = "first" type = "numeric" required = "no" default = "0">
        <cfargument name = "second" type = "numeric" required = "no" default = "0">
        <cfreturn Round(first - second)>
    </cffunction>
    <cffunction name = multiply returntype = "numeric" output = "no"
            description = "Multiply two values">
        <cfargument name = "first" type = "numeric" required = "no" default = "0">
        <cfargument name = "second" type = "numeric" required = "no" default = "0">
        <cfreturn Round(first * second)>
    </cffunction>
    <cffunction name = divide returntype = "numeric" output = "no"
description = "Divide two values">
        <cfargument name = "first" type = "numeric" required = "no" default = "0">
        <cfargument name = "second" type = "numeric" required = "no" default = "1">
```

```
        <cfreturn Round(first / second)>
    </cffunction>
</cfcomponent>
```

The testMath.cfm file uses the integerMath component methods to calculate integer values, as follows:

```
<cfscript>
    arguments = StructNew();
    arguments.first = pi();
    arguments.second = "2.71828182845904523536028747135266";
</cfscript>

<cfobject name = "iMathObj" component = "integerMath">
<cfoutput>
<h3>Function Arguments</h3>
argument 1: #arguments.first#<br>
argument 2: #arguments.second#<br>

<h3>Addition</h3>
#iMathObj.add(argumentCollection = arguments)#
<h3>Subtraction</h3>
#iMathObj.subtract(argumentCollection = arguments)#
<h3>Multiplication</h3>
#iMathObj.multiply(argumentCollection = arguments)#
<h3>Division</h3>
#iMathObj.divide(argumentCollection = arguments)#
</cfoutput>
```

cfinvoke

Description

Does either of the following:

- Invokes a component method from within a ColdFusion page or component.
- Invokes a web service.

This tag works as follows:

- Transiently instantiates a component or web service and invokes a method on it.
- Invokes a method on an instantiated component or web service.

This tag can pass parameters to a method in the following ways:

- With the `cfinvokeargument` tag
- As named attribute-value pairs, one attribute per parameter
- As a structure, in the `argumentCollection` attribute

Category

Extensibility tags

Syntax

```
<!--- Syntax 1: This syntax invokes a method of a component. --->
<cfinvoke
    component = "component name or reference"
    method = "method name"
    returnVariable = "variable name"
    argumentCollection = "argument collection"
    ...>

OR

<!--- Syntax 2: This syntax can invoke a method of a component only from within the component.
--->
<cfinvoke
    method = "method name"
    returnVariable = "variable name"
    argumentCollection = "argument collection"
    ...>

OR

<!--- Syntax 3: This syntax invokes a web service. --->
<cfinvoke
    webservice = "Web service name or WSDL URL"
    method = "operation name"
    password = "password"
    proxyPassword = "password for proxy server"
    proxyPort = "port on proxy server"
    proxyServer = "WSDL proxy server URL"
    proxyUser = "user ID for proxy server"
    returnVariable = "variable name"
    refreshWSDL = "yes|no"
    servicePort = "WSDL port name"
    timeout = "request timeout in seconds"
    username = "user name"
```

```
    wsdl2javaArgs = "argument string">

OR

<!--- Syntax 4A: This syntax invokes a component.
    This syntax shows instantiation with the cfobject tag.
    This cfinvoke syntax applies to instantiating a component
    with the cfobject tag and to instantiating a component
    with the CreateObject function. --->
<cfobject
    component = "component name"
    name = "name for instantiated component">
    <cfinvoke
    <!--- Value is object name, within number signs. --->
    component = "#name of instantiated component#"
    method = "method name"
    returnVariable = "variable name"
    argumentCollection = "argument collection"
    ...>

OR

<!--- Syntax 4B: This syntax invokes a web service.
    This syntax shows instantiation with the cfobject tag.
    This cfinvoke syntax applies to instantiating a web service with the cfobject tag and to
    instantiating a web service with the CreateObject function. --->
<cfobject
    webservice = "web service name or WSDL URL"
    name = "name for instantiated object"
    (optional cfobject attibutes)>
<cfinvoke
    webservice = "#cfobject name attribute value#"
    method = "method name"
    password = "password"
    proxyPassword = "password for proxy server"
    proxyPort = "port on proxy server"
    proxyServer = "name or IP address of WSDL proxy server"
    proxyUser = "user ID for proxy server"
    returnVariable = "variable name"
    refreshWSDL = "yes|no"
    servicePort = "WSDL port name"
    timeout = "request time-out in seconds"
    username = "user name"
    wsdl2javaArgs = "argument string">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfargument`, `cfcomponent`, `cffunction`, `cfinvokeargument`, `cfobject`, `cfproperty`, `cfreturn`

History

ColdFusion 8: Added the following attributes: `refreshWSDL`, `wsdl2javaArgs` attributes.

ColdFusion MX 7: Added the `servicePort` attribute.

ColdFusion MX 6.1: Added the following attributes: `timeout`, `proxyServer`, `proxyPort`, `proxyUser`, and `proxyPassword`.

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`argumentCollection`	Optional		Name of a structure; associative array of arguments to pass to the method.
`component`	See Usage.		String or component object; a reference to a component, or component to instantiate.
`input_params ...`			Input parameters. For each named input parameter specify *paramName=paramValue*.
`method`	See Usage.		Name of a method. For a web service, the name of an operation.
`password`	Optional	Password set in the Administrator, if any	The password to use to access the web service. If the `webservice` attribute specifies a web service name configured in the Administrator, overrides any user name specified in the Administrator entry.
`proxyPassword`	Optional	http.proxyPassword system property, if any	The user's password on the proxy server.
`proxyPort`	Optional	http.proxyPort system property, if any.	The port to use on the proxy server.
`proxyServer`	Optional	http.proxyHost system property, if any.	The proxy server required to access the webservice URL.
`proxyUser`	Optional	http.proxyUser system property, if any	The user ID to send to the proxy server.
`refreshWSDL`	Optional	`no`	• `yes`: reload the WSDL file and regenerate the artifacts used to consume the web service • `no`
`returnVariable`	Optional		Name of a variable for the invocation result.
`servicePort`	Optional	First port found in the WSDL	The port name for the web service. This value is case-sensitive and corresponds to the `port` element's `name` attribute under the `service` element. Specify this attribute if the web service contains multiple ports.
`timeout`	Optional	0 (no timeout)	The time-out for the web service request, in seconds.
`username`	Optional	User name set in the Administrator, if any	The user name to use to access the web service. If the `webservice` attribute specifies a web service name configured in the Administrator, overrides any user name specified in the Administrator entry.

Attribute	Req/Opt	Default	Description
webservice	See Usage		One of the following: • The absolute URL of the web service WSDL. • The Name (string) assigned in the ColdFusion Administrator to the web service.
wsdl2javaArgs	See Usage		A string that contains a space-delimited list of arguments to pass to the WSDL2Java tool that generates Java stubs for the web services. Useful arguments include the following: • -W or --noWrapped: Turns off the special treatment of wrapped document/literal style operations. • -a or --all: Generates code for all elements in the WSDL, even unreferenced ones. • -w or --wrapArrays: Prefers building beans to straight arrays for wrapped XML array types. This switch is not in included in the Axis documentation. For detailed information on valid arguments, see the Apache Axis WSDL2Java Reference.

Note: *If you do not specify any the attributes of the proxy server, and a corresponding system property is set (typically in the JVM startup arguments) ColdFusion uses the system property value.*

Usage

The following table shows when you can use each attribute:

This attribute is required, optional, ignored, or invalid:	For this cfinvoke tag syntax:				
	Syntax 1	Syntax 2	Syntax 3	Syntax 4A	Syntax 4B
argumentCollection	Optional	Optional	Optional	Optional	Optional
component	Required	Optional	Invalid	Required	Invalid
input_params ...	Optional	Optional	Optional	Optional	Optional
method	Required	Required	Required	Required	Required
password	Ignored	Ignored	Optional	Ignored	Optional
proxyPassword	Invalid	Invalid	Optional	Invalid	Optional
proxyPort	Invalid	Invalid	Optional	Invalid	Optional
proxyServer	Invalid	Invalid	Optional	Invalid	Optional
proxyUser	Invalid	Invalid	Optional	Invalid	Optional
returnVariable	Optional	Optional	Optional	Optional	Optional
servicePort	Invalid	Invalid	Optional	Invalid	Optional
timeout	Invalid	Invalid	Optional	Invalid	Optional
username	Ignored	Ignored	Optional	Ignored	Optional
webservice	Invalid	Invalid	Required	Invalid	Required
wsdl2javaArgs	Invalid	Invalid	Optional	Invalid	Optional

If the component attribute specifies a component name, the component with the corresponding name is instantiated, the requested method is invoked, and then the component instance is immediately destroyed. If the attribute contains a reference to an instantiated component object, no instantiation or destruction of the component occurs.

On UNIX systems, ColdFusion searches first for a file with a name that matches the specified component name, but is all lower case. If it does not find the file, it looks for a file name that matches the component name exactly, with the identical character casing.

Method arguments can be passed in any of the following ways. If an argument is passed in more than one way with the same name, this order of precedence applies:

1 Using the cfinvokeargument tag

2 Passing directly as attributes of the cfinvoke tag (they cannot have the same name as a registered cfinvoke attribute: method, component, webservice, returnVariable)

3 Passing as struct keys, using the argumentCollection attribute

For example, the params struct contains three keys: a=1, b=1, c=1. The following call is evaluated as if the arguments were passed to the method in the order a=3, b=2, c=1:

```
<cfinvoke ... a=2 b=2 argumentCollection=params>
        <cfinvokeargument name="a" value="3">
    </cfinvoke>
```

Note: *The following cfinvoke tag attribute names are reserved; they cannot be used for argument names:* `component`, `method`, `argumentCollection`, *and* `result`.

Example1

This example uses Syntax 1.

```
<!--- Immediate instantiation and destruction. --->
<cfinvoke
    component="nasdaq.quote"
    method="getLastTradePrice"
    returnVariable="res">
    <cfinvokeargument
        name="symbol"
        value="macr">
</cfinvoke>
<cfoutput>#res#</cfoutput>
```

Example2

This example uses Syntax 1.

```
<!--- Passing the arguments using argumentCollection. --->
<cfset args = StructNew()>
<cfset args.symbol = "macr">
<cfinvoke
    component="nasdaq.quote"
    method="getLastTradePrice"
    argumentCollection="#args#"
    returnVariable="res">
<cfoutput>#res#</cfoutput>
```

Example3

This example uses Syntax 2.

```
<!--- Called only from within a component, MyComponent. --->
<cfinvoke
```

```
    method = "a method name of MyComponent"
    returnVariable = "variable name">
```

Example4

This example uses Syntax 3.

```
<!--- Using cfinvoke to consume a web service using a ColdFusion component. --->
<cfinvoke
    webservice="http://www.xmethods.net/sd/2001/TemperatureService.wsdl"
    method="getTemp"
    returnvariable="aTemp">
<cfinvokeargument name="zipcode" value="55987"/>
</cfinvoke>
<cfoutput>The temperature at zip code 55987 is #aTemp#</cfoutput>
```

For more information on web services, see “Using Web Services” on page 902 in the *ColdFusion Developer’s Guide*.

Example5

This example uses Syntax 4A.

```
<!--- Separate instantiation and method invocation; useful for multiple invocations using
    different methods or values. --->
<cfobject
    name="quoteService"
    component="nasdaq.quote">
<cfinvoke
    component="#quoteService#"
    method="getLastTradePrice"
    symbol="macr"
    returnVariable="res_macr">
<cfoutput>#res#</cfoutput>
<cfinvoke
    component="#quoteService#"
    method="getLastTradePrice"
    symbol="mot"
    returnVariable="res_mot">
<cfoutput>#res#</cfoutput>
```

cfinvokeargument

Description

Passes the name and value of a parameter to a component method or a web service. This tag is used in the `cfinvoke` tag.

Category

Extensibility tags

Syntax

```
<cfinvokeargument
    name="argument name"
    value="argument value"
    omit = "yes|no">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfargument`, `cfcomponent`, `cffunction`, `cfinvoke`, `cfobject`, `cfproperty`, `cfreturn`

History

ColdFusion MX 7: Added the `omit` attribute.

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		Argument name.
`value`	Required		Argument value.
`omit`	Optional	`no`	Enables you to omit a parameter when invoking a web service. It is an error to specify `omit="yes"` if the `cfinvoke webservice` attribute is not specified. `yes`: omit this parameter when invoking a web service. `no`: do not omit this parameter when invoking a web service.

Usage

You can have multiple `cfinvokeargument` tags in a `cfinvoke` tag body.

You can use `cfinvokeargument` tag to dynamically determine the arguments to be passed. For example, you can use conditional processing to determine the argument name, or you can use a `cfif` tag to determine whether to execute the `cfinvokeargument` tag.

If you are invoking a web service, you can omit a parameter by setting the `omit` attribute to `"yes"`. If the WSDL specifies that the argument is nillable, ColdFusion MX sets the associated argument to null. If the WSDL specifies minoccurs=0, ColdFusion MX omits the argument from the WSDL.

Example1

```
<cfinvoke
    component="nasdaq.quote"
    method="getLastTradePrice"
    returnVariable="res">
    <cfinvokeargument name="symbol" value="mot">
```

```
    <cfinvokeargument name="symbol" value="macr">
</cfinvoke>

<cfoutput>#res#</cfoutput>
```

Example2

```
<!--- Using cfinvoke to consume a web service using a ColdFusion component. --->
<cfinvoke
    webservice="http://www.xmethods.net/sd/2001/TemperatureService.wsdl"
    method="getTemp"
    returnvariable="aTemp">
        <cfinvokeargument name="zipcode" value="55987"/>
</cfinvoke>
<cfoutput>The temperature at zip code 55987 is #aTemp#</cfoutput>
```

cflayout

Description

Creates a region of its container (such as the browser window or a `cflayoutarea` tag) with a specific layout behavior: a bordered area, a horizontal or vertically arranged box, or a tabbed navigator.

Category

Display management tags

Syntax

```
<cflayout
    type="border|hbox|tab|vbox"
    align="center|justify|left|right
    name="string"
    padding="integer"
    style="CSS style specification"
    tabHeight="measurement">
    tabPosition="top|bottom">

        cflayoutarea tags

</cflayout>
```

Note: *You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.*

See also

`cfajaximport`, `cfdiv`, `cflayoutarea`, `cfpod`, `cfwindow`, "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Applies to	Description
type	Required		all	The type of layout. The following values are valid: • border: a box with a border and up to five layout areas, each with a border. For more information, see Usage. • hbox: a horizontal box where all immediate child cflayoutarea controls are arranged horizontally. • tab: a tabbed display where the current child cflayoutarea tag occupies the display area of the layout, and each layout area has a tab that the user can select to display its contents. • vbox: a vertical box where all immediate child cflayoutarea controls are arranged vertically.
align	Optional	Determined by browser layout direction	all	Specifies the default alignment of the content of child layout areas. Each cflayoutarea tag can specify an alignment attribute to override this value. The following values are valid: • center • justify • left • right
name	Optional		all	The name of the layout region. Should be unique on a page.
padding	Optional	0	hbox, vbox	• For hbox layouts, specifies the padding on the right side of each child layout area. • For vbox layouts, specifies the padding at the bottom of each child layout area. You can use any valid CSS length or percent format, such as 10, 10% 10px, or 10em, for this attribute. The padding is included in the child layout area and takes the style of the layout area.
style	Optional		all	A CSS style specification that defines layout styles.
tabHeight	Optional		tab	Specifies the height of the content area of all child layout areas. You can override this setting by specifying a height setting in a individual cflayoutarea tag style attributes.
tabPosition	Optional	top	tab	Specifies the location of the tabs relative to the tab region contents. • bottom: the tabs appear at the bottom of the layout. • top: the tabs appear at the top of the layout.

Usage

The immediate children of a cflayout tag must be cflayoutarea tags or nondisplay tags whose bodies contain one or more cflayoutarea tags at the top level. For example, a cflayout tag could have a tag such as cfloop or cfquery as a child, and these tags would have cflayoutarea tags in their bodies.

The border type layout has the following characteristics:

- The layout control and each of its immediate layout area children is surrounded by a border.
- The control can have up to five children positioned at the left, right, center, top, and bottom of the layout.

- You can configure the child layout areas, except for the center area, to have splitters so that users can expand and collapse them or close them completely.
- The center child layout area occupies all available space in the layout that is not used by any of the other layout areas.
- To specify layout height, use the `height` setting of the `style` attribute.

Note: *If you specify a border layout on a page that has a* `DOCTYPE` *declaration, the layout cannot properly determine its height and you must specify the height in a* `cflayout` *tag* `style` *attribute.*

You can use the following JavaScript functions to access the underlying Ext JS - JavaScript Library objects for border and tab type `cflayout` controls.

Function	Description
ColdFusion.Layout.getBorderLayout	Gets the underlying Ext JS - JavaScript Library object for the specified border type `cflayout` control.
ColdFusion.Layout.getTabLayout	Gets the underlying Ext JS - JavaScript Library object for the specified tab type `cflayout` control.

For more information on configuring layout areas, see `cflayoutarea`.

Example

The following example shows a set of nested layouts. The outer layout is a vbox, with two layout areas. The top layout area has a border layout, the bottom layout area contains a form with buttons to control the display of the border layout areas.

```
<html>
<head>
</head>
<body>
<cflayout name="outerlayout" type="vbox">
    <cflayoutarea style="height:400;">
        <cflayout name="thelayout" type="border">
            <!--- The 100% height style ensures that the background color fills
                the area. --->
            <cflayoutarea position="top" size="100" splitter="true"
                    style="background-color:##00FFFF; height:100%">
                This is text in layout area 1: top
            </cflayoutarea>
            <cflayoutarea title="Left layout area" position="left"
                    closable="true"
                    collapsible="true" name="left" splitter="true"
                    style="background-color:##FF00FF; height:100%">
                This is text in layout area 2: left<br />
                You can close and collapse this area.
            </cflayoutarea>
            <cflayoutarea position="center"
                    style="background-color:##FFFF00; height:100%">
                This is text in layout area 3: center<br />
            </cflayoutarea>
            <cflayoutarea position="right" collapsible="true"
                    title="Right Layout Area" initcollapsed="true"
                    style="background-color:##FF00FF; height:100%" >
                This is text in layout area 4: right<br />
                You can collapse this, but not close it.<br />
                It is initially collapsed.
            </cflayoutarea>
```

```
            <cflayoutarea position="bottom" size="100" splitter="true"
                     style="background-color:##00FFFF; height:100%">
                This is text in layout area 5: bottom
            </cflayoutarea>
        </cflayout>
    </cflayoutarea>

    <cflayoutarea style="height:100; ; background-color:##FFCCFF">
        <h3>Change the state of Area 2</h3>
        <cfform>
            <cfinput name="expand2" width="100" value="Expand Area 2" type="button"
                onClick="ColdFusion.Layout.expandArea('thelayout', 'left');">
            <cfinput name="collapse2" width="100" value="Collapse Area 2" type="button"
                onClick="ColdFusion.Layout.collapseArea('thelayout', 'left');">
            <cfinput name="show2" width="100" value="Show Area 2" type="button"
                onClick="ColdFusion.Layout.showArea('thelayout', 'left');">
            <cfinput name="hide2" width="100" value="Hide Area 2" type="button"
                onClick="ColdFusion.Layout.hideArea('thelayout', 'left');">
        </cfform>
    </cflayoutarea>
</cflayout>
</body>
</html>
```

cflayoutarea

Description

Defines a region within a `cflayout` tag body, such as an individual tab of a tabbed layout.

Category

Display management tags

Syntax

```
In a border layout
<cflayoutarea
    required
    position="bottom|center|left|right|top"
    optional
    align="left|center|justify|right"
    closable="false|true"
    collapsible="false|true"
    initcollapsed="false|true"
    inithide="false|true"
    maxSize="number of pixels"
    minSize="number of pixels"
    name="string"
    onBindError = "JavaScript function name"
    overflow = "auto|hidden|scroll|visble"
    size="number of pixels"
    source="URL"
    splitter="false|true"
    style="CSS style specification"
    title="string">

    area elements

</cflayoutarea>

In a hbox or vbox layout
<cflayoutarea
    optional
    name="string"
    onBindError = "JavaScript function name"
    overflow = "auto|hidden|scroll|visble"
    size="number of pixels"
    source="URL"
    style="CSS style specification">

    area elements

</layoutarea>

In a tab layout
<cflayoutarea
    optional
    closable="false|true"
    disabled="false|true"
    inithide="false|true"
    name="string"
    onBindError = "JavaScript function name"
    overflow = "auto|hidden|scroll|visble"
    refreshOnActivate = "false|true"
```

```
    selected="false|true"
    source="URL"
    style="CSS style specification"
    title="string">

    area elements

</layoutarea>
```

If you specify a `source` attribute, all child tags are ignored. If you do not have child tags, close the tag with />.

Note: *You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.*

See also

`cfdiv`, `cflayout`, `cfpod`, `cfwindow`, "AJAX JavaScript Functions" on page 1246, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag

Attributes

Attribute	Req/Opt	Default	Applies to	Description
`align`	Optional	The `cflayout` tag `align` attribute value	all	Specifies how to align child controls within the layout area. The following values are valid: • `center` • `justify` • `left` • `right`
`closable`	Optional	false	border, tab	A Boolean value that specifies whether the area can close. Specifying this attribute adds an x icon on the tab or title bar that a user can click to close the area. You cannot use this attribute for border layout areas with a `position` attribute value of `center`.
`collapsible`	Optional	false	border	A Boolean value that specifies whether the area can collapse. Specifying this attribute adds a >> or << icon on the title bar that a user can click to collapse the area. You cannot use this attribute for border layout areas with a `position` attribute value of `center`.
`position`	Required if the `cflayout` type is `border`		border	The position of the area in the layout. Must be one of the following values: • `top`: Position the area across the top of the full layout. • `bottom`: Position the area across the bottom of the full layout. • `left`: Position the area on the left side of the layout, between any visible top and bottom areas. • `right`: Position the area on the right side of the layout, between any visible top and bottom areas. • `center`: Position the area in the space not taken by the top, bottom, left, and right areas. Border style layouts can have at most one layout area of each type.

Attribute	Req/Opt	Default	Applies to	Description
disabled	Optional	false	tab	A Boolean value that specifies whether the tab is disabled, that is, whether a user can select the tab to display its contents. Disabled tabs are greyed out. Ignored if the selected attribute value is true.
initCollapsed	Optional	false	border	A Boolean value that specifies whether the area is initially collapsed. You cannot use this attribute for border layout areas with a position attribute value of center. Ignored if the collapsible attribute value is false.
initHide	Optional	false	border, tab	A Boolean value that specifies whether the area is initially hidden. To show an initially hidden area, use the ColdFusion.Layout.showArea or ColdFusion.Layout.showTab function. You cannot use this attribute for border layout areas with a position attribute value of center.
maxSize	Optional	-1 (no maximum size)	border	If the position attribute value is top or bottom, the maximum height of the area, in pixels, that you can set by dragging a splitter. If the position attribute value is left or right, the maximum width of the area. You cannot use this attribute for border layout areas with a position attribute value of center.
minSize	Optional	-1 (no maximum size)	border	If the position attribute value is top or bottom, the minimum height of the area, in pixels, that you can set by dragging a splitter. If the position attribute value is left or right, the minimum width of the area. You cannot use this attribute for border layout areas with a position attribute value of center.
name	Optional		all	The name of the layout area.
onBindError	Optional	See Description	all	The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and have specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up displays.

Attribute	Req/Opt	Default	Applies to	Description
overflow	Optional	auto	all	Specifies how to display child content whose size would cause the control to overflow the window boundaries. The following values are valid: • auto: show scroll bars when necessary. • hidden: do not allow access to overflowing content. • scroll: always show horizontal and vertical scroll bars, even if they are not needed. • visible: content can display outside the bounds of the layout area. **Notes:** • You cannot use visible or scroll for layout areas in border layouts. • In Internet Explorer, layout areas with the visible setting expand to fit the size of the contents, rather than having the contents extend beyond the layout area.
refreshOnActivate	Optional	false	tab	• true: Refresh the contents of the tab by running the source bind expression whenever the tab display region shows (for example, when the user selects the tab), in addition to when bind events occur. • false: Refresh the tab display region only when the bind expression is triggered by its bind event. To use this attribute, you must also specify a source attribute.
selected	Optional	first tab is selected	tab	A Boolean value that specifies whether this tab is initially selected so that its contents appears in the layout.
size	Optional	-1 calculate initial size dynamically	border, hbox, vbox	For hbox layouts and border layouts with position attribute values of top or bottom, the initial height of the area. For vbox layouts and border layouts with position attribute values of left or right, the initial width of the area. For hbox and vbox layouts, you can use any valid CSS length or percent format (such as 10, 10% 10px, or 10em) for this attribute. For border layouts, this attribute value must be an integer number of pixels. You cannot use this attribute for border layout areas with a position attribute value of center. ColdFusion automatically determines the center size based on the size of all other layout areas. **Note:** If a layout area in a border layout contains only AJAX controls such as HTML format cftree tags, you must specify a size attribute. Otherwise, the AJAX components may not be visible until the layout area is resized.
source	Optional		all	A URL that returns the layout area contents. ColdFusion uses standard page path resolution rules. You can use a bind expression with dependencies in this attribute. If a file specified in this attribute includes tags that use AJAX features, such as cfform, cfgrid, and cfpod, you must use the cfajaximport tag on the page that includes the cflayoutarea tag. For more information, see cfajaximport. For more information on the source attribute, see Usage.

Attribute	Req/Opt	Default	Applies to	Description
splitter	Optional	false	border	A Boolean value that specifies whether the layout area has a divider between it and the adjacent layoutarea control. Users can drag the splitter to change the relative sizes of the areas. If this attribute is set to true on a left or right position layout area, the splitter resizes the area and its adjacent area horizontally. If this attribute is set to true on a top or bottom position layout area, the splitter resizes the layout vertically. You cannot use this attribute for border layout areas with a position attribute value of center.
style	Optional		all	A CSS style specification that controls the appearance of the area.
title	Optional; required for tab layouts		border, tab	For tab layouts, the text to display on the tab. For border layouts, if you specify this attribute, ColdFusion creates a title bar for the layout area with the specified text as the title. By default, border layouts that are not closable or collapsible do not have a title bar. You cannot use this attribute for border layout areas with a position attribute value of center.

Usage

All cflayoutarea tags must be children of cflayout tags and cannot have cflayoutarea tags as immediate children, but they can contain cflayout tags. However, the cflayoutarea tags do not have to be direct children of the cflayout tag; instead, the cflayout tag could have a tag such as cfloop or cfquery as a child, and the cflayoutarea tags could be in the body of the cfloop or cfquery tag. These rules let you create arbitrarily complex combinations of different layouts.

Note: *You cannot put a layout of type* `border` *inside a layout of type* `tab`*.*

If you do not specify a size attribute value, ColdFusion attempts to determine the required size for the layout area contents. However, in some cases, such as when the layout area contains AJAX controls, ColdFusion might not be able to determine the required size, and you must specify the size attribute to make the AJAX control appear. In these cases, a scroll bar appears for the layout area.

You can use a source attribute or a tag body to specify the layout area contents; if you specify both, ColdFusion uses the contents specified by the source attribute and ignores the tag body. If you use a source attribute, an animated icon and the text "Loading..." appears while the contents is being fetched.

If the source attribute specifies a page that defines JavaScript functions, the function definitions on that page must have the following format:

```
functionName = function(arguments) {function body}
```

Function definitions that use the following format may not work:

```
function functionName (arguments) {function body}
```

However, Adobe recommends that you include all custom JavaScript in external JavaScript files and import them on the application's main page, and not write them inline in code that you get using the source attribute. Imported pages do not have this function definition format restriction.

If you use the source attribute, you can use a *bind expression* to include form field values or other form control attributes as part of the source specification. You can bind to HTML format form controls only. For detailed information on using bind expressions see "Using Ajax Data and Development Features" on page 648 in the *ColdFusion Developer's Guide*.

In `border` type layouts, a center layout area always takes up any space that is not used by the other areas, even if you do not specify a `cflayoutarea` tag with a `center` `position` attribute. Therefore, if you want only two layout areas in either direction, one of the two must be the center area, or you must explicitly size the two areas to take up the full layout area.

When you nest layouts, you must set the inner layout area initial sizes appropriately to ensure that they appear.

Use the following JavaScript functions to enable, disable, show, hide, expand, collapse, and select layout areas:

Function	Description
ColdFusion.Layout.createTab	Creates a tab in an existing tabbed layout.
ColdFusion.Layout.disableTab	Disables the specified tab so it cannot be selected.
ColdFusion.Layout.enableTab	Enables the specified tab so users can select it and display the area contents.
ColdFusion.Layout.hideTab	Hides a tab.
ColdFusion.Layout.selectTab	Selects a tab and displays the layout area contents.
ColdFusion.Layout.showTab	Shows a tab that was hidden using the `inithide` attribute or the `hideTab()` function.
ColdFusion.Layout.collapseArea	Collapses an area of a border layout.
ColdFusion.Layout.expandArea	Expands a collapsed area of a border layout.
ColdFusion.Layout.getTabLayout	Hides an area of a border layout.
ColdFusion.Layout.hideArea	Hides an area of a border layout.
ColdFusion.Layout.showArea	Shows an area of a border layout that was hidden using the `inithide` attribute or the `hideArea()` function.

Note: *When you use the `style` attribute to specify the background color of a border layout area, you must specify a `height` style of 100% to make the background color cover the entire layout area. This is because the style specification applies to an inner content area of the layout area, not the layout area itself, and the 100% specification ensures that the content area takes up all available space in the layout area.*

Example

The following example creates a three-tabbed layout and lets you use buttons to dynamically control the second tab.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
<h3>Atab</h3>
<cflayout type="tab" name="thelayout" tabheight="175" style="background-color:##CCffFF;
        color:red; height:200">
    <cflayoutarea title="Tab 1" style="background-color:##FFAAFF;" closable="true">
        This is text in layout area 1
    </cflayoutarea>
    <cflayoutarea name="area2" title="Tab 2" inithide="true"
            style="background-color:##FFCCFF" >
        This is text in layout area 2
    </cflayoutarea>
    <cflayoutarea title="Tab 3" style="background-color:##FF99FF;">
        This is text in layout area 3
    </cflayoutarea>
</cflayout>
```

```
<br />
<cfform>
    <cfinput name="show" width="40" value="show tab" type="button"
        onClick="ColdFusion.Layout.showTab('thelayout', 'area2');">
    <cfinput name="hide" width="40" value="hide tab" type="button"
        onClick="ColdFusion.Layout.hideTab('thelayout', 'area2');">
    <cfinput name="enable" width="40" value="enable tab" type="button"
        onClick="ColdFusion.Layout.enableTab('thelayout', 'area2');">
    <cfinput name="disable" width="40" value="disable tab" type="button"
        onClick="ColdFusion.Layout.disableTab('thelayout', 'area2');">
    <cfinput name="select" width="40" value="select tab" type="button"
        onClick="ColdFusion.Layout.selectTab('thelayout', 'area2');">
</cfform>
</body>
</html>
```

cfldap

Description

Provides an interface to a Lightweight Directory Access Protocol (LDAP) directory server, such as the Netscape Directory Server.

Category

Internet protocol tags

Syntax

```
<cfldap
    action = "action"
    server = "server name"
    attributes = "attribute, attribute"
    delimiter = "delimiter character"
    dn = "distinguished name"
    filter = "filter"
    maxRows = "number"
    modifyType = "replace|add|delete"
    name = "name"
    password = "password"
    port = "port number"
    rebind = "yes|no"
    referral = "number of allowed hops"
    returnAsBinary = "column name, column name"
    scope = "scope"
    secure = "multifield security string"
    separator = "separator character"
    sort = "attribute[, attribute]..."
    sortControl = "nocase|desc|asc"
    start = "distinguished name"
    startRow = "row number"
    timeout = "milliseconds"
    username = "user name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfftp`, `cfhttp`, `cfmail`, `cfmailparam`, `cfpop`, "Managing LDAP Directories" on page 435 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the ability to use a comma as a delimiter when specifying a list of variables in the `returnAsBinary` attribute, for example, `returnAsBinary="objectGUID,objectSID"`. Previously, the allowed delimiter was a space.

ColdFusion MX 7: Added the `returnAsBinary` attribute. Added SSL V2 client based authentication; this means that ColdFusion supports the CFSSL_CLIENT_AUTH option. If CFSSL_CLIENT_AUTH is selected, ColdFusion assumes that the first certificate in the cacerts (or the certificate database) contains the Client Certificate.

ColdFusion MX:

- Changed the `name` attribute behavior: this tag validates the query name in the `name` attribute.

• Changed sorting behavior: this tag does not support client-side sorting of query results. (It supports server-side sorting; use the `sort` and `sortcontrol` attributes.)

• Changed how results are sorted: server-side sorting results might be sorted slightly differently than in ColdFusion 5. If you attempt a sort against a server that does not support it, ColdFusion MX throws an error.

• Deprecated the `filterConfig` and `filterFile` attributes. They might not work, and might cause an error, in later releases.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required	`query`	• `query`: returns LDAP entry information only. Requires `name`, `start`, and `attributes` attributes. • `add`: adds LDAP entries to LDAP server. Requires `attributes` attribute. • `modify`: modifies LDAP entries, except distinguished name `dn` attribute, on LDAP server. Requires `dn`. See `modifyType` attribute. • `modifyDN`: modifies distinguished name attribute for LDAP entries on LDAP server. Requires `dn`. • `delete`: deletes LDAP entries on an LDAP server. Requires `dn`.
`server`	Required		Host name or IP address of LDAP server.
`attributes`	Required if `action = "Query"`, `"Add"`, `"ModifyDN"`, or `"Modify"`		For queries: comma-delimited list of attributes to return. For queries, to get all attributes, specify `"*"`. If `action = "add" or "modify"`, you can specify a list of update columns. Separate attributes with a semicolon. If `action = "ModifyDN"`, ColdFusion passes attributes to the LDAP server without syntax checking.
`delimiter`	Optional	`;` (semicolon)	Separator between attribute name-value pairs. Use this attribute if either of these situations exist: • The `attributes` attribute specifies more than one item. • An attribute contains the default delimiter (semicolon), for example: `mgrpmsgrejecttext;lang-en` Used by `query`, `add`, and `modify` actions, and by `cfldap` to output multivalue attributes. For example, if $ (dollar sign), you could specify `"cn = Double Tree Inn$street = 1111 Elm; Suite 100`, where the semi-colon is part of the street value.
`dn`	Required if `action = "Add"`, `"Modify"`, `"ModifyDN"`, or `"delete"`		Distinguished name, for `update` action, for example, `"cn = Bob Jensen, o = Ace Industry, c = US"`
`filter`	Optional	`"objectclass = *"`	Search criteria for `action = "query"`. List attributes in the form: `"(attribute operator value)"` For example: `"(sn = Smith)"`
`maxRows`	Optional		Maximum number of entries for LDAP queries.

Attribute	Req/Opt	Default	Description
modifyType	Optional	replace	How to process an attribute in a multivalue list: • add: appends it to any attributes. • delete: deletes it from the set of attributes. • replace: replaces it with specified attributes. You cannot add an attribute that is already present or that is empty.
name	Required if action = "Query"		Name of LDAP query. The tag validates the value.
password	Required if secure = "CFSSL_BASIC"		Password that corresponds to user name. If secure = "CFSSL_BASIC", V2 encrypts the password before transmission.
port	Optional	389	Port.
rebind	Optional	no	• yes: attempts to rebind referral callback and reissue query by referred address using original credentials. • no: referred connections are anonymous.
referral	Optional		Integer. Number of hops allowed in a referral. A value of 0 disables referred addresses for LDAP; no data is returned.
returnAsBinary	Optional		A space-delimited list of columns that are to be returned as binary values.
scope	Optional	oneLevel	Scope of search, from entry specified in start attribute for action = "Query". • oneLevel: entries one level below entry. • base: only the entry. • subtree: entry and all levels below it.
secure	Optional		Security to employ, and required information. One of the following: • CFSSL_BASIC provides V2 SSL encryption and server authentication
separator	Optional	, (comma)	Delimiter to separate attribute values of multi-value attributes. Used by query, add, and modify actions, and by cfldap to output multi-value attributes. For example, if $ (dollar sign), the attributes attribute could be "objectclass = top$person", where the first value of objectclass is top, and the second value is person. This avoids confusion if values include commas.
sort	Optional		Attribute(s) by which to sort query results. Use a comma delimiter.
sortControl	Optional	asc	• nocase: case-insensitive sort. • asc: ascending (a to z) case-sensitive sort. • desc: descending (z to a) case-sensitive sort. You can enter a combination of sort types; for example, sortControl = "nocase, asc".
start	Required if action = "Query"		Distinguished name of entry to be used to start a search.
startRow	Optional	1	Used with action = "query". First row of LDAP query to insert into a ColdFusion query.

Attribute	Req/Opt	Default	Description
timeout	Optional	60000	Maximum length of time, in milliseconds, to wait for LDAP processing.
username	Required if secure = "CFSSL_BASIC"	(anonymous)	User ID.

Usage

If you use the query action, cfldap creates a query object, allowing access to information in the query variables, as follows:

Variable name	Description
queryname.recordCount	Number of records returned by query
queryname.currentRow	Current row of query that cfoutput is processing
queryname.columnList	Column names in query

If you use the security="CFSSL_BASIC" option, ColdFusion determines whether to trust the server by comparing the server's certificate with the information in the jre/lib/security/cacerts keystore of the JRE used by ColdFusion. The ColdFusion default cacerts file contains information about many certificate granting authorities. If you must update the file with additional information, you can use the keytool utility in the ColdFusion jre/bin directory to import certificates that are in X.509 format. For example, enter the following:

```
keytool -import -keystore cacerts -alias ldap -file ldap.crt -keypass bl19mq
```

Then restart ColdFusion. The keytool utility initial keypass password is "change it". For more information on using the keytool utility, see the Sun JDK documentation.

Characters that are illegal in ColdFusion can be used in LDAP attribute names. As a result, the cfldap tag could create columns in the query result set whose names contain illegal characters and are, therefore, inaccessible in CFML. In ColdFusion, illegal characters are automatically mapped to the underscore character; therefore, column names in the query result set might not exactly match the names of the LDAP attributes.

For usage examples, see the *ColdFusion Developer's Guide*.

Example

```
<h3>cfldap Example</h3>
<p>Provides an interface to LDAP directory servers. The example uses the
University of Connecticut public LDAP server. For more public LDAP servers,
see <a href="http://www.emailman.com">http://www.emailman.com</a>.</p>
<p>Enter a name and search the public LDAP resource.
An asterisk before or after the name acts as a wildcard.</p>
<!--- If form.name exists, the form was submitted; run the query. --->
<cfif IsDefined("form.name")>
    <!--- Check to see that there is a name listed. --->
    <cfif form.name is not "">
        <!--- Make the LDAP query. --->
        <cfldap
             server = "ldap.uconn.edu"
             action = "query"
             name = "results"
             start = "dc=uconn,dc=edu"
             filter = "cn=#name#"
             attributes = "cn,o,title,mail,telephonenumber"
             sort = "cn ASC">
        <!--- Display results. --->
```

```
        <center>
        <table border = 0 cellspacing = 2 cellpadding = 2>
            <tr>
                <th colspan = 5>
                    <cfoutput>#results.recordCount# matches found </cfoutput></TH>
            </tr>
            <tr>
                <th><font size = "-2">Name</font></TH>
                <th><font size = "-2">Organization</font></TH>
                <th><font size = "-2">Title</font></TH>
                <th><font size = "-2">E-Mail</font></TH>
                <th><font size = "-2">Phone</font></TH>
            </tr>
            <cfoutput query = "results">
                <tr>
                    <td><font size = "-2">#cn#</font></td>
                    <td><font size = "-2">#o#</font></td>
                    <td><font size = "-2">#title#</font></td>
                    <td><font size = "-2">
                        <A href = "mailto:#mail#">#mail#</A></font></td>
                    <td><font size = "-2">#telephonenumber#</font></td>
                </tr>
            </cfoutput>
            </table>
            </center>
        </cfif>
</cfif>

<form action="#cgi.script_name#" method="POST">
    <p>Enter a name to search in the database.</p>
    <input type="Text" name="name">
    <input type="Submit" value="Search" name="">
</form>
```

cflocation

Description

Stops execution of the current page and opens a ColdFusion page or HTML file.

Category

Flow-control tags, Page processing tags

Syntax

```
<cflocation
    url = "URL"
    addToken = "yes|no"
    statusCode = "300|301|302|303|304|305|307">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfabort`, `cfbreak`, `cfexecute`, `cfexit`, `cfif`, `cfloop`, `cfswitch`, `cfthrow`, `cftry`

History

ColdFusion 8: Added the `statusCode` attribute.

Attributes

Attribute	Req/Opt	Default	Description
url	Required		URL of HTML file or CFML page to open.
addToken	Optional		The `clientManagement` attribute must be enabled (see `cfapplication`). • *yes*: appends client variable information to URL. • *no*
statusCode	Optional		The HTTP status code, as follows: • 300 HTTP_MULTIPLE_CHOICES: The requested address refers to more than one entity. • 301 HTTP_MOVED_PERMANENTLY: The page is assigned a new URI. The change is permanent. • 302 HTTP_MOVED_TEMPORARILY: The page is assigned a new URI. The change is temporary. • 303 HTTP_SEE_OTHER: The client should try another network address. • 304 HTTP_NOT_MODIFIED: The requested resource has not been modified. • 305 HTTP_USE_PROXY: The requested resource must be accessed through the proxy given by the Location field. • 307 HTTP_TEMPORARY_REDIRECT: The requested data temporarily resides at a new location. The status codes from 304 to 307 do not redirect you to the page specified in a URL, unless you also follow the guidelines specified in the most recent HTTP RFC.

Usage

You might write a standard message or response in a file, and call it from several applications. Use this tag to redirect the user's browser to the standard file.

This tag has no effect if you use it after the `cfflush` tag on a page.

Example

```
<h3>cflocation Example</h3>
<p>This tag redirects the browser to a web resource; normally, you would use this tag to go
to a CF page or an HTML file on the same server. The addToken attribute lets you send client
information to the target page.</p>
<p>If you remove the comments, this code redirects you to CFDOCS home page:</p>

<!--- <cflocation url = "http://localhost:8500/cfdocs/dochome.htm" addToken = "no"> --->
```

cflock

Description

Ensures the integrity of shared data. Instantiates the following kinds of locks:

- Exclusive - Allows single-thread access to the CFML constructs in its body. The tag body can be executed by one request at a time. No other requests can start executing code within the tag while a request has an exclusive lock. ColdFusion issues exclusive locks on a first-come, first-served basis.
- Read-only - Allows multiple requests to access CFML constructs within the tag body concurrently. Use a read-only lock only when shared data is read and not modified. If another request has an exclusive lock on shared data, the new request waits for the exclusive lock to be released.

Category

Application framework tags

Syntax

```
<cflock
    timeout = "time-out in seconds"
    name = "lock name"
    scope = "Application|Server|Session|Request"
    throwOnTimeout = "yes|no"
    type = "readOnly|exclusive">
    <!--- CFML to be synchronized. --->
</cflock>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfapplication`, `cfassociate`, `cfmodule`, "Using Persistent Data and Locking" on page 273 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added `Request` value to `scope` attribute.

Attributes

Attribute	Req/Opt	Default	Description
`timeout`	Required		Maximum length of time, in seconds, to wait to obtain a lock. If lock is obtained, tag execution continues. Otherwise, behavior depends on `throwOnTimeout` attribute value. If you set `timout="0"`, the timeout is determined by the "Timeout Requests after *x*" setting in the ColdFusion Administrator Settings page, if that setting is enabled. However, if the setting is not enabled, and you set `timeout="0"`, ColdFusion can wait indefinitely to obtain the lock.
`name`	Optional		Locks name. Mutually exclusive with the `scope` attribute. Only one request can execute the code within a `cflock` tag with a given name at a time. Cannot be an empty string. Permits synchronizing access to resources from different parts of an application. Lock names are global to a ColdFusion server. They are shared among applications and user sessions, but not clustered servers.

Attribute	Req/Opt	Default	Description
scope	Optional		Locks scope. Mutually exclusive with the name attribute. Only one request in the specified scope can execute the code within this tag (or within any other cflock tag with the same lock scope) at a time. • Application • Request • Server • Session
throwOnTimeout	Optional	yes	How time-out conditions are handled: • yes: exception is generated for the time-out. • no: execution continues past this tag.
type	Optional	exclusive	• readOnly: lets more than one request read shared data. • exclusive: lets one request read or write shared data.

Note: *Limit the scope of code that updates shared data structures, files, and CFXs. Exclusive locks are required to ensure the integrity of updates, but read-only locks are faster. In a performance-sensitive application, substitute read-only locks for exclusive locks where possible, for example, when reading shared data.*

Usage

ColdFusion MX is a multithreaded server; it can process multiple page requests at a time. Use the cflock tag for these purposes:

- To ensure that modifications to shared data and objects made in concurrently executing requests occur sequentially.
- Around file manipulation constructs, to ensure that file updates do not fail because files are open for writing by other applications or tags.
- Around CFX invocations, to ensure that ColdFusion can safely invoke CFXs that are not implemented in a thread-safe manner. (This applies only to CFXs developed in C++.)

To work safely with ColdFusion, a C++ CFX that maintains and manipulates shared (global) data structures must be made thread-safe; however, this requires advanced knowledge. You can use a CFML custom tag wrapper around a CFX to make its invocation thread-safe.

When you display, set, or update variables in a shared scope, use the scope attribute to identify the scope as Server, Application or Session.

Deadlocks

A *deadlock* is a state in which no request can execute the locked section of a page. After a deadlock occurs, neither user can break it, because all requests to the protected section of the page are blocked until the deadlock can be resolved by a lock time-out.

The cflock tag uses kernel level synchronization objects that are released automatically upon time out and/or the abnormal termination of the thread that owns them. Therefore, while processing a cflock tag, ColdFusion never deadlocks for an infinite period of time. However, very large time-outs can block request threads for long periods, and radically decrease throughput. To prevent this, always use the minimum time-out value.

Another cause of blocked request threads is inconsistent nesting of cflock tags and inconsistent naming of locks. If you nest locks, everyone accessing the locked variables must consistently nest cflock tags in the same order. Otherwise, a deadlock can occur.

These examples show situations that cause deadlocks:

Example deadlock with two users	
User 1	**User 2**
Locks the session scope.	Locks the Application scope.
Deadlock: Tries to lock the Application scope, but it is already locked by User 2.	Deadlock: Tries to lock the Session scope, but it is already locked by User 1.

The following deadlock could occur if you tried to nest an exclusive lock inside a read lock:

Example deadlock with one user
User 1
Locks the Session scope with a read lock.
Attempts to lock the Session scope with an exclusive lock.
Deadlock: Cannot lock the Session scope with an exclusive lock because the scope is already locked for reading.

The following code shows this scenario:

```
<cflock timeout = "60" scope = "SESSION" type = "readOnly">
    ..............
    <cflock timeout = "60" scope = "SESSION" type = "Exclusive">
    .........
    </cflock>
</cflock>
```

To avoid a deadlock, everyone who nests locks must do so in a well-specified order and name the locks consistently. If you must lock access to the Server, Application, and Session scopes, you must do so in this order:

1. Lock the Session scope. In the `cflock` tag, specify `scope = "session"`.
2. Lock the Application scope. In the `cflock` tag, specify `scope = "Application"`.
3. Lock the Server scope. In the `cflock` tag, specify `scope = "server"`.
4. Unlock the Server scope.
5. Unlock the Application scope.
6. Unlock the Session scope.

Note: *If you do not have to lock a scope, you can skip any pair of these lock/unlock steps. For example, if you do not have to lock the Server scope, you can skip Steps 3 and 4. Similar rules apply for named locks.*

For more information, see the following:

- "Using Persistent Data and Locking" on page 273 in the *ColdFusion Developer's Guide*.
- "Locking thread data and resource access" on page 307 in the *ColdFusion Developer's Guide* (for information on locking the Request scope when you use the `cfthread` tag to create multithreaded ColdFusion applications).
- *ColdFusion Locking Best Practices*, on the Adobe support website.

Example

```
<!---
This example shows how cflock can guarantee consistency of data updates to variables in the
    Application, Server, and Session scopes. --->
```

```
<!--- Copy the following code into an Application.cfm file in the
    application root directory. --->
<!----------------   Beginning of Application.cfm code   --------------->
<!--- cfapplication defines scoping for a ColdFusion application and enables or disables
    storing of application and session variables. Put this tag in a special file called
    Application.cfm. It is run before any other ColdFusion page in its directory. --->

<!--- Enable session management for this application. --->
<cfapplication name = "ETurtle"
    sessionTimeout = #CreateTimeSpan(0,0, 0, 60)#
    sessionManagement = "yes">

<!--- Initialize session and application variables used by E-Turtleneck. Use session scope
    for the session variables. --->
<cflock scope = "Session"
    timeout = "30" type = "Exclusive">
    <cfif NOT IsDefined("session.size")>
        <cfset session.size = "">
    </cfif>
    <cfif NOT IsDefined("session.color")>
        <cfset session.color = "">
    </cfif>
</cflock>

<!--- Use an application lock for the application-wide variable that keeps track of the
    number of turtlenecks sold. For a more efficient, but more complex, way of handling
    Application scope locking, see the "ColdFusion Developer's Guide"--->
<cflock scope = "Application" timeout = "30" type = "Exclusive">
    <cfif NOT IsDefined("application.number")>
        <cfset application.number = 0>
    </cfif>
</cflock>

<!----------------------- End of Application.cfm ----------------------->

<h3>cflock Example</h3>

<cfif IsDefined("form.submit")>
<!--- The form has been submitted; process the request. --->
    <cfoutput>
        Thanks for shopping E-Turtleneck. You chose size <b>#form.size#</b>,
        color <b>#form.color#</b>.<br><br>
    </cfoutput>

<!--- Lock the code that assigns values to session variables. ---->
    <cflock scope = "Session" timeout = "30" type = "Exclusive">
        <cfparam name = session.size Default = #form.size#>
        <cfparam name = session.color Default = #form.color#>
    </cflock>

<!---- Lock the code that updates the Application scope number of turtlenecks sold. --->
    <cflock scope = "Application" timeout = "30" type = "Exclusive">
        <cfset application.number = application.number + 1>
    <cfoutput>
        E-Turtleneck has now sold #application.number# turtlenecks!
    </cfoutput>
    </cflock>

<cfelse>
<!--- Show the form only if it has not been submitted. --->
```

```
    <cflock scope = "Application" timeout = "30" type = "Readonly">
        <cfoutput>
            E-Turtleneck has sold #application.number# turtlenecks to date.
        </cfoutput>
    </cflock>

    <form method="post" action="cflocktest.cfm">
        <p>Congratulations! You selected the most comfortable turtleneck in the world.
        Please select color and size.</p>
        <table cellspacing = "2" cellpadding = "2" border = "0">
            <tr>
                <td>Select a color.</td>
                <td><select type = "Text" name = "color">
                        <option>red
                        <option>white
                        <option>blue
                        <option>turquoise
                        <option>black
                        <option>forest green
                    </select>
                </td>
            </tr>
            <tr>
                <td>Select a size.</td>
                <td><select type = "Text" name = "size" >
                        <option>XXsmall
                        <option>Xsmall
                        <option>small
                        <option>medium
                        <option>large
                        <option>Xlarge
                    </select>
                </td>
            </tr>
            <tr>
                <td>Press Submit when you are finished making your selection.</td>
                <td><input type = "Submit" name = "submit" value = "Submit"> </td>
            </tr>
        </table>
    </form>
</cfif>
```

cflog

Description

Writes a message to a log file.

Category

Data output tags

Syntax

```
<cflog
    text = "text"
    application = "yes|no"
    file = "filename"
    log = "log type"
    type = "information|warning|error|fatal">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcol`, `cfcontent`, `cfoutput`, `cftable`

History

ColdFusion MX: Deprecated the `thread`, `date`, and `time` attributes. They might not work, and might cause an error, in later releases. (In earlier releases, these attributes determined whether the respective data items were output to the log. In ColdFusion MX, this data is always output.)

Attributes

Attribute	Req/Opt	Default	Description
`text`	Required		Message text to log.
`application`	Optional	`yes`	• `yes`: logs the application name, if it is specified in a `cfapplication` tag or Application.cfc file. • `no`
`file`	Optional		Message file. Specify only the main part of the filename. For example, to log to the Testing.log file, specify "Testing". The file must be located in the default log directory. You cannot specify a directory path. If the file does not exist, it is created automatically, with the extension .log.
`log`	Optional		If you omit the `file` attribute, writes messages to standard log file. Ignored, if you specify `file` attribute. • `Application`: writes to Application.log, normally used for application-specific messages. • `Scheduler`: writes to Scheduler.log, normally used to log the execution of scheduled tasks.
`type`	Optional	`Information`	Type (severity) of the message: • `Information` • `Warning` • `Error` • `Fatal`

Usage

This tag logs custom messages to standard or custom log files. You can specify a file for the log message or send messages to the default application or scheduler log. The log message can include ColdFusion expressions. Log files must have the extension .log and must be located in the ColdFusion log directory.

Log entries are written as comma-delimited lists with these fields:

- type
- thread
- date
- time
- application
- text

Values are enclosed in double quotation marks. If you specify `no` for the `application` attribute, the corresponding entry in the list is empty.

You can disable `cflog` tag execution. For more information, see the ColdFusion Administrator Basic Security page.

The following example logs the name of a user that logs on an application. The message is logged to the file myAppLog.log in the ColdFusion log directory. It includes the date, time, and thread ID, but not the application name.

```
<cflog file="myAppLog" application="no"
    text="User #Form.username# logged on.">
```

For example, if a user enters "Sang Thornfield" in a form's username field, this entry is added to the myApplog.log file entry:

"Information","153","02/28/01","14:53:40",,"User Sang Thornfield logged on."

cflogin

Description

A container for user login and authentication code. ColdFusion runs the code in this tag if a user is not already logged in. You put code in the tag that authenticates the user and identifies the user with a set of roles. Used with `cfloginuser` tag.

Category

Security tags

Syntax

```
<cflogin
    applicationToken = "token"
    cookieDomain = "domain"
    idletimeout = "value">
    ...
    <cfloginuser
        name = "name"
        password = "password"
        roles = "roles">
</cflogin>
```

See also

`cfloginuser`, `cflogout`, `GetAuthUser`, `GetUserRoles`, `IsUserInAnyRole`, `IsUserInRole`, `IsUserLoggedIn`, "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: The `applicationtoken` attribute lets you specify a unique application identifier for each application, or the same value for multiple applications.

ColdFusion MX 6.1: Changed behavior: the `cflogin` variable exists when ColdFusion receives a request with NTLM or Digest (HTTP Negotiated header) authentication information.

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`applicationtoken`	Optional	The current application name	The login that applies to the application. To let users log in to only one application, specify a unique value for that application. To let users log in to multiple applications, specify the same value for those applications. If you do not set a value for the `applicationtoken` attribute, the default value is `CFAUTHORIZATION_`*`applicationname`*.
`cookiedomain`	Optional		Domain of the cookie that is used to mark a user as logged in. Use this attribute to enable a user login cookie to work with multiple clustered servers in the same domain.
`idletimeout`	Optional	1800	Time interval, in seconds, after which ColdFusion logs off the user.

Usage

The body of this tag executes only if there is no logged-in user. When using application-based security, you put code in the body of the `cflogin` tag to check the user-provided ID and password against a data source, LDAP directory, or other repository of login identification. The body must include a `cfloginuser` tag to establish the authenticated user's identity in ColdFusion.

You control the data source and are responsible for coding the SQL within the cflogin tag; you must make sure that the associated database has user, password, and role information.

The cflogin tag has a built-in cflogin structure that contains two variables, cflogin.name and cflogin.password, if the page is executing in response to any of the following:

- Submission of a form that contains input fields with the names j_username and j_password.
- A request that uses HTTP Basic authentication and, therefore, includes an Authorization header with the user name and password.
- A request that uses NTLM or Digest authentication. In this case, the username and password are hashed using a one-way algorithm in the Authorization header; ColdFusion gets the username from the web server and sets the cflogin.password value to the empty string.

You can use these values in the cflogin tag body to authenticate the user, and, in the cfloginuser tag, to log the user in. The structure is only available in the cflogin tag body.

Example

The following example shows a simple authentication. This code is typically in the Application.cfc onRequestStart method or in the application.cfm page.

```
<cflogin>
    <cfif NOT IsDefined("cflogin")>
        <cfinclude template="loginform.cfm">
        <cfabort>
    <cfelse>
        <cfif cflogin.name eq "admin">
            <cfset roles = "user,admin">
        <cfelse>
            <cfset roles = "user">
        </cfif>
        <cfloginuser name = "#cflogin.name#" password = "#cflogin.password#"
            roles = "#roles#"/>
    </cfif>
</cflogin>
```

The following view-only example checks the user ID and password against a data source:

```
<cfquery name="qSecurity"
    datasource="UserRolesDb">
    SELECT Roles FROM SecurityRoles
    WHERE username=<cfqueryparam value='#cflogin.name#' CFSQLTYPE="CF_SQL_VARCHAR"
    AND password=<cfqueryparam value='#cflogin.password#' CFSQLTYPE='CF_SQL_VARCHAR'
</cfquery>

<cfif qSecurity.recordcount gt 0>
<cfloginuser name = "#cflogin.name#"
    password = "#cflogin.password#"
    roles = "#trim(qSecurity.Roles)#" >
</cfif>
```

cfloginuser

Description

Identifies an authenticated user to ColdFusion. Specifies the user ID and roles. Used within a `cflogin` tag.

Category

Security tags

Syntax

```
<cfloginuser
    name = "name"
    password = "password"
    roles = "roles">
```

See also

`cflogin`, `cflogout`, `GetAuthUser`, `GetUserRoles`, `IsUserInAnyRole`, `IsUserInRole`, `IsUserLoggedIn`, "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 6.1: Changed behavior: if the Session scope is enabled, and the `cfapplication` tag `loginStorage` attribute is set to Session, the login remains in effect until the session expires or the user is logged out by the `cflogout` tag.

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		A user name.
`password`	Required		A user password.
`roles`	Required		A comma-delimited list of role identifiers.
			ColdFusion processes spaces in a list element as part of the element.

Usage

Used inside the `cflogin` tag to identify the authenticated user to ColdFusion. After you call this function, the `GetAuthUser` and `IsUserInRole` return the user name and role information.

Note: *By default, the user information is stored as memory-only cookies. The `cfapplication` tag or the Application.cfc This.loginStorage variable can specify that login information is stored in the Session scope.*

Example

See "cflogin" on page 373.

cflogout

Description

Logs the current user out. Removes knowledge of the user ID, password, and roles from the server. If you do not use this tag, the user is automatically logged out when the session ends.

Category

Security tags

Syntax

```
<cflogout>
```

See also

cflogin, cfloginuser, GetAuthUser, GetUserRoles, IsUserInAnyRole, IsUserInRole, IsUserLoggedIn, "Securing Applications" on page 312 in *ColdFusion Developer's Guide*

History

ColdFusion MX 6.1: Changed behavior: if the Session scope is enabled, a login remains in effect until the session expires or the user is logged out by the `cflogout` tag.

ColdFusion MX: Added this tag.

Example

```
<cflogin>
    <cfloginuser
        name = "foo"
        password ="bar"
        roles = "admin">
</cflogin>
<cfoutput>Authorized user: #getAuthUser()#</cfoutput>
<cflogout>
<cfoutput>Authorized user: #getAuthUser()#</cfoutput>
```

cfloop

Description

Looping is a programming technique that repeats a set of instructions or displays output repeatedly until one or more conditions are met. This tag supports the following types of loops:

- "cfloop: index loop" on page 378
- "cfloop: conditional loop" on page 380
- "cfloop: looping over a date or time range" on page 381
- "cfloop: looping over a query" on page 382
- "cfloop: looping over a list, a file, or an array" on page 383
- "cfloop: looping over a COM collection or structure" on page 385

For more information, see "cfloop and cfbreak" on page 19 and "Populating arrays with data" on page 75 in the *ColdFusion Developer's Guide*.

Category

Flow-control tags

cfloop: index loop

Description

An index loop repeats for a number of times that is determined by a numeric value. An index loop is also known as a FOR loop.

Syntax

```
<cfloop
    index = "parameter name"
    from = "beginning value"
    to = "ending value"
    step = "increment">
        HTML or CFML code ...
</cfloop>
```

See also

cfabort, cfbreak, cfdirectory, cfexecute, cfexit, cfif, cflocation, cfrethrow, cfswitch, cfthrow, cftry; "cfloop and cfbreak" on page 19 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
index	Required		Index value. ColdFusion sets it to the from value and increments or decrements by step value, until it equals the to value.
from	Required		Beginning value of index.
to	Required		Ending value of index.
step	Optional	1	Step by which to increment or decrement the index value.

Usage

Using anything other than integer values in the from and to attributes of an index loop can product unexpected results. For example, if you increment through an index loop from 1 to 2, with a step of 0.1, ColdFusion outputs "1,1.1,1.2,...,1.9", but not "2". This is a programming language problem regarding the internal representation of floating point numbers.

Note: *The to value is evaluated once, when the cfloop tag is encountered. Any change to this value within the loop block, or within the expression that evaluates to this value, does not affect the number of times the loop is executed.*

Example

In this example, the code loops five times, displaying the index value each time:

```
<cfloop index = "LoopCount" from = "1" to = "5">
    The loop index is <cfoutput>#LoopCount#</cfoutput>.<br>
</cfloop>
```

The output of this loop is as follows:

```
The loop index is 1.
The loop index is 2.
The loop index is 3.
The loop index is 4.
The loop index is 5.
```

In this example, the code loops four times, displaying the index value each time. The value of j is decreased by one for each iteration. This does not affect the value of to, because it is a copy of j that is made before entering the loop.

```
<cfset j = 4>
<cfloop index = "LoopCount" from = "1" to = #j#>
    <cfoutput>The loop index is #LoopCount#</cfoutput>.<br>
    <cfset j = j - 1>
</cfloop>
```

The output of this loop is as follows:

```
The loop index is 1.
The loop index is 2.
The loop index is 3.
The loop index is 4.
```

As before, the value of `j` is decremented by one for each iteration, but this does not affect the value of `to`, because its value is a copy of `j` that is made before the loop is entered.

In this example, `step` has the default value, 1. The code decrements the index:

```
<cfloop index = "LoopCount"
    from = "5"
    to = "1"
    step = "-1">
The loop index is <cfoutput>#LoopCount#</cfoutput>.<br>
</cfloop>
```

The output of this loop is as follows:

```
The loop index is 5.
The loop index is 4.
The loop index is 3.
The loop index is 2.
The loop index is 1.
```

cfloop: conditional loop

Description

A conditional loop iterates over a set of instructions as long as a condition is True. To use this type of loop correctly, the instructions must change the condition every time the loop iterates, until the condition is False. Conditional loops are known as WHILE loops, as in, "loop WHILE this condition is true."

Syntax

```
<cfloop
    condition = "expression">
    ...
</cfloop>
```

See also

cfabort, cfbreak, cfexecute, cfexit, cfif, cflocation, cfswitch, cfthrow, cftry; "cfloop and cfbreak" on page 19 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
condition	Required		Condition that controls the loop.

Example

The following example increments `CountVar` from 1 to 5.

```
<!--- Set the variable CountVar to 0. --->
<cfset CountVar = 0>
<!--- Loop until CountVar = 5. --->
<cfloop condition = "CountVar LESS THAN OR EQUAL TO 5">
    <cfset CountVar = CountVar + 1>
    The loop index is <cfoutput>#CountVar#</cfoutput>.<br>
</cfloop>
```

The output of this loop is as follows:

```
The loop index is 1.
The loop index is 2.
The loop index is 3.
The loop index is 4.
The loop index is 5.
```

cfloop: looping over a date or time range

Description

Loops over the date or time range specified by the `from` and `to` attributes. By default, the step is 1 day, but you can change the step by creating a timespan. The `cfloop` tag loops over tags that cannot be used within a `cfoutput` tag.

Syntax

```
<cfloop
    from = "start time"
    to = "end time"
    index = "current value"
    step = "increment">
</cfloop>
```

See also

`cfabort`, `cfbreak`, `cfdirectory`, `cfexecute`, `cfexit`, `cfif`, `cflocation`, `cfrethrow`, `cfswitch`, `cfthrow`, `cftry`; "cfloop and cfbreak" on page 19 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`from`	Required		The beginning of the date or time range.
`to`	Required		The end of the date or time range.
`index`	Required	1 day	Index value. ColdFusion sets it to the `from` value and increments by the `step` value, until it equals the `to` value.
`step`	Optional		Step, expressed as a timespan, by which the index increments.

Example

The following example loops from today's date to today's date plus 30 days, stepping by 7 days at a time and displaying the date:

```
<cfset startDate = Now()>
<cfset endDate = Now() + 30>
<cfloop from="#startDate#" to="#endDate#" index="i" step="#CreateTimeSpan(7,0,0,0)#">
<cfoutput>#dateformat(i, "mm/dd/yyyy")#<br /></cfoutput>
</cfloop>
```

The following example displays the time in 30-minute increments, starting from midnight and ending 23 hours, 59 minutes, and 59 seconds later:

```
<cfset startTime = CreateTime(0,0,0)>
<cfset endTime = CreateTime(23,59,59)>
<cfloop from="#startTime#" to="#endTime#" index="i" step="#CreateTimeSpan(0,0,30,0)#">
    <cfoutput>#TimeFormat(i, "hh:mm tt")#<br /></cfoutput>
</cfloop>
```

cfloop: looping over a query

Description

A loop over a query executes for each record in a query record set. The results are similar to those of the `cfoutput` tag. During each iteration, the columns of the current row are available for output. The `cfloop` tag loops over tags that cannot be used within a `cfoutput` tag.

Syntax

```
<cfloop
    query = "query name"
    startRow = "row number"
    endRow = "row number">
</cfloop>
```

See also

`cfabort`, `cfbreak`, `cfexecute`, `cfexit`, `cfif`, `cflocation`, `cfoutput`, `cfswitch`, `cfthrow`, `cftry`; For more information, see "cfloop and cfbreak" on page 19 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`query`	Required		Query that controls the loop.
`startRow`	Optional		First row of query that is included in the loop.
`endRow`	Optional		Last row of query that is included in the loop.

Example

```
<cfquery name = "MessageRecords" dataSource = "cfdocexamples">
    SELECT * FROM Messages
</cfquery>
<cfloop query = "MessageRecords">
<cfoutput>#Message_ID#</cfoutput><br>
</cfloop>
```

The `cfloop` tag also iterates over a record set with dynamic start and stop points. This gets the next *n* sets of records from a query. This example loops from the fifth through the tenth record returned by the MessageRecords query:

```
<cfset Start = 5>
<cfset End = 10>
<cfloop query = "MessageRecords"
    startRow = "#Start#"
    endRow = "#End#">
<cfoutput>#MessageRecords.Message_ID#</cfoutput><br>
</cfloop>
```

The loop stops when there are no more records, or when the current record index is greater than the value of the `endRow` attribute. The following example combines the pages that are returned by a query of a list of page names into one document, using the `cfinclude` tag:

```
<cfquery name = "GetTemplate" dataSource = "Library" maxRows = "5">
    SELECT TemplateName
    FROM Templates
</cfquery>
<cfloop query = "GetTemplate">
    <cfinclude template = "#TemplateName#">
</cfloop>
```

cfloop: looping over a list, a file, or an array

Description

Looping over a list steps through elements contained in any of these entities:

- A variable
- A value that is returned from an expression
- An array
- A file

Looping over a file does not open the entire file in memory.

Syntax

```
<cfloop
    index = "index name"
    array = "array"
    characters = "number of characters"
    delimiters = "item delimiter"
    file = "absolute path and filename">
    list = "list items"
    ...
</cfloop>
```

See also

cfabort, cfbreak, cfexecute, cfexit, cfif, cflocation, cfswitch, cfthrow, cftry; "cfloop and cfbreak" on page 19 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `characters`, `file`, and `array` attributes.

Attributes

Attribute	Req/Opt	Default	Description
`index`	Required		In a list loop, the variable to receive the next list element.
`list`	Required unless you specify a filename in the `file` attribute		A list, variable, or filename; contains a list.
`array`	Optional		An array.
`characters`	Optional		The number of characters to read during each iteration of the loop from the file specified in the `file` attribute. If the value of the `characters` attribute is more than the number of characters in the file, ColdFusion uses the number of characters in the file.
`delimiters`	Optional		Characters that separate items in list.
`file`	Optional		The absolute path and filename of the text file to read, one line at a time. This is helpful when reading large text files, because you can reuse the value of the index variable, which contains the current line of the file. When the loop completes, ColdFusion closes the file.

Example

This loop displays four names:

```
<cfloop index = "ListElement" list = "John,Paul,George,Ringo">
```

```
    <cfoutput>#ListElement#</cfoutput><br>
</cfloop>
```

You can put more than one character in the `delimiters` attribute, in any order. For example, this loop processes commas, colons, and slashes as list delimiters:

```
<cfloop index = "ListElement" list = "John/Paul,George::Ringo" delimiters = ",:/">
    <cfoutput>#ListElement#</cfoutput><br>
</cfloop>
```

ColdFusion skips the second and subsequent consecutive delimiters between list elements. Thus, in the example, the two colons between "George" and "Ringo" are processed as one delimiter.

To loop over each line of a file, use the tag as follows:

```
<cfloop file="c:\temp\simplefile.txt" index="line">
    <cfoutput>#line#</cfoutput><br>
</cfloop>
```

To read a specified number of characters from a text file during each iteration of the loop, use the tag as follows:

```
<cfloop file="c:\temp\simplefile.txt" index="chars" characters="12">
    <cfoutput>#chars#</cfoutput><br>
</cfloop>
```

When you read the following text file, ColdFusion reads 12 characters during each iteration of the loop; the result appears as follows:

Text file	Result
This is line 1.	This is line
This is line 2.	1. This is
This is line 3.	line 2. Th
This is line 4.	is is line 3
This is line 5.	. This is l
This is line 6.	ine 4. This
This is line 7.	is line 5.
This is line 8.	This is lin
This is line 9.	e 6. This i
This is line 10.	s line 7. T
This is line 11.	his is line
	8. This is
	line 9. Thi
	s is line 10
	. This is l
	ine 11.

To loop over an array, you can do the following:

```
<cfset x = ["mars","earth", "venus", "jupiter"]>
<cfloop array=#x# index="name">
    <cfoutput>#name#</cfoutput><br>
</cfloop>
```

cfloop: looping over a COM collection or structure

Description

The `cfloop collection` attribute loops over every object within a COM/DCOM collection object, or every element in a structure:

- A COM/DCOM collection object is a set of similar items referenced as a group. For example, the group of open documents in an application is a collection.
- A structure contains a related set of items, or it can be used as an associative array. Looping is particularly useful when using a structure as an associative array.

In the loop, each item is referenced by the variable name in the `item` attribute. The loop executes until all items have been accessed.

The `collection` attribute is used with the `item` attribute. In the example that follows, `item` is assigned a variable called `file2`, so that with each cycle in the `cfloop`, each item in the collection is referenced. In the `cfoutput` section, the `name` property of the `file2` item is referenced for display.

For more information, see "Integrating COM and CORBA Objects in CFML Applications" on page 974 in the *ColdFusion Developer's Guide*.

Example

This example uses a COM object to output a list of files. In this example, `FFunc` is a collection of `file2` objects.

```
<cfobject
    class = FileFunctions.files
    name = FFunc
    action = Create>
<cfset FFunc.Path = "c:\">
<cfset FFunc.Mask = "*.*" >
<cfset FFunc.attributes = 16 >
<cfset x = FFunc.GetFileList()>
<cfloop collection = #FFUNC# item = "file2">
    <cfoutput> #file2.name# <br> </cfoutput>
</cfloop>
<!---Loop through a structure that is used as an associative array: --->
...
<!--- Create a structure and loop through its contents. --->
<cfset Departments = StructNew()>
<cfset val = StructInsert(Departments, "John ", "Sales ")>
<cfset val = StructInsert(Departments, "Tom ", "Finance ")>
<cfset val = StructInsert(Departments, "Mike ", "Education ")>
<!--- Build a table to display the contents --->
<cfoutput>
<table cellpadding = "2 " cellspacing = "2 ">
    <tr>
        <td><b>Employee</b></td>
        <td><b>Dept.</b></td>
    </tr>
<!--- Use item to create the variable person to hold value of key as loop runs. --->
<cfloop collection = #Departments# item = "person ">
    <tr>
        <td>#person#</td>
        <td>#StructFind(Departments, person)#</td></tr>
</cfloop>
</table>
</cfoutput>
```

cfmail

Description

Sends an e-mail message that optionally contains query output, using an SMTP server.

Category

Communications tags, Internet protocol tags

Syntax

```
<cfmail
   from = "e-mail address"
   to = "comma-delimited list"
   bcc = "comma-delimited list"
   cc = "comma-delimited list"
   charset = "character encoding"
   debug = "yes|no"
   failto = "e-mail address"
   group = "query column"
   groupcasesensitive = "yes|no"
   mailerid = "header id"
   maxrows = "integer"
   mimeattach = "path"
   password = "string"
   port = "integer"
   priority = "integer or string priority level"
   query = "query name"
   replyto = "e-mail address"
   server = "SMTP server address"
   spoolenable = "yes|no"
   startrow = "query row number"
   subject = "string"
   timeout = "number of seconds"
   type = "mime type"
   username = "SMTP user ID"
   useSSL = "yes|no"
   useTLS = "yes|no"
   wraptext = "column number">

(Optional) Mail message body and/or cfhttpparam tags

</cfmail>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfmailparam`, `cfmailpart`, `cfpop`, `cfftp`, `cfhttp`, `cfldap`, `Wrap`; "Using ColdFusion with mail servers" on page 998 in "Sending and Receiving E-Mail" on page 998 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added `priority`, `useSSL`, and `useTLS` attributes.

ColdFusion MX 7:

- The `cfmail` tag no longer lets you send multipart mail by embedding the entire MIME-encoded message in the tag body. Use the `cfmailpart` tag, instead.

- The `cfmail` tag renders nonproportional fonts proportionately. This is a behavior change from ColdFusion 5. ColdFusion MX 7 uses UTF-8 and sends this in the mail header (Content-Type: text/plain; charset=UTF-8). ColdFusion 5 uses ISO-8859-1 (Latin 1). To avoid this behavior, add the `charset="ISO-8859-1"` attribute to restore the default ColdFusion 5 encoding. Alternatively, you can change the encoding on the Mail page in the ColdFusion Administrator.

ColdFusion MX 6.1:

- Added the following attributes: `charset`, `failto`, `replyto`, `username`, `password` and `wraptext`.
- Added support for multiple mail servers in the `server` attribute.
- Added several configuration options to the ColdFusion Administrator Mail Settings page.

ColdFusion MX: Added the `SpoolEnable` attribute.

Attributes

Attribute	Req/Opt	Default	Description
`bcc`	Optional		Addresses to which to copy the message, without listing them in the message header. To specify multiple addresses, separate the addresses with commas.
`cc`	Optional		Addresses to which to copy the message. To specify multiple addresses, separate the address with commas.
`charset`	Optional	Character encoding selected in ColdFusion Administrator Mail page; `utf-8`	Character encoding of the mail message, including the headers. The following list includes commonly used values: • `utf-8` • `iso-8859-1` • `windows-1252` • `us-ascii` • `shift_jis` • `iso-2022-jp` • `euc-jp` • `euc-kr` • `big5` • `hz-gb-2312` • `euc-cn` • `utf-16` For more information on character encodings, see www.w3.org/International/O-charset.html.
`debug`	Optional	`no`	• `yes`: sends debugging output to standard output. By default, if the console window is unavailable, ColdFusion sends output to cf_root\runtime\logs\coldfusion-out.log on server configurations. On J2EE configurations, with JRun, the default location is *jrun_home*/logs/*servername*-out.log. *Caution*: If you set this option to `yes`, ColdFusion writes detailed debugging information to the log, including all message contents, and can generate large logs quickly. • `no`: does not generate debugging output.
`failto`	Optional		Address to which mailing systems should send delivery failure notifications. Sets the mail envelope reverse-path value.

Attribute	Req/Opt	Default	Description
from	Required		E-mail message sender: • A static string; for example, "support@mex.com" • A variable; for example, "#GetUser.EMailAddress#". This attribute does not have to be a valid Internet address; it can be any text string.
to	Required		Message recipient e-mail addresses: • Static address, for example, "support@.com" . • Variable that contains an address, for example, "#Form.Email#". • Name of a query column that contains an address, for example, "#EMail#". An e-mail message is sent for each returned row. To specify multiple addresses, separate the addresses with commas.
subject	Required		Message subject. Can be dynamically generated. For example, to send messages that give customers status updates: "Status of Order Number #Order_ID#".
group	Optional	CurrentRow	Query column to use when you group sets of records to send as a message. For example, to send a set of billing statements to a customer, group on "Customer_ID." Case-sensitive. Eliminates adjacent duplicates when data is sorted by the specified field.
groupcasesensitive	Optional	No	Boolean. Whether to consider case when using the group attribute. To group on case-sensitive records, set this attribute to Yes.
mailerid	Optional	ColdFusion Application Server	Mailer ID to be passed in X-Mailer SMTP header, which identifies the mailer application.
maxrows	Optional		Maximum number of messages to send when looping over a query.
mimeattach	Optional		Path of file to attach to message. Attached file is MIME-encoded. ColdFusion attempts to determine the MIME type of the file; use the cfmailparam tag to send an attachment and specify the MIME type.
password	Optional		A password to send to SMTP servers that require authentication. Requires a username attribute.
port	Optional		TCP/IP port on which SMTP server listens for requests (normally 25). A value here overrides the Administrator.
priority	Optional	3	The message priority level. Can be one of the following values: • An integer in the range 1-5; 1 represents the highest priority. • One of the following string values, which correspond to the numeric values: highest or urgent, high, normal, low, and lowest or non-urgent.
query	Optional		Name of cfquery from which to draw data for messages. Use this attribute to send more than one message, or to send query results within a message.
replyto	Optional		Addresses to which the recipient is directed to send replies.
server	Optional		SMTP server address, or (Enterprise edition only) a comma-delimited list of server addresses, to use for sending messages. At least one server must be specified here or in the ColdFusion Administrator. A value here overrides the Administrator. A value that includes a port specification overrides the port attribute. For details, see Usage.

Attribute	Req/Opt	Default	Description
spoolenable	Optional		Whether to spool mail or always send it Immediately. Overrides the ColdFusion Administrator Spool mail messages to disk for delivery setting. • yes: saves a copy of the message until the sending operation is complete. Pages that use this option might run slower than those that use the No option. • no: queues the message for sending, without storing a copy until the operation is complete. If a delivery error occurs when this option is No, ColdFusion generates an Application exception and logs the error to the mail.log file.
startrow	Optional	1	Row in a query to start from.
timeout	Optional		Number of seconds to wait before timing out connection to SMTP server. A value here overrides the Administrator.
type	Optional	text/plain	MIME type of the message. Can be a valid MIME media type or one of the following: • text: specifies text/plain type. • plain: specifies text/plain type. • html: specifies text/html type. For a list of all registered MIME media types, see www.iana.org/assignments/media-types/.
username	Optional		A user name to send to SMTP servers that require authentication. Requires a password attribute.
useSSL	Optional		Whether to use Secure Sockets Layer.
useTLS	Optional		Whether to use Transport Level Security.
wraptext	Optional	Do not wrap text	The maximum line length, in characters of the mail text. If a line has more than the specified number of characters, replaces the last white space character, such as a tab or space, preceding the specified position with a line break. If there are no white space characters, inserts a line break at the specified position. A common value for this attribute is 72.

Usage

Sends a mail message to the specified address. Mail messages can include attachments. The tag body can include CFML code to generate mail output. The cfmailparam and cfmailpart tags can only be used in the cfmail tag body.

Mail messages can be single or multipart. If you send a multi-part mail message, all message content must be in cfmailpart tags; ColdFusion ignores multipart message text that is not in cfmailpart tags.

Note: *The cfmail tag does not make copies of attachments when spooling mail to disk. If you use the cfmail tag to send a message with an attachment with spooling enabled and you use the cffile tag to delete the attachment file, ColdFusion might not send the mail because the mailing process might execute after the file was deleted. (When this happens, the mail log includes a FileNotFound exception and the e-mail is not sent.) You can prevent this problem by setting SpoolEnable="No" in the attribute or disabling spooling in the ColdFusion Administrator. Disabling spooling causes the e-mail to be delivered immediately.*

Mail addressing

Mail addresses can have any of the following forms:

Format	Example
user@server	rsmith@company.com
<user@server>	<rsmith@company.com>
DisplayName <user@server>	Rob Smith <rsmith@company.com>
"DisplayName" <user@server>	"Rob Smith" <rsmith@company.com>
user@server (DisplayName)	rsmith@company.com (Rob Smith)

Specifying mail servers

The `server` attribute can specify one or more mail servers.

Note: *If you specify multiple mail servers in ColdFusion Standard, the* `cfmail` *tag uses only the first server in the specification. ColdFusion logs a warning message to the mail log file and ignores the additional servers.*

For each server, you can optionally specify a user name, password, and port. These values override the corresponding attributes, if any. The `server` attribute has the following format:

```
[user:password@]server[:port],[user:password@]server[:port],....
```

For example, the following line specifies one server, mail.myco.com that uses the default port and no user or password, and a second server with a user, password, and specific port:

```
server=mail.myco.com,mail_admin:adm2qzf@mail2.myco.com:24
```

When you specify multiple mail servers in ColdFusion Enterprise, ColdFusion tries the available servers in the order they are listed until it connects to a server. ColdFusion does not try to connect to a server that was unavailable in the last 60 seconds.

Example

```
<h3>cfmail Example</h3>

<!--- Delete the surrounding comments to use this example.

<cfif IsDefined("form.mailto")>
    <cfif form.mailto is not "" AND form.mailfrom is not "" AND form.Subject is not "">
        <cfmail to = "#form.mailto#" from = "#form.mailFrom#" subject = "#form.subject#">
                This message was sent by an automatic mailer built with cfmail:
                = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                #form.body#
        </cfmail>
        <h3>Thank you</h3>
        <p>Thank you, <cfoutput>#mailfrom#: your message, #subject#, has been sent to
            #mailto#</cfoutput>.</p>
    </cfif>
</cfif>
<p>
<form action = "cfmail.cfm" method="POST">
    <pre>
    TO: <input type = "Text" name = "MailTo">
    FROM: <input type = "Text" name = "MailFrom">
    SUBJECT: <input type = "Text" name = "Subject">
    <hr>
    MESSAGE BODY:
    <textarea name ="body" cols="40" rows="5" wrap="virtual"></textarea>
    </pre>
    <!--- Establish required fields. --->
```

```
    <input type = "hidden" name = "MailTo_required" value = "You must enter a recipient">
    <input type = "hidden" name = "MailFrom_required" value = "You must enter a sender">
    <input type = "hidden" name = "Subject_required" value = "You must enter a subject">
    <input type = "hidden" name = "Body_required" value = "You must enter some text">
    <p><input type = "Submit" name = ""></p>
</p>
</form>
```

cfmailparam

Description

Attaches a file or adds a header to an e-mail message.

Category

Communications tags, Internet protocol tags

Syntax

```
<cfmail
    to = "recipient"
    subject = "message subject"
    from = "sender"
        more attributes... >
    <cfmailparam
        contentID = "content ID"
        disposition = "disposition type">
        file = "filename"
        type ="media type"

OR

    <cfmailparam
        name = "header name"
        value = "header value">
    ...
</cfmail>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfmail`, `cfmailpart`, `cfftp`, `cfhttp`, `cfldap`, `cfpop`; "Using the cfmailparam tag" on page 1004 in "Sending and Receiving E-Mail" on page 998 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 6.x: Added the `Disposition` and `ContentID` attributes.

ColdFusion MX 6.1: Added the `type` attribute.

Attributes

Attribute	Req/Opt	Default	Description
`contentID`	Optional		The Identifier for the attached file. This ID should be globally unique and is used to identify the file in an IMG or other tag in the mail body that references the file content.
`disposition`	Optional	`attachment`	How the attached file is to be handled. Can be one of the following: • `attachment`: presents the file as an attachment. • `inline`: displays the file contents in the message.
`file`	Required if you do not specify `name` attribute		Attaches a file in a message. Mutually exclusive with `name` attribute. The file is MIME encoded before sending.
`name`	Required if you do not specify `file` attribute		Name of header. Case-insensitive. Mutually exclusive with `file` attribute.

Attribute	Req/Opt	Default	Description
type	Optional		The MIME media type of the file. Not used with the name attribute. Can be a valid MIME media type or one of the following: • text: specifies text/plain type. • plain: specifies text/plain type. • html: specifies text/html type. If you specify the type, the value you specify becomes the content type header; otherwise, ColdFusion generates the content type header. **Note:** For a list of all registered MIME media types, see www.iana.org/assignments/media-types/.
value	Optional		Value of the header. Not used with the file attribute.

Usage

This tag attaches a file or adds a header to an e-mail message. It can only be used in the cfmail tag. You can use multiple cfmailparam tags within a cfmail tag.

You can use this tag to include a file, such as an image, in an HTML mail message. The file can be displayed inline in an HTML message, or as an attachment, as Example 2 shows. To include multiple files, use multiple cfmailparam tags.

Display a file inline in a mail message

1 Specify type="html" in the cfmail tag.

2 Specify disposition="inline" and a ContentID attribute in the cfmailparam tag.

3 Use a src="cid:*ContentIDValue*" attribute to identify the content to include in the HTML tag such as the img tag.

Example

Example 1: This view-only example uses the cfmailparamtag to add a header to a message, attach files, and to return a receipt to the sender.

```
<cfmail from = "peter@domain.com" To = "paul@domain.com"
    Subject = "See Important Attachments and Reply">
    <cfmailparam name = "Importance" value = "High">
    Please review the new logo. Tell us what you think.
    <cfmailparam file = "c:\work\readme.txt" type="text/plain">
    <cfmailparam file = "c:\work\logo.gif" type="image/gif">
    <cfmailparam name="Disposition-Notification-To" value="peter@domain.com">
</cfmail>
```

Example 2: This view-only example displays an image in the body of an HTML message.

```
<cfmail type="HTML"
        to = "#form.mailto#"
        from = "#form.mailFrom#"
        subject = "Sample inline image">
    <cfmailparam file="C:\Inetpub\wwwroot\web.gif"
        disposition="inline"
        contentID="image1">
    <p>There should be an image here</p>
    <img src="cid:image1">
    <p>After the picture</p>
</cfmail>
```

cfmailpart

Description

Specifies one part of a multipart e-mail message. Can only be used in the `cfmail` tag. You can use more than one `cfmailpart` tag within a `cfmail` tag.

Category

Communications tags, Internet protocol tags

Syntax

```
<cfmail
    ... >
    (Optional cfmailparam entries)
    <cfmailpart
        charset="character encoding"
        type="mime type"
        wraptext="number"
    >
    Mail part contents
    </cfmailpart>
    ...
</cfmail>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfmail`, `cfmailparam`, `cfpop`, `cfftp`, `cfhttp`, `cfldap`, `cfcontent`, `Wrap`; "E-mail" on page 350 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 6.1: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
charset	Optional	Character encoding specified by charset attribute of cfmail tag	The character encoding in which the part text is encoded. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • hz-gb-2312 • euc-cn • utf-16 For more information on character encodings, see www.w3.org/International/O-charset.html.
type	Required		The MIME media type of the part. Can be a can be valid MIME media type or one of the following: • text: specifies text/plain type. • plain: specifies text/plain type. • html: specifies text/html type. **Note**: For a list of all registered MIME media types, see www.iana.org/assignments/media-types/.
wraptext	Optional	Do not wrap text	Specifies the maximum line length, in characters of the mail text. If a line has more than the specified number of characters, replaces the last white space character, such as a tab or space, preceding the specified position with a line break. If there are no white space characters, inserts a line break at the specified position. A common value for this attribute is 72.

Usage

Use this tag to create mail messages with alternative versions of the message that duplicate the content in multiple formats. The most common use is to send a plain text version of the message that can be read by all mail readers followed by a version formatted in HTML for display by HTML-compatible mail readers. Specify the simplest version first, with more complex versions afterwards. For more information, see www.ietf.org/rfc/rfc2046.txt.

Example

```
<h3>cfmailpart Example</h3>
<cfmail from = "peter@domain.com" To = "paul@domain.com"
    Subject = "Which version do you see?">
    <cfmailpart type="text" wraptext="74">
        You are reading this message as plain text, because your mail reader does not handle
            HTML text.
    </cfmailpart>
    <cfmailpart type="html">
        <h3>HTML Mail Message</h3>
        <p>You are reading this message as <strong>HTML</strong>.</p>
```

```
        <p>Your mail reader handles HTML text.</p>
    </cfmailpart>
</cfmail>
```

cfmenu

Description

Creates a horizontal or vertical menu. Any menu item can be the top level of a submenu.

Category

Display management tags

Syntax

```
<cfmenu
    bgcolor="HTML color value"
    childStyle="CSS style specification"
    font="HTML font family"
    fontColor="HTML color value"
    fontSize="Number of pixels"
    menuStyle="CSS style specification"
    name="string"
    selectedFontColor="HTML color value"
    selectedItemColor="HTML color value"
    type="horizontal|vertical
    width="Number of pixels">

        cfmenuitem tags

</cfmenu>
```

The `cfmenu` tag must have a body that contains at least one `cfmenuitem` tag to define the menu items and an end `</cfmenu>` tag.

Note: *You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.*

See also

`cfajaximport`, `cfmenuitem`, "Using menus and toolbars" on page 624 in "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
bgColor	Optional	Background color style of the menu	The color of the menu background. You can use any valid HTML color specification. This specification has the following behaviors: • You can override it locally by specifying the menuStyle attribute of this tag and any cfmenuitem tag. • It controls the background of color surrounding a submenu whose background is specified by a childStyle attribute.
childStyle	Optional		A CSS style specification that applies to the following menu items: • The items of the top-level menu • All child menu items, including the children of submenus This attribute lets you use a single style specification for all menu items.
font	Optional	Browser default font	The font to use for all child menu items. Use any valid HTML font-family style attribute. Some common values are serif, sans-serif, Times, Courier, and Arial.
fontColor	Optional	black	The color of the menu text. Use any valid HTML color specification.
fontSize	Optional	Font size of the menu item	The size of the font. Use a numeric value, such as 8, to specify a pixel character size. Use a percentage value, such as 80%, to specify a size relative to the default font size. Font sizes larger than 20 pixels can result in submenu text exceeding the menu boundary.
menuStyle	Optional		A CSS style specification that applies to the menu, including any parts of the menu that do not have items. If you do not specify style information in the cfmenuitem tags, this attribute controls the style of the top-level items.
name	Optional		The name of the menu.
selectedFontColor	Optional	black	The color of the text for the menu item that has the focus. Use any valid HTML color specification.
selectedItemColor	Optional	light blue	The color that highlights the menu item that has the focus. You can use any valid HTML color specification.
type	Optional	horizontal	The orientation of the menu. The following values are valid: • horizontal: Menu items are arranged horizontally. • vertical: Menu items are arranged vertically. Submenus of both menu types are always arranged vertically.
width	Optional	Width of the container	The width of a vertical menu; not valid for horizontal menus. Use a numeric value, such as 50, to specify a pixel size. Use a percentage value, such as 30%, to specify a size relative to the parent element's size.

Usage

The cfmenu tag defines a horizontal or vertical ColdFusion menu. You use a single cfmenu tag to define the general menu characteristics, and you use cfmenuitem child tags to define the individual menu entries and any submenus. You create submenus by putting cfmenuitem tags in the body of a cfmenuitem tag.

You cannot nest a cfmenu tag inside a form or inside a cfmenu tag or cfmenuitem tag.

Example

The following example creates a simple menu bar. When you click an entry in the bar, the browser displays the Adobe website page for the selected product. You can expand the ColdFusion item by clicking the icon, and then select an item to display a specific ColdFusion web page.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
<cfmenu name="menu" type="horizontal" fontsize="14" bgcolor="##CCFFFF">
    <cfmenuitem name="acrobat" href="http://www.adobe.com/acrobat" display="Acrobat"/>
    <cfmenuitem name="aftereffects" href="http://www.adobe.com/aftereffects"
        display="After Effects"/>
    <!--- The ColdFusion menu item has a pop-up menu. --->
    <cfmenuitem name="coldfusion"
            href="http://www.adobe.com/products/coldfusion" display="ColdFusion">
        <cfmenuitem name="buy"
            href="http://www.adobe.com/products/coldfusion/buy/" display="Buy"/>
        <cfmenuitem name="devcenter"
            href="http://www.adobe.com/devnet/coldfusion/" display="Developer Center"/>
        <cfmenuitem name="documentation"
            href="http://www.adobe.com/support/documentation/en/coldfusion/"
                display="Documentation"/>
        <cfmenuitem name="support" href="http://www.adobe.com/support/coldfusion/"
            display="Support"/>
    </cfmenuitem>
    <cfmenuitem name="flex" href="http://www.adobe.com/flex" display="Flex"/>
</cfmenu>
</body>
</html>
```

cfmenuitem

Description

Defines an entry in a menu, including an item that is the head of a submenu.

Category

Display management tags

Syntax

```
<cfmenuitem
    display="string"
    childStyle="CSS style specification"
    href="URL or JavaScript function"
    image="path"
    menuStyle="CSS style specification"
    name="string"
    style="CSS style specification"
    target="location identifier">
        Optional child menuitem tags
</cfmenuitem>

OR

<cfmenuitem
    divider[="true"]/>
```

If the `cfmenuitem` tag does not have a body with an end `</cfmenuitem>` tag, you must close the tag with a forward slash character before the closing greater than character (`/>`), for example, `<cfmenuitem divider="true"/>`.

Note: *You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.*

See also

`cfmenu`, "Using menus and toolbars" on page 624 in "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`display`	Required if `divider` attribute is not specified		The text to show as the menu item label.
`childStyle`	Optional	Style determined by parent	A CSS style specification that applies to all child menu items, including the children of submenus.
`divider`	Optional		This attribute specifies that the item is a divider. If you specify this attribute, you cannot specify any other attributes. You can use this attribute without a value, as in the following example: `<cfmenuitem divider />`. You cannot use this attribute in a top-level horizontal menu.
`href`	Optional		A URL link to activate or JavaScript function to call when the user clicks the menu item.

Attribute	Req/Opt	Default	Description
image	Optional		URL of an image to display at the left side of the menu item. The file type can be any format that the browser can display. For most displays, you should use 15x15 pixel images, because larger images conflict with the menu item text.
menuStyle	Optional	Style determined by parent	A CSS style specification that controls the overall style of any submenu of this menu item. This attribute controls the submenu of the current menu item, but not any child submenus of the submenu.
style	Optional	Style determined by parent	A CSS style specification that applies to the current menu item only. It is not overridden by the childStyle attribute.
name	Optional		The name of the menu item.
target	Optional	The current window and frame (if any)	The target in which to display the contents returned by the href attribute. The attribute can be a browser window or frame name, or an HTML target value, such as _self.

Usage

Every cfmenuitem tag must be a child of a cfmenu tag or a cfmenuitem tag. To create a submenu, put the cfmenuitem tags for submenu items in the body of the cfmenuitem tag for the submenu root in the parent menu. For an example of a simple submenu, see cfmenu.

Example

The following menu shows the effects of the various style attributes on the menu and menu item appearance.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
<cfmenu name="menu" type="horizontal" fontsize="14" bgcolor="##FF9999"
    childStyle="font-weight:bold; font-size:12px; border:medium; background-color:##99FF99"
    menuStyle="font-weight:bold; font-style:italic; font-size:14px;
    background-color:##9999FF">
    <cfmenuitem name="acrobatInfo"
        href="http://www.adobe.com/acrobat" display="Acrobat"/>
    <cfmenuitem name="aftereffectsInfo"
        href="http://www.adobe.com/aftereffects" display="After Effects"/>
    <!--- The ColdFusion menu item has a pop-up menu. --->
    <cfmenuitem name="cfInfo"
        childStyle="font-weight:bold; font-size:12px; border:medium;
        background-color:##FF0000" style="font-weight:bold;
        font-style:italic; font-size:16px; border:medium; background color:##00FF00"
        menuStyle="font-weight:bold; font-style:italic; font-size:16px;
        border:medium; background-color:##0000FF"
        href="http://www.adobe.com/products/coldfusion" display="ColdFusion">
    <cfmenuitem name="cfbuy"
        href="http://www.adobe.com/products/coldfusion/buy/" display="Buy"/>
        <cfmenuitem divider="true"/>
        <cfmenuitem name="cfdevcenter"
            href="http://www.adobe.com/devnet/coldfusion/" display="Developer Center"/>
        <cfmenuitem name="cfdocumentation"
            href="http://www.adobe.com/support/documentation/en/coldfusion/"
            display="Documentation">
            <cfmenuitem name="cfmanuals"
                href="http://www.adobe.com/support/documentation/en/coldfusion/
                index.html##manuals" display="Product Manuals"/>
            <cfmenuitem name="cfrelnotes"
                href="http://www.adobe.com/support/documentation/en/coldfusion/
```

```
                releasenotes.html" display="Release Notes"/>
        </cfmenuitem>
        <cfmenuitem name="cfsupport"
                href="http://www.adobe.com/support/coldfusion/" display="Support"/>
    </cfmenuitem>
    <cfmenuitem name="flexInfo" href="http://www.adobe.com/flex" display="Flex">
        <cfmenuitem name="fldocumentation"
            href="http://www.adobe.com/support/documentation/en/flex/"
            display="Documentation" >
            <cfmenuitem name="flmanuals"
                href="http://www.adobe.com/support/documentation/en/flex/
                index.html##manuals" display="Product Manuals" />
        </cfmenuitem>
    </cfmenuitem>
</cfmenu>
</body>
</html>
```

cfmodule

Description

Invokes a custom tag for use in ColdFusion application pages. This tag processes custom tag name conflicts.

Category

Application framework tags

Syntax

```
<cfmodule
    attributeCollection = "collection structure"
    attribute_name1 = "valuea"
    attribute_name2 = "valueb"
    name = "tag name"
    template = "path"
    ...>
```

See also

cfapplication, cfassociate, cflock; "Creating and Using Custom CFML Tags" on page 190 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Changed behavior when using this tag within a custom tag: if the attribute_name parameter is the same as a key element within the attributeCollection parameter, ColdFusion now uses the name value that is within the attributeCollection parameter. (Earlier releases did not process this consistently.)

Attributes

Attribute	Req/Opt	Default	Description
attributeCollection	Optional		Structure. A collection of key-value pairs that represent attribute names and values. You can specify multiple key-value pairs. You can specify this attribute only once. **Note:** This attribute functions differently from the attributeCollection attribute that is supported by most other tags. You must specify the name and template attributes as direct cfmodule tag attributes, not in the attributeCollection structure.
attribute_name	Optional		Attribute for a custom tag. You can include multiple instances of this attribute to specify the parameters of a custom tag.
name	Required unless template attribute is used		Mutually exclusive with the template attribute. A custom tag name, in the form "Name.Name.Name..." Identifies subdirectory, under the ColdFusion tag root directory, that contains custom tag page, for example (Windows format): <cfmodule name = ".Forums40. GetUserOptions"> This identifies the page GetUserOptions.cfm in the directory CustomTags\\Forums40 under the ColdFusion root directory.
template	Required unless name attribute is used		Mutually exclusive with the name attribute. A path to the page that implements the tag. • Relative path: expanded from the current page. • Absolute path: expanded by using ColdFusion mapping. A physical path is not valid.

Usage

To name a ColdFusion page that contains the custom tag definition, including its path, use the `template` attribute. To refer to the custom tag in the ColdFusion installation directory, using dot notation to indicate its location, use the `name` attribute.

On UNIX systems, ColdFusion searches first for a file with a name that matches the `name` attribute, but is all lower case. If it does not find the file, it looks for a file name that matches the attribute with identical character casing.

You can use the `attributeCollection` attribute and explicit custom tag attributes in the same call.

Within the custom tag code, the attributes passed with `attributeCollection` are saved as independent attribute values, with no indication that they are grouped into a structure by the custom tag's caller.

Similarly, if the custom tag uses a `cfassociate` tag to save its attributes, the attributes passed with `attributeCollection` are saved as independent attribute values, with no indication that they are grouped into a structure by the custom tag's caller.

If you specify an end tag to `cfmodule`, ColdFusion calls your custom tag as if it had both a start and an end tag. For more information, see "Handling end tags" on page 198 in the *ColdFusion Developer's Guide*.

Example

```
<h3>cfmodule Example</h3>
<p>This view-only example shows use of cfmodule to call a custom tag inline.</p>
<p>This example uses a sample custom tag that is saved in myTag.cfm in the snippets directory.
You can also save ColdFusion custom tags in the CFusionMX7\CustomTags directory.</p>
<cfset attrCollection1 = StructNew()>
    <cfparam name="attrCollection1.value1" default="22">
    <cfparam name="attrCollection1.value2" default="45">
    <cfparam name="attrcollection1.value3" default="88">
<!--- Call the tag with CFMODULE with Name--->
<cfmodule
    Template="myTag.cfm"
    X="3"
    attributeCollection=#attrCollection1#
    Y="4">
<!--- Show the code. --->
<HR size="2" color="#0000A0">
<P>Here is one way in which to invoke the custom tag, using the TEMPLATE attribute.</P>
<cfoutput>#HTMLCodeFormat(" <CFMODULE
    Template=""myTag.cfm""
    X=3
    attributeCollection=##attrCollection1##
    Y=4>")#
</cfoutput>
<p>The result: <cfoutput>#result#</cfoutput></p>
<!--- Call the tag with CFMODULE with Name.--->
<!---
<CFMODULE
    Name="myTag"
    X="3"
    attributeCollection=#attrCollection1#
    Y="4">
--->
<!--- Show the code. --->
<HR size="2" color="#0000A0">
<p>Here is another way to invoke the custom tag, using the NAME attribute.</p>
<cfoutput>#HTMLCodeFormat(" <CFMODULE
    NAME='myTag'
    X=3
```

```
    attributeCollection=##attrCollection1##
    Y=4>")#
</cfoutput>
<P>The result: <cfoutput>#result#</cfoutput></p>
<!--- Call the tag using the shortcut notation. --->
<!---
<CF_myTag
    X="3"
    attributeCollection=#attrCollection1#
    Y="4">
--->

<!--- Show the code. --->
<p>Here is the short cut to invoking the same tag.</p>
<cfoutput>#HTMLCodeFormat("<cf_mytag
    x = 3
    attributeCollection = ##attrcollection1##
    y = 4>")#
</cfoutput>
<p>The result: <cfoutput>#result#</cfoutput></p>
```

cfNTauthenticate

Description

Authenticates a user name and password against the Windows NT domain on which the ColdFusion server is running, and optionally retrieves the user's groups.

Category

Security tags

Syntax

```
<cfNTauthenticate
    domain="NT domain"
    password="password"
    username="user name"
    listGroups = "yes|no"
    result="result variable"
    throwOnError = "yes|no">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cflogin`, `cfloginuser`, `IsUserInAnyRole`, `GetAuthUser`

History

ColdFusion MX 7: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`domain`	Required		Domain against which to authenticate the user. The ColdFusion J2EE server must be running on this domain.
`password`	Required		User's password.
`username`	Required		User's login name.
`listGroups`	Optional	No	Boolean value that specifies whether to include a comma-delimited list of the user's groups in the result structure.
`result`	Optional	`cfntauthenticate`	Name of the variable in which to return the results.
`throwOnError`	Optional	`no`	Boolean value that specifies whether to throw an exception if the validation fails. If this attribute is `yes`, ColdFusion throws an error if the username or password is invalid; the application must handle such errors in a try/catch block or ColdFusion error handler page.

Usage

Use this function to authenticate a user against a Windows NT domain and optionally get the user's groups. This function does not work with the Microsoft Active Directory directory service, and does nothing on UNIX and Linux systems. You typically use this tag inside a `cflogin` tag to authenticate the user for a `cfloginuser` tag, as the example shows.

Note: *ColdFusion must run as a user that has the privilege to authenticate other users in the specified domain.*

The structure specified in the `result` attribute contains the following information:

Field	Value
auth	Whether the user is authenticated: • yes • no
groups	A comma-delimited list of the user's groups in the specified domain. The structure includes this field only if the listGroups attribute is yes.
name	The user name; equals the tag's username attribute.
status	The authentication status. One of the following: • success • UserNotInDirFailure: the user is not listed in the directory. • AuthenticationFailure: the user is in the directory, but the password is not valid.

This tag provides two models for handling authentication: status checking and exception handling. If the throwOnError attribute is no, use the result variable's auth and status fields to determine whether the user was authenticated and, if not, the reason for the failure. If the throwOnError attribute is yes, ColdFusion throws an exception error if the user is not valid. In this case, use try/catch error handling. The catch block must handle any authentication failure.

Example

The following example uses the auth and status fields to determine whether the user is authenticated and the failure cause. It consists of three files that you put in the same directory:

- A main cfntauthexample.cfm page that displays the name if the user is authenticated and contains a logout link.
- A login form page that is displayed if the user is not logged in.
- The Application.cfm page, which contains all the login, authentication, and logout processing code.

For a full description of login processing, see the *ColdFusion Developer's Guide*. For information on how this example works, see the comments in the code.

Save the following page as cfntauthenticateexample.cfm. To run the example, request this page in your browser or IDE.

```
<!--- The Application.cfm page, which is processed each time a user
    requests this page, ensures that you log in first. --->
<cfoutput>
    <h3>Welcome #GetAuthUser()#</h3>
    <!--- A link to log out the user. --->
    <a href="#CGI.script_name#?logout=Yes">Log Out</a>
</cfoutput>
```

Save the following page as loginform.cfm:

```
<!--- A simple login form that posts back to the page whose request initiated the login. --->
<h2>Please Log In</h2>
<cfform action="#CGI.script_name#">
    <!--- j_username and j_password are special names that populate cflogin tag
        variables. --->
    User Name: <cfinput type="text" name="j_username" value="cfqa_user1" required="Yes"><br>
    Password: <cfinput type="password" name="j_password" value="cfqa_user1"
        required="Yes"><br>
    Domain: <cfinput type="text" name="domain" value="rnd" required="Yes"><br>
    <input type="submit" value="Log In">
</cfform>
```

Save the following page as Application.cfm:

```
<!--- If this page is executing in response to the user clicking a logout link,
        log out the user. The cflogin tag code will then run. --->
<cfif IsDefined("URL.logout") AND URL.logout>
 <cflogout>
</cfif>

<!--- The cflogin body code runs only if a user is not logged in. --->
<cflogin>
    <!--- cflogin variable exists only if login credentials are available. --->
    <cfif NOT IsDefined("cflogin")>
        <!--- Show a login form that posts back to the page whose request
            initiated the login, and do not process the rest of this page. --->
        <cfinclude template="loginform.cfm">
        <cfabort>
    <cfelse>
        <!--- Trim any leading or trailing spaces from the username and password
            submitted by the form. --->
        <cfset theusername=trim(form.j_username)>
        <cfset thepassword=trim(form.j_password)>
        <cfset thedomain=trim(form.domain)>
        <cfntauthenticate username="#theusername#" password="#thepassword#"
            domain="#thedomain#" result="authresult" listgroups="yes">
        <!--- authresult.auth is True if the user is authenticated. --->
        <cfif authresult.auth>
            <!--- Log user in to ColdFusion and set roles to the user's Groups. --->
            <cfloginuser name="#theusername#" password="#thepassword#"
                roles="#authresult.groups#">
        <cfelse>
            <!--- The user was not authenticated.
                    Display an error message and the login form. --->
            <cfoutput>
                <cfif authresult.status IS "AuthenticationFailure">
                    <!--- The user is valid, but not the password. --->
                    <h2>The password for #theusername# is not correct<br>
                        Please Try again</h2>
                <cfelse>
                    <!--- There is one other status value, invalid user name. --->
                    <H2>The user name #theusername# is not valid<br>
                        Please Try again</h2>
                </cfif>
            </cfoutput>
            <cfinclude template="loginform.cfm">
            <cfabort>
        </cfif>
    </cfif>
</cflogin>
```

cfobject

Description

Creates a ColdFusion object of a specified type.

Note: *You can enable and disable this tag in the ColdFusion Administrator page, under ColdFusion Security > Sandbox Security.*

Category

Extensibility tags

Syntax

The tag syntax depends on the object type. Some types use the `type` attribute; others do not. See the following sections:

- "cfobject: .NET object" on page 410
- "cfobject: COM object" on page 413
- "cfobject: component object" on page 415
- "cfobject: CORBA object" on page 416
- "cfobject: Java or EJB object" on page 418
- "cfobject: web service object" on page 420

Note: *On UNIX, this tag does not support COM objects.*

See also

`cfargument`, `cfcomponent`, `cffunction`, `cfinvoke`, `cfinvokeargument`, `cfproperty`, `cfreturn`; "Using Java objects" on page 938 in the *ColdFusion Developer's Guide*

History

ColdFusion 8:

- Added `password`, `proxyPassword`, `proxyPort`, `proxyServer`, `proxyUser`, `refreshWSDL`, `userName`, `wsdl2JavaArgs`, and `wsportname` attributes to for use with web service objects.
- Added .NET/dotnet type and the associated `assembly`, `port`, `protocol`, and `secure` attributes.

ColdFusion MX:

- Changed instantiation behavior: this tag, and the `CreateObject` function, can now instantiate ColdFusion components (CFCs); you can use them within the `cfscript` tag.
- For CORBA object: changed the Naming Service separator format for addresses from a dot to a forward slash. For example, if "`context=NameService`", for a class, use either of the following formats for the `class` parameter:
  - "`/Eng/CF`"
  - "`.current/Eng.current/CF`"

  (In earlier releases, the format was "`.Eng.CF`".)
- For CORBA object: changed the `locale` attribute; it specifies the Java configuration that contains the properties file.

cfobject: .NET object

Description

Creates a .NET object, that is, a ColdFusion proxy for accessing a class in a local or remote .NET assembly.

Syntax

```
<cfobject
    class="class name"
    name="instance name"
    type=".NET|dotnet"
    action="create"
    assembly="absolute path"
    port="6086"
    protocol="tcp|http"
    secure="no|yes"
    server = "localhost">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`CreateObject: .NET object`, `DotNetToCFType`, "Using Microsoft .NET Assemblies" on page 952 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added `.NET` and `dotnet` type values, and the `assembly`, `port`, `protocol`, and `secure` attributes.

Attributes

Attribute	Req/Opt	Default	Description
`class`	Required		Name of the .NET class to instantiate as an object.
`name`	Required		String; reference name of the component to use in your application.
`type`	Required for .NET		Object type. Must be `.NET` or `dotnet` for .NET objects.
`action`	Optional	create	Action to take. Must be `create`.
`assembly`	Optional.	mscorlib.dll which contains the .NET core classes.	**For local .NET assemblies**, the absolute path or paths to the assembly or assemblies (EXE or DLL files) from which to access the .NET class and its supporting classes. If a class in an assembly requires supporting classes that are in other assemblies, you must also specify those assemblies. You can, however, omit the supporting assemblies for the following types of supporting classes: • .NET core classes (classes in mscorlib.dll) • Classes in assemblies that are in the global assembly cache (GAC) To specify multiple assemblies, use a comma-delimited list. **For remote .NET assemblies**, you must specify the absolute path or paths of the local proxy JAR file or files that represent the assemblies. If you omit this attribute, and there is no local .NET installation, the tag fails without generating an error. If you omit this attribute, there is a local .NET installation, and the specified class is not in the .NET core classes, ColdFusion generates an error.
`port`	Optional	6086	Port number at which the .NET-side agent is listening.

Attribute	Req/Opt	Default	Description
protocol	Optional	tcp	Protocol to use to use for communication between ColdFusion and .NET. Must be one of the following values: • http: Use HTTP/SOAP communication protocol. This option is slower than tcp, but might be required for access through a firewall. • tcp: Use binary TCP/IP protocol. This method is more efficient than HTTP.
secure	Optional	false	Whether to secure communications with the .NET-side agent. If true, ColdFusion uses SSL to communicate with .NET.
server	Optional	localhost	Host name or IP address of the server where the .NET-side agent is running. Can be in any of these forms: • server name (for example, myserver) • IP address (for example, 127.0.0.1) You must specify this attribute to access .NET components on a remote server.

Usage

The cfobject tag with a .NET or dotnet value for the type attribute creates a reference to a .NET object of a given class. Using the reference, you can access the .NET object's fields and methods. The .NET classes do not have to be local, and you can use the cfobject tag on a system that does not have .NET installed, including UNIX-based or OS-X systems.

To access .NET assemblies, you must do the following:

• Install the ColdFusion 8 .NET extension and run the .NET extension service on the system on which the assemblies are installed. You do not have to install the extension or run the extension service on a ColdFusion system that accesses *only* remote assemblies. For installation instructions, see *Installing and Using ColdFusion*.

• If the assemblies are located on a remote system, create Java proxies for the .NET classes that you use, copy the proxies to the ColdFusion system, and configure the remote system for access by the proxies. For information on these steps, see "Using Microsoft .NET Assemblies" on page 952 in the *ColdFusion Developer's Guide*. If the .NET assemblies are on your ColdFusion system, you do not have to perform these steps.

Accessing methods and fields

You call .NET methods as you use any other ColdFusion object methods. In the simplest case, your application code uses the following format to call a .NET class method:

```
<cfobject type=".NET" name="mathInstance" class="mathClass">
    assembly="C:/Net/Assemblies/math.dll">
<cfset myVar=mathInstance.multiply(1,2)>
```

If a .NET class has multiple constructors, and you do not want ColdFusion to use the default constructor to create the object, invoke a specific constructor by calling the special init method of the ColdFusion object with the constructor's arguments. For example, you can use the following tags to instantiate com.foo.MyClass(int, int):

```
<cfobject type=".NET" class="com.foo.MyClass"
    assembly="c:\temp\myLib.dll" name="myObj" >
<cfset myObj.init(10, 5)>
```

You access and change .NET class public fields by calling the following methods:

```
Get_fieldName()
Set_fieldName()
```

For example, if the .NET class has a public field named account, you can access and modify its value by using Get_acount() and Set_account() methods, respectively.

You can access, but not modify final fields, so you can only call `Get_fieldName()` for these fields.

Example

The following example uses the GetProcess method of the .NET System.Diagnostics.Process class to get and display information about the processes running on the local system. Because it uses a core .NET class, for which ColdFusion automatically generates proxies, you do not have to specify an assembly name in the `cfobject` tag.

For more complex examples, including examples that use custom .NET classes, see "Using Microsoft .NET Assemblies" on page 952 in the *ColdFusion Developer's Guide*.

```
<cfobject type=".NET" name="proc" class="System.Diagnostics.Process">
<cfset processes = proc.GetProcesses()>
<cfset arrLen = arrayLen(processes)>

<table border=0 cellspacing="3" cellpadding="3">
    <tr bgcolor="#33CCCC">
        <td style="font-size:12px; font-weight:bold" nowrap>Process ID</td>
        <td style="font-size:12px; font-weight:bold" nowrap>Name</td>
        <td style="font-size:12px; font-weight:bold" nowrap>Memory (KB)</td>
        <td style="font-size:12px; font-weight:bold" nowrap>Peak Memory (KB)</td>
        <td style="font-size:12px; font-weight:bold" nowrap>Virtual Memory Size (KB)</td>
        <td style="font-size:12px; font-weight:bold" nowrap>Start Time</td>
        <td style="font-size:12px; font-weight:bold" nowrap>Total Processor Time</td>
    </tr>
    <cfloop from = 1 to="#arrLen#" index=i>
        <cfset process = processes[i]>
        <cfset id = process.Get_Id()>
        <cfif id neq 0>
            <cfoutput>
            <tr>
                <td align="right">#process.Get_Id()#</td>
                <td>#process.Get_ProcessName()#</td>
                <td align="right">#process.Get_PagedMemorySize()/1000#</td>
                <td align="right">#process.Get_PeakPagedMemorySize()/1000#</td>
                <td align="right">#process.Get_VirtualMemorySize()/1000#</td>
                <td>#process.Get_StartTime()#</td>
                <td>#process.Get_TotalProcessorTime()#</td>
            </tr>
            </cfoutput>
        </cfif>
    </cfloop>
</table>
```

cfobject: COM object

Description

Creates and manipulates a Component Object Model (COM) object. Invokes a registered automation server object type.

For information on OLEView, and about COM and DCOM, see the Microsoft OLE Development website: www.microsoft.com.

To use this tag, you must provide the object's program ID or filename, the methods and properties available through the IDispatch interface, and the arguments and return types of the object's methods. For most COM objects, you can get this information with the OLEView utility.

Note: *On UNIX, the* `cfobject` *tag does not support COM objects.*

Syntax

```
<cfobject
    class = "program ID"
    name = "instance name"
    action = "create|connect"
    context = "inproc|local|remote"
    server = "server name">
    type = "com"
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`ReleaseComObject`, `cfcollection`, `cfexecute`; "COM" on page 351 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`class`	Required		Component ProgID for the object to invoke. When using Java stubs to connect to the COM object, the class must be the ProgID of the COM object.
`name`	Required		String; name for the instantiated component.
`action`	Optional	`create`	• `create`: instantiates a COM object (typically, a DLL) before invoking methods or properties. • `connect`: ionnects to a COM object (typically, an EXE) running on server.
`context`	Optional		• `inproc` • `local` • `remote` In Windows, if not specified, uses Registry setting.

Attribute	Req/Opt	Default	Description
server	Required if context = "Remote"		Server name, using Universal Naming Convention (UNC) or Domain Name Serve (DNS) convention, in one of these forms: • \\lanserver • lanserver • http://www.servername.com • www.servername.com • 127.0.0.1
type	Optional		Object type. The value com specifies COM objects:

Example

```
<h3>cfobject (COM) Example</h3>
<!--- Create a COM object as an inproc server (DLL). (class = prog-id)--->
<cfobject action = "Create"
    type = "COM"
    class = Allaire.DocEx1.1
    name = "obj">

<!--- Call a method. Methods that expect no arguments should be called by using
    empty parentheses. --->
<cfset obj.Init()>

<!--- This is a collection object. It should support, at a minimum:
    Property : Count
    Method : Item(inarg, outarg)
    and a special property called _NewEnum
--->
<cfoutput>
    This object has #obj.Count# items.
    <br> <HR>
</cfoutput>

<!--- Get the 3rd object in the collection. --->
<cfset emp = obj.Item(3)>
<cfoutput>
    The last name in the third item is #emp.lastname#.
    <br> <HR>
</cfoutput>
<!---Loop over all the objects in the collection.--->
<p>Looping through all items in the collection:
<br>
<cfloop
    collection = #obj#
    item = file2>
    <cfoutput>Last name: #file2.lastname# <br></cfoutput>
</cfloop>
```

cfobject: component object

Description

Creates an instance of a ColdFusion component (CFC) object.

Syntax

```
<cfobject
    component = "component name"
    name = "instance name"
    type = "component">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcollection`, `cfcomponent`, `cfexecute`, `cfindex`, `IsInstanceOf`, `cfreport`, `cfsearch`, `cfwddx`; "Using ColdFusion components" on page 170 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`component`	Required		Name of component to instantiate.
`name`	Required		String; name for the instantiated component. The name must not have a period as the first or last character.
`type`	Optional	`component`	The object type. You can omit this attribute or specify `component`. ColdFusion automatically sets the type to component.

Usage

When the `cfobject` tag creates an instance of the CFC, ColdFusion executes any constructor code in the CFC; that is, it runs code that is not in the method definitions.

On UNIX systems, ColdFusion searches first for a file with a name that matches the specified component name, but is all lowercase. If it does not find the file, it looks for a filename that matches the component name exactly, with the identical character casing.

Example

```
<!--- Separate instantiation and method invocation; --->
<!--- permits multiple invocations. --->
<cfobject
    name="quoteService"
    component="nasdaq.quote">
<cfinvoke
    component="#quoteService#"
    method="getLastTradePrice"
    symbol="macr"
    returnVariable="res">
<cfoutput>#res#</cfoutput><br>

<cfinvoke
    component="#quoteService#"
    method="getLastTradePrice"
    symbol="mot"
    returnVariable="res">
<cfoutput>#res#</cfoutput>
```

cfobject: CORBA object

Description

Calls methods on a registered CORBA object.

Syntax

```
<cfobject
    class = "filepath or naming service"
    context = "ior|nameservice"
    name = "instance name"
    type = "corba"
    locale = "type-value arguments">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcollection`, `cfexecute`, `cfindex`, `cfreport`, `cfsearch`, `cfwddx`; "CORBA" on page 351 in the *ColdFusion Developer's Guide*

History

See the History section of the main `cfobject` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`class`	Required		• If `context = "ior"`, absolute path of file that contains string version of the Interoperable Object Reference (IOR). ColdFusion must be able to read file; it should be local to ColdFusion server or accessible on network. • If `context = "nameservice"`, forward slash-delimited naming context for naming service, for example: `Allaire//Doc/empobject`.
`context`	Required		• `ior`: ColdFusion uses Interoperable Object Reference (IOR) to access CORBA server. • `nameservice`: ColdFusion uses naming service to access server. This option is valid only with the InitialContext of a VisiBroker Orb.
`locale`	Optional		Sets arguments for a call to `init_orb`. Use this attribute only for VisiBroker ORBs. It is available on C++, Version 3.2. The value must be in the form: `locale = " -ORBagentAddr 199.99.129.33  -ORBagentPort 19000"` Each type-value pair must start with a hyphen.
`name`	Required		String; name for the instantiated component. An application uses it to reference the CORBA object's methods and attributes.
`type`	Required for CORBA		Object type. Must be `corba` for CORBA objects.

Usage

ColdFusion Enterprise version 4.0 and later supports CORBA through the Dynamic Invocation Interface (DII). To use `cfobject` with CORBA objects, you must provide the name of the file that contains a string-formatted version of the IOR, or the object's naming context in the naming service; and the object's attributes, method names, and method signatures.

User-defined types (for example, structures) are not supported.

Example

```
<cfobject type = "corba"
    context = "ior"
    class = "c:\\myobject.ior"
    name = "GetName">
```

cfobject: Java or EJB object

Description

Creates and manipulates a Java and Enterprise Java Bean (EJB) object.

Syntax

```
<cfobject
    class = "Java class"
    type = "Java"
    name = "instance name"
    action = "create">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcollection`, `cfexecute`, `cfindex`, `IsInstanceOf`, `cfreport`, `cfsearch`, `cfwddx`; "Using Java objects" on page 938 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`action`	Optional	`create`	Only the default `create` action, which creates the object, is supported.
`class`	Required		The Java class.
`name`	Required		String; name for the instantiated component.
`type`	Required for Java		Object type. Must be `java` for Java and EJB objects.

Usage

To call Java CFXs or Java objects, ColdFusion uses a Java Virtual Machine (JVM) that is embedded in the process. You can configure JVM loading, location, and settings in the ColdFusion Administrator.

Any Java class available in the class path that is specified in the ColdFusion Administrator can be loaded and used from ColdFusion, by using the `cfobject` tag.

Access Java methods and fields

1. Call the `cfobject` tag, to load the class. See the example code.
2. Use the `init` method with appropriate arguments, to call a constructor. For example:

```
<cfset ret = myObj.init(arg1, arg2)>
```

Calling a public method on the object without first calling the `init` method results in an implicit call to the default constructor. Arguments and return values can be any Java type (simple, array, object). ColdFusion makes the conversions if strings are passed as arguments, but not if they are received as return values.

Overloaded methods are supported if the number of arguments is different.

Calling EJBs

To create and call EJB objects, use the `cfobject` tag. In the second example in the following section, the WebLogic JNDI is used to register and find EJBHome instances.

Example

```
<!--- Example of a Java Object, this cfobject call loads the class MyClass
    but does not create an instance object. Static methods and fields
    are accessible after a call to cfobject. --->
<cfobject
    action = "create"
    type = "java"
    class = "myclass"
    name = "myobj">


<!---- Example of an EJB - The cfobject tag creates the Weblogic Environment
    object, which is used to get InitialContext. The context object is
    used to look up the EJBHome interface. The call to Create() results
    in getting an instance of stateless session EJB. --->

<cfobject
    action = "create"
    type = "java"
    class = "weblogic/jndi/Environment"
    name = "wlEnv">

<cfset ctx = wlEnv.getInitialContext()>
<cfset ejbHome = ctx.lookup("statelessSession.TraderHome")>
<cfset trader = ejbHome.Create()>
<cfset value = trader.shareValue(20, 55.45)>
<cfoutput>
     Share value = #value#
</cfoutput>
<cfset value = trader.remove()>
```

cfobject: web service object

Description

Creates a web service proxy object.

Syntax

```
<cfobject
    name = "local name">
    webservice= "service identifier"
    password = "string"
    proxyPassword = "string"
    proxyPort = "port number"
    proxyServer = "URL or IP address"
    proxyUser = "string"
    refreshWSDL = "no|yes"
    type = "webservice"
    username = "string"
    wsdl2javaArgs = "argument string"
    wsportname = "port name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcollection`, `cfexecute`, `cfindex`, `cfreport`, `cfsearch`, `cfwddx`; "Consuming web services" on page 906 in "Using Web Services" on page 902 in the *ColdFusion Developer's Guide*

History

See the History section of the main `cfobject` tag page.

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		Local name for the web service. String.
`webservice`	Required		One of the following: • The absolute URL of the web service. • The name (string) assigned in the ColdFusion Administrator to the web service.
`password`	Optional	Password set in the Administrator, if any	The password to use to access the web service. If the `webservice` attribute specifies a web service name configured in the ColdFusion Administrator, overrides any password specified in the Administrator entry.
`proxyPassword`	Optional	http.proxyPassword system property, if any	The user's password on the proxy server.
`proxyPort`	Optional	http.proxyPort system property, if any.	The port to use on the proxy server.
`proxyServer`	Optional	http.proxyHost system property, if any.	The proxy server required to access the web service URL.
`proxyUser`	Optional	http.proxyUser system property, if any	The user ID to send to the proxy server.

Attribute	Req/Opt	Default	Description
refreshWSDL	Optional	no	• yes: reloads the WSDL file and regenerates the artifacts used to consume the web service • no
type	Optional		The object type. You can omit this attribute or specify webservice.
username	Optional	User name set in the Administrator, if any	The user name to use to access the web service. If the webservice attribute specifies a web service configured name in the ColdFusion Administrator, overrides any user name specified in the Administrator entry.
wsdl2javaArgs	Optional		A string that contains a space-delimited list of arguments to pass to the WSDL2Java tool that generates Java stubs for the web services. Useful arguments include the following: • -W or --noWrapped: turns off the special treatment of wrapped document/literal style operations. • -a or --all: generates code for all elements in the WSDL, even unreferenced ones. • -w or --wrapArrays: prefers building beans to straight arrays for wrapped XML array types. This switch is not in included in the Axis documentation. For detailed information on valid arguments, see the Apache Axis WSDL2Java Reference.
wsportname	Optional	First port in the WSDL	The port name for the web service. This value is case-sensitive and corresponds to the port element's name attribute under the service element. Specify this parameter if the web service contains multiple ports.

Usage

Instantiates a proxy object for a web service. You can enter the absolute URL in this tag, or refer to a web service that is entered in the ColdFusion Administrator. To minimize potential code maintenance, enter the web service in the Administrator, and then refer to that name in this tag.

cfobjectcache

Description

Flushes the query cache.

Category

Database manipulation tags

Syntax

```
<cfobjectcache
    action = "clear">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfobject`

History

ColdFusion 5: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		clear: clears queries from the cache in the Application scope.

cfoutput

Description

Displays output that can contain the results of processing ColdFusion variables and functions. Can loop over the results of a database query.

Category

Data output tags

Syntax

```
<cfoutput
    group = "query column"
    groupCaseSensitive = "yes|no"
    maxRows = "maximum rows to display"
    query = "query name"
    startRow = "start row">
</cfoutput>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcol`, `cfcontent`, `cfdirectory`, `cftable`

History

ColdFusion 4.5.0: Added the `groupCaseSensitive` attribute.

Attributes

Attribute	Req/Opt	Default	Description
`group`	Optional		Query column to use to group sets of records. Eliminates adjacent duplicate rows when data is sorted. Use if you retrieved a record set ordered on one or more a query columns. For example, if a record set is ordered on "`Customer_ID`" in the `cfquery` tag, you can group the output on "`Customer_ID`."
`groupCaseSensitive`	Optional	`yes`	Boolean. Whether to consider the case in grouping rows.
`maxRows`	Optional	Displays all rows	Maximum number of rows to display.
`query`	Optional		Name of `cfquery` from which to draw data for output section.
`startRow`	Optional	1	Row from which to start output.

Usage

In the `cfoutput` tag body, ColdFusion treats text that is surrounded by number signs (#) as a ColdFusion variable or function call. For example, the following code displays the text "Hello World!":

```
<cfset myVar="Hello World!">
<cfoutput>#myVar#</cfoutput>
```

When you specify a `query` attribute, this tag loops over the query rows and produces output for each row within the range specified by the `startRow` and `maxRows` values, and groups or eliminates duplicate entries as specified by the grouping attribute values, if any. It also sets the *query*.currentRow variable to the current row being processed.

If you nest `cfoutput` blocks that process a query, you specify the `query` and `group` attributes at the top-most level; you can specify a `group` attribute for each inner block except the innermost `cfoutput` block.

This tag requires an end tag.

Example

```
<!--- EXAMPLE: This example shows how cfoutput operates. --->
<!--- Run a sample query. --->
<cfquery name = "GetCourses" dataSource = "cfdocexamples">
    SELECT Dept_ID, CorName, CorLevel
    FROM courseList
    ORDER by Dept_ID, CorLevel, CorName
</cfquery>
<h3>cfoutput Example</h3>
<p>cfoutput tells ColdFusion Server to begin processing, and then to hand back control of
page rendering to the web server.
<p>For example, to show today's date, you could write #DateFormat("#Now()#"). If you enclosed
that expression in cfoutput, the result would be<cfoutput>#DateFormat(Now())#</cfoutput>.

<p>In addition, cfoutput may be used to show the results of a query operation, or only a
partial result, as shown:

<p>There are <cfoutput>#getCourses.recordCount#</cfoutput> total records in our query. Using
the maxRows parameter, we are limiting our display to 4 rows.
<p><cfoutput query = "GetCourses" maxRows = 4>
    #Dept_ID# #CorName# #CorLevel#<br>
</cfoutput>

<p>EXAMPLE: The next example uses the group attribute to eliminate duplicate lines from a
    list of course levels taught in each department.</p>
<p><cfquery name = "GetCourses" dataSource = "cfdocexamples"></p>
    SELECT Dept_ID, CorLevel
    FROM courseList
    ORDER by Dept_ID, CorLevel
</cfquery>
<p><cfoutput query = "GetCourses" group="CorLevel" GroupCaseSensitive="True">
    #Dept_ID# #CorLevel#<br></p>
</cfoutput>

<p>cfoutput can also show the results of a more complex expression,
such as getting the day of the week from today's date. We first
extract the integer representing the Day of the Week from
the server function Now() and then apply the result to
the DayofWeekAsString function:</p>

<br>Today is #DayofWeekAsString(DayofWeek(Now()))#
<br>Today is <cfoutput>#DayofWeekAsString(DayofWeek(Now()))#</cfoutput>

<p> EXAMPLE: This last example shows nested cfoutput tags:</p>
<cfquery datasource="cfdocexamples" name="empSalary">
    SELECT Emp_ID, firstname, lastname, e.dept_id, salary, d.dept_name
    FROM employee e, departmt d
    WHERE e.dept_id = d.dept_id
    ORDER BY d.dept_name
</cfquery>

<!--- Outer cfoutput. --->
<cfoutput query="empSalary" group="dept_id">
 <h2>#dept_name#</h2>
 <table width="95%" border="2" cellspacing="2" cellpadding="2" >
```

```
<tr>
 <th>Employee</th>
 <th>Salary</th>
</tr>
<cfset deptTotal = 0 >
<!--- Inner cfoutput. --->
<cfoutput>
    <tr>
<td>#empSalary.lastname#, #empSalary.firstname#</td>
<td align="right">#DollarFormat(empSalary.salary)#</td>
</tr>
    <cfset deptTotal = deptTotal + empSalary.salary>
</cfoutput>
    <tr>
<td align="right">Total</td>
    <td align="right">#DollarFormat(deptTotal)#</td>
</tr>
 <cfset deptTotal = 0>
 </table>
</cfoutput>
```

cfparam

Description

Tests for the existence of a parameter (that is, a variable), validates its data, and, if a default value is not assigned, optionally provides one.

History

ColdFusion MX 7:

- Added `min`, `max`, and `pattern` attributes.
- Added `creditcard`, `email`, `eurodate`, `float`, `integer`, `range`, `regex`, `regular_expression`, `ssn`, `social_security_number`, `time`, `URL`, `USdate`, `XML`, and `zipcode` values of the `type` attribute.

Category

Variable manipulation tags

Syntax

```
<cfparam
    name = "parameter name"
    default = "value"
    max = "value"
    min = "value"
    pattern = "regular expression"
    type = "data_type">
```

See also

`cfcookie`, `cfregistry`, `cfsavecontent`, `cfschedule`, `cfset`; "Validating data with the IsValid function and the cfparam tag" on page 573 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		Name of the parameter (variable) to test (such as "Client.Email " or "Cookie.BackgroundColor "). If omitted, and if the parameter does not exist, an error is thrown.
`default`	Optional		Value to set parameter to if it does not exist. Any expression used for the default attribute is evaluated, even if the parameter exists. The result is not assigned if the parameter exists, but if the expression has side effects, they still occur.
`max`	Optional		The maximum valid value; used only for `range` validation.
`min`	Optional		The minimum valid value; used only for `range` validation.
`pattern`	Optional		A JavaScript regular expression that the parameter must match; used only for `regex` or `regular_expression` validation.

Attribute	Req/Opt	Default	Description
type	Optional	`any`	The valid format for the data; one of the following. For detailed information on validation algorithms, see "Validating form data using hidden fields" on page 566 in "Validating Data" on page 554 in the *ColdFusion Developer's Guide*. • `any`: any type of value. • `array`: an array of values. • `binary`: a binary value. • `boolean`: a Boolean value: yes, no, true, false, or a number. • `creditcard`: a 13-16 digit number conforming to the mod10 algorithm. • `date` or `time`: a date-time value. • `email`: a valid e-mail address. • `eurodate`: a date-time value. Any date part must be in the format dd/mm/yy. The format can use /, -, or . characters as delimiters. • `float` or `numeric`: a numeric value. • `guid`: a Universally Unique Identifier of the form "`XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX`" where 'X' is a hexadecimal number. • `integer`: an integer. • `query`: a query object. • `range`: a numeric range, specified by the `min` and `max` attributes. • `regex` or `regular_expression`: matches input against `pattern` attribute. • `ssn` or `social_security_number`: a U.S. social security number. • `string`: a string value or single character. • `struct`: a structure. • `telephone`: a standard U.S. telephone number. • `URL`: an http, https, ftp, file, mailto, or news URL. • `UUID`: a ColdFusion Universally Unique Identifier, formatted '`XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXXXXXX`', where 'X' is a hexadecimal number. See "CreateUUID" on page 739. • `USdate`: a U.S. date of the format mm/dd/yy, with 1-2 digit days and months, 1-4 digit years. • `variableName`: a string formatted according to ColdFusion variable naming conventions. • `xml`: XML objects and XML strings. • zipcode: U.S., 5- or 9-digit format ZIP codes.

Usage

You can use this tag to make the following tests:

- To test whether a required variable exists, use this tag with only the `name` attribute. If it does not exist, ColdFusion MX stops processing the page and returns an error.
- To test whether a required variable exists, and that it is of the specified type, use this tag with the `name` and `type` attributes. If the variable does not exist or its value is not of the specified type, ColdFusion returns an error.
- To set a default value for the variable, use this tag with the `name` and `default` attributes. If the variable does not exist, it is created and set to the `default` attribute value. If the variable exists, processing continues; the value is not changed.

If you specify `variableName` for the `type` attribute, the parameter's value must be a string that is in ColdFusion variable name format; that is, starts with a letter, underscore (_), or Unicode currency symbol, and contains letters, numbers, underscores, periods, and Unicode currency symbols, only. ColdFusion does not check whether the parameter value corresponds to an existing ColdFusion variable.

To improve performance, avoid using the `cfparam` tag in ColdFusion functions, including in CFC methods. Instead, place the `cfparam` tags in the body of the CFML pages.

Example

```
<!--- This example shows how to use CFPARAM to define default values for page variables. --->
<cfparam name = "storeTempVar" default = "my default value">
<cfparam name = "tempVar" default = "my default value">

<!--- Check if form.tempVar was passed. --->
<cfif IsDefined("form.tempVar") is "True">
    <!--- Check if form.tempVar is not blank. --->
    <cfif form.tempVar is not "">
        <!--- If not, set tempVar to value of form.tempVar --->
        <cfset tempVar = form.tempVar>
    </cfif>
</cfif>

<body>
<h3>cfparam Example</h3>
<p>cfparam is used to set default values so that a developer does not have to
check for the existence of a variable using a function like IsDefined.</p>

<p>The default value of our tempVar is "<cfoutput>#StoreTempVar# </cfoutput>"</p>

<!--- Check if tempVar is still the same as StoreTempVar and that tempVar is not blank. --->
<cfif tempVar is not #StoreTempVar#
    and tempVar is not "">
        <h3>The value of tempVar has changed: the new value is
        <cfoutput>#tempVar#</cfoutput></h3>
</cfif>

<p>
<form action = "cfparam.cfm" method = "post">
    Type in a new value for tempVar, and hit submit:<br>
    <input type = "Text" name = "tempVar">
    <input type = "Submit" name = "" value = "submit">
</form>
```

cfpdf

Description

Manipulates existing PDF documents. The following list describes some of the tasks you can perform with the `cfpdf` tag:

- Merge several PDF documents into one PDF document.
- Delete pages from a PDF document.
- Merge pages from one or more PDF documents and generate a new PDF document.
- Linearize PDF documents for faster web display.
- Remove interactivity from forms created in Acrobat® to generate flat PDF documents.
- Encrypt and add password protection to PDF documents.
- Generate thumbnail images from PDF documents or pages.
- Add or remove watermarks from PDF documents or pages.
- Retrieve information associated with a PDF document, such as the software used to generate the file or the author, and set information for a PDF document, such as the title, author and keywords.

History

ColdFusion 8: Added this tag.

Category

Data output tags

Syntax

```
Add a watermark to a PDF document
<cfpdf
    required
    action = "addwatermark"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
        one of the following:
    copyfrom = "absolute or relative pathname to a PDF file from which the
        first page is used as a watermark"
    image = "absolute or relative pathname to image file|image variable used as a watermark"
    optional
    foreground = "yes|no"
    isBase64 = "yes|no"
    opacity = "watermark opacity"
    overwrite = "yes|no"
    pages = "page or pages to add the watermark"
    password = "user or owner password for the PDF source file"
    position = "position on the page where the watermark is placed"
    rotation = "degree of rotation of the watermark"
    showonprint = "yes|no">
        one of the following:
    destination = "PDF output file pathname"
    name = "PDF document variable name"

Delete pages from a PDF document
<cfpdf
    required
    action = "deletepages"
```

```
    pages = "page or pages to delete"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
    optional
    overwrite = "yes|no"
    password = "PDF source file password"
        one of the following:
    destination = "PDF output file pathname"
    name = "PDF document variable name">

Retrieve information about a PDF document
<cfpdf
    required
    action = "getinfo"
    name = "structure variable name"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
    optional
    password = "PDF source file password">

Merge PDF documents into an output PDF file
<cfpdf
    required
    action = "merge"
        one of the following:
    directory = "directory of PDF files to merge"
    source = "comma-separated list of PDF source files|absolute or relative pathname
        to a PDF file|PDF document variable|cfdocument variable"
    <cfpdfparam ...>
        required if directory is specified:
    order = "name|time"
        one of the following if <cfpdfparam ...> is specified:
    name = "PDF document variable name"
    destination = "PDF output file pathname"
    optional
    ascending = "yes|no"
    keepBookmark = "yes|no"
    overwrite = "yes|no"
    pages = "pages to merge in PDF source file"
    password = "PDF source file password"
    stopOnError = "yes|no"
        one of the following:
    destination = "PDF output file pathname"
        name = "PDF document variable name">

Use DDX instructions to manipulate PDF documents
<cfpdf
    required
    ddxfile = "DDX filepath|DDX string"
    inputfiles = "#inputStruct#"
    outputfiles = "#outputStruct#"
    name = "structure name">
    optional
    action="processddx"

Set passwords and encrypt PDF documnets
<cfpdf
    required
    action = "protect"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
```

```
        at least one of the following:
    newUserPassword = "password"
    newOwnerPassword = "password"
        if newOwnerPassword is specified:
    permissions =
"All|AllowAssembly|AllowDegradedPrinting|AllowCopy|AllowFillIn|AllowModifyAnnotations|
        AllowModifyContents|AllowPrinting|AllowScreenReaders|AllowSecure|None|
        comma-separated list"
    optional
    destination = "PDF output file pathname"
    encrypt = "RC4_40|RC4_128|RC4_128M|AES_128|none"
    overwrite = "yes|no"
    password = "source file password">

Name a PDF document variable
<cfpdf
    required
    action = "read"
    name = "PDF document variable name"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
    optional
    password = "PDF source file password">

Remove a watermark from a PDF document
<cfpdf
    required
    action = "removeWatermark"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
    optional
    overwrite = "yes|no"
    pages = "page or pages from which to remove the watermark"
    password = "PDF source file password">
        one of the following:
    destination = "PDF output file pathname"
    name = "PDF document variable name"

Set information about a PDF document
<cfpdf
    required
    action = "setinfo"
    info = "#structure variable name#"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
    optional
    destination = "PDF output file pathname"
    overwrite = "yes|no"
    password = "PDF source file password">

Generate thumbnails from pages in a PDF document
<cfpdf
    required
    action = "thumbnail"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
    optional
    destination = "directory path where the thumbnail images are written"
    format = "png|jpeg|tiff"
    imagePrefix = "string used as a prefix in the output filename"
    overwrite = "yes|no"
```

```
    password = "PDF source file password">
    pages = "page or pages to make into thumbnails"
    resolution= "low|high"
    scale = "percentage between 1 and 100"
    transparent = "yes|no">

Write a PDF document to an output file
<cfpdf
    required
    action = "write"
    destination = "PDF output file pathname"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
    optional
    flatten = "yes|no"
    overwrite = "yes|no"
    password = "PDF source file password"
    saveOption = "linear|incremental|full"
    version = "1.1|1.2|1.3|1.4|1.5|1.6">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdocument`, `cfdocumentsection`, `cfpdfform`, `cfpdfformparamUsage`, `cfpdfparam`, `cfpdfsubform`, `cfprint`, `IsDDX`, `IsPDFFile`, `IsPDFObject`, "Assembling PDF Documents" on page 741 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Action	Req/Opt	Default	Description
`action`	N/A	Optional	*processddx*	Action to take: • `addWa termark` • `deletePages` • `getInfo` • `merge` • `processddx` • `protect` • `read` • `removeWatermark` • `setInfo` • `thumbnail` • `write`
`ascending`	`merge`	Optional	`no`	Order in which the PDF files are sorted: • `yes`: Files are sorted in ascending order • `no`: Files are sorted in descending order Applicable only when you specify the `directory` attribute.
`copyFrom`	`addWatermark`	Optional		Pathname of the PDF document from which to use the first page as a watermark
`ddxfile`	`processddx`	Required		Pathname of the DDX file, or a string with DDX instructions

Attribute	Action	Req/Opt	Default	Description
destination	addWatermark deletePages merge protect removeWatermark setInfo thumbnail write	Required for the write action Optional for all other actions		Pathname of the modified PDF document. If the destination file exists, you must set the overwrite attribute to yes. If the destination file does not exist, ColdFusion creates the file, if the parent directory exists. You can specify the destination attribute or the name attribute, but not both. For the thumbnail action, the destination is the directory path where the images are written. If you specify a relative pathname to the destination directory, the destination directory is relative to the template directory. If you do not specify a destination directory, ColdFusion creates a directory called thumbnails in the directory in the template directory.
directory	merge	Optional		Directory of the PDF documents to merge. You must specify either the directory attribute or the source attribute. If you specify the directory attribute, ColdFusion orders the documents by filename in descending order, by default. To change the order of the files, use the order attribute.
encrypt	protect	Optional	RC4_128 (Acrobat 5.0 or higher)	Encryption type for the PDF output file: • RC4_40 • RC4_128 • RC4_128M • AES_128 • None For more information, see "Encryption for PDF documents" on page 445.
flatten	write	Optional	no	Applies to forms created in Acrobat only (not forms created in LiveCycle); specifies whether interactivity is turned off: • yes: the form fields are no longer interactive. • no: the form fields remain interactive.
foreground	addWatermark	Optional	no	Placement of the watermark on the page: • yes: the watermark appears in the foreground (over the page content). • no: the watermark appears in the background (behind the page content).
format	thumbnail	Optional	jpg	File type of thumbnail image output: • jpg • tiff • png
image	addWatermark	Optional		Image used as a watermark. You can specify a pathname, a variable that contains an image file, or a ColdFusion image variable.

Attribute	Action	Req/Opt	Default	Description
imagePrefix	thumbnail	Optional	If the source is a pathname, the filename is used as the prefix; otherwise thumbnail is the prefix	Prefix used for each image thumbnail file generated. The image filenames use the format: *imagePrefix*_page_*n.format*. For example, the thumbnail for page 1 of a document with the imagePrefix attribute set to myThumbnail is myThumbnail_page_1.jpg.
info	setInfo	Required		Structure variable for relevant information, for example, "#infoStruct#". You can specify the Author, Subject, Title, and Keywords for the PDF output file.
inputFiles	processddx	Required		Structure that maps the PDF source files to the input variables in the DDX file, or a string of elements and their pathname.
isBase64	addWatermark	Optional	no	Valid only when the image attribute is specified. Specifies whether the image used as a watermark is in Base64 format: • yes: the image is in Base64 format. • no: the image is not in Base64 format.
keepBookmark	merge	Optional	no	Specifies whether bookmarks from the source PDF documents are retained in the merged document: • yes: the bookmarks are retained. • no: the bookmarks are removed.
name	addWatermark deletePages getInfo merge processddx protect read removeWatemark write	Required: getInfo processddx read Optional: addWatermark deletePages merge protect removeWatermark write		PDF document variable name, for example, *myBook*. If the source is a PDF document variable, you cannot specify the name attribute again; you can write the modified PDF document to the destination. You can specify the destination attribute or the name attribute, but not both. For the processddx action, the name represents the structure that is populated with the success or failure of the output variables.
newOwnerPassword	protect	Optional (see Description)		Password used to set permissions on a PDF document. To change the default permissions, you must specify the newOwnerPassword attribute. For more information, see "PDF document passwords" on page 444.
newUserPassword	protect	Optional (see Description)		Password used to open PDF document. You must specify either the newUserPassword attribute or a newOwnerPassword attribute; if you specify both, the passwords must differ. For more information, see "PDF document passwords" on page 444.

Attribute	Action	Req/Opt	Default	Description
opacity	addWatermark	Optional	3	Opacity of the watermark. Valid values are integers in the range 0 (transparent) through 10 (opaque).
order	merge	Optional	time	Order in which the PDF documents in the directory are merged: • name: orders the documents alphabetically by filename. • time: orders the documents by timestamp. By default, ColdFusion merges the files in descending order (for example, from Z to A). To change this, set the ascending attribute to yes.
outputFiles	processddx	Required		Structure that contains the output files in the DDX file or string as keys and the pathname to the result file as the value.
overwrite	addWatermark deletePages merge protect removeWatermark setInfo thumbnail write	Optional	no	Specifies whether PDF output overwrites the destination file: • yes: overwrites the destination file. • no: does not overwrite the destination file. For the thumbnail action, specifies whether to overwrite the destination directory. If the directory exists, the thumbnails are not generated unless overwrite is set to yes.
pages	addWatermark deletePages merge removeWatermark	Required: deletePages Optional: addWatermark merge removeWatermark thumbnail	all	Page or pages in the source PDF document on which to perform the action. You can specify multiple pages and page ranges as follows: "1,6—9,56—89,100, 110—120". For the removeWatermark action, the pages attribute applies only to the watermark type. ColdFusion ignores duplicate pages and numbers greater than the total page count.
password	addWatermark deletePages getInfo merge protect read removeWatermark setInfo thumbnail write	Optional		Owner or user password of the source PDF document, if the document is password-protected.

Attribute	Action	Req/Opt	Default	Description
permissions	protect	Optional	All	Type of permissions on the PDF document: • All • AllowAssembly • AllowCopy • AllowDegradedPrinting • AllowFillIn • AllowModifyAnnotations • AllowModifyContents • AllowPrinting • AllowScreenReaders • AllowSecure • None Except for All or None, you can specify a comma-separated list of permissions. To set permissions, you must also set the newOwnerPassword attribute.
position	addWatermark	Optional		Position on the page where the watermark is placed. The position represents the top-left corner of the watermark. Specify the x and y coordinates; for example "50,30".
resolution	thumbnail	Optional	high	Image quality used to generate thumbnail images: • high: use high resolution (uses more memory). • low: use low resolution.
rotation	addWatermark	Optional		Degree of rotation of the watermark image on the page, for example, "30".
saveOption	write	Optional	full	Save options for the PDF output: • full : normal save (default) • incremental: required to save modifications to a signed PDF document. • linear: for faster display.
scale	thumbnail	Optional	25	Size of the thumbnail relative to the source page. The value represents a percentage from 1 through 100.
showOnPrint	addWatermark	Optional	no	Specify whether to print the watermark with the PDF document: • yes: the watermark is printed with the PDF document. • no: the watermark is display-only.

Attribute	Action	Req/Opt	Default	Description
source	addWatermark deletePages getInfo merge protect read removeWatermark setInfo thumbnail write	Required (see Usage section for merge)		PDF document used as the source. The source can be one of the following: • An absolute or relative pathname to a PDF document, for example, c:\work\myPDF.pdf or myPDF.pdf. • A PDF document variable in memory that is generated by the cfdocument tag or the cfpdf tag, for example, "myPDFdoc".
stopOnError	merge	Optional	no	Valid only if the directory attribute is specified. If the specified directory contains files other then ColdFusion-readable PDF files, ColdFusion either stops merge process or continues. • yes: stops the merge process if invalid PDF files exist in the specified directory. • no: continues the merge process even if invalid files exist in the specified directory.
transparent	thumbnail	Optional	no	(format="png" only) Specifies whether the image background is transparent or opaque: • yes: the background is transparent. • no: the background is opaque.
version	write	Optional		Version of the PDF used to write the document: • 1.1 • 1.2 • 1.3 • 1.4 • 1.5 • 1.6 For more information, see "PDF versions" on page 447.

Note: *To modify the PDF source document, specify the same file pathname for the* `source` *and* `destination` *attributes, and set the* `overwrite` *attribute to* `yes`*.*

Usage

You use the `cfpdf` tag to manipulate and assemble existing PDF documents. Although the `cfpdf` tag provides much of the functionality available in Acrobat, you cannot use this tag to generate a PDF document from another file format. To create PDF output from HTML and CFML content, use the `cfdocument` tag.

You cannot embed a `cfpdf` tag within a `cfdocument` tag or embed a `cfdocument` tag within a `cfdpdf` tag; however, you can write the output of a `cfdocument` tag to a variable and pass the variable to the `cfpdf` tag. The following example shows how to use the `cfdocument` tag to create a cover page and add it to a merged PDF document:

```
<!--- Use the cfdocument tag to create a cover page and write the output to a variable called
    cfdoc. --->
<cfdocument format="PDF" name="cfdoc">
```

```
<html>
<body>
<h1>Here is a cover page</h1>
</body>
</html>
</cfdocument>

<!--- Use the cfpdf tag and cfpdfparam tags to merge individual PDF documents into a new PDF
document called new.pdf. Notice that the cfdoc variable created by using the cfdocument tag
is the source value of the first cfpdfparam tag. --->
<cfpdf action="merge" destination="/samtemp/pdfs/new.pdf" overwrite="yes">
    <cfpdfparam source="cfdoc">
    <cfpdfparam source="/samtemp/pdfs/pdf2.pdf">
    <cfpdfparam source="/samtemp/pdfs/pdf1.pdf">
</cfpdf>
```

You can use the `cfpdf` tag to assemble interactive PDF form files into a single PDF document and flatten forms created in Acrobat (by using the `flatten` attribute with the `write` action); however, to process PDF form data, use the `cfpdfform` and related tags. You cannot use the `cfpdf` tag to flatten forms created in Adobe LiveCycle Designer ES.

Reading and writing PDF files

The `cfpdf` tag provides several options for reading and writing PDF files. You can specify a PDF variable or a PDF file as the source, and you can write the output to a variable or to a file (but not both). The following table explains the read and write operations:

Task	Attributes	Example
Overwrite a source PDF file	Specify the PDF file pathname as the `source` and do not specify a `destination`.	`<cfpdf action="addWatermark" source="myPDF" image="myImage.jpg">`
Write a PDF document in memory to a file	Specify the PDF variable as the `source` and a PDF file pathname for the `destination`.	`<cfpdf action="addWatermark" source="myPDF" image="myImage.jpg" destination="outputFile.pdf">`
Write a PDF document to a new file	Specify a PDF file pathname as the `source` and a different PDF file pathname as the `destination`.	`<cfpdf action="addWatermark" source="sourceFile.pdf" image="myImage.jpg" destination="outputFile.pdf">`
Write a PDF file to a PDF variable	Specify the PDF file pathname as the `source` and a PDF variable `name`.	`<cfpdf action="addWatermark" source="sourceFile.pdf" image="myImage.jpg" name="myPDF">`
Overwrite a PDF document in memory	Specify the PDF variable name as the `source` and do not specify a destination.	`<cfpdf action="addWatermark" source="myPDF" image="myImage.jpg">`

Working with PDF files in memory

ColdFusion gives you the option to write a PDF file to a variable by using the `name` attribute, which is useful if you want to perform multiple operations on a document before writing it to a file. However, this is practical for small files only because of memory requirements. If you are working with large PDF documents, write the PDF documents to files.

ColdFusion recommends that you do not specify the `name` attribute when you specify a variable as the source because it creates a copy, which increases processing. In most cases, this is unnecessary because you can reuse variables even after you write them to files.

Note: *When you use PDF variables within a try/catch block and ColdFusion generates an error, the variables are unusable after the error is generated.*

Printing PDF documents

Use the cfprint tag to print PDF documents. Markups, such as sticky notes, comments, and editorial revisions, are not printed with the document.

addWatermark action Use the addwatermark action to add a watermark to specified pages in a PDF document. You can add a watermark in one of the following ways:

- Use the first page of another PDF document as a watermark. ColdFusion overlays the copyfrom page on the source document, without enlarging the image.
- Specify an image file to use as a watermark.
- Specify an image in memory by using an image variable.

The following code shows how to use the first page of a PDF document as a watermark:

```
<cfpdf action="addWatermark" source="c:\myBook.pdf" copyFrom="e:\yourBook.pdf"
    destination="ourBook.pdf" overwrite="yes">
```

By default, ColdFusion applies the watermark to all of the pages in the output file, with the watermark image centered on the page. The following code applies a JPEG image as a watermark to the first page of the output file:

```
<cfpdf action="addWatermark" source="Book.pdf"
    image="../cfdocs/images/artgallery/paul01.jpg" destination="newBook.pdf" pages="1"
    overwrite="yes">
```

To specify a ColdFusion image as a watermark, use the cfimage tag or Image functions. The following example converts an image to grayscale and applies it as a watermark to a PDF file:

```
<!--- Use the ImageNew function to create a ColdFusion image from a JPEG file. --->
<cfset myImage=ImageNew("../cfdocs/images/artgallery/jeff05.jpg")>

<!--- Use the ImageGrayscale function to convert the image to grayscale in memory. --->
<cfset ImageGrayscale(myImage)>

<!--- Specify the image variable to apply the grayscale image as a watermark in the Book.pdf
file. Because the source and destination are the same and the overwrite attribute is set to
yes, ColdFusion overwrites the source file. --->
<cfpdf action="addWatermark" source="Book.pdf" destination="Book.pdf" overwrite="yes"
image="#myImage#">
```

For more information on ColdFusion images, see “Creating and Manipulating ColdFusion Images” on page 765 in the *ColdFusion Developer’s Guide*.

deletePages action Use the deletePages action to remove pages from a specified PDF document. You can specify a single page, a page range, or a comma-separated list of pages, as the following code shows:

```
<cfpdf action="deletePages" source="c:\myBook.pdf" pages="1,16-32,89,100-147"
    destination="myLittleBook.pdf">
```

getInfo action Use the getInfo action to extract information associated with the PDF document, such as the author, title, and creation date. You specify the name of the structure variable that contains the relevant data associated with the file, as the following code shows:

```
<cfpdf action="getInfo" source="myBook.pdf" name="PDFInfo">
<p><cfoutput>#PDFInfo.title#</cfoutput></p>
<p><cfoutput>#PDFInfo.author#</cfoutput></p>
<p><cfoutput>#PDFInfo.keywords#</cfoutput></p>
<p><cfoutput>#PDFInfo.created#</cfoutput></p>
```

For a complete list of information elements, use the cfdump tag, as the following code shows:

```
<cfdump var="#PDFInfo#">
```

Note: *To view the permissions for a PDF document that is password-protected, specify the user password, not the owner password. If you specify the owner password, all permissions are set to* `Allowed`*.*

PDF file information elements

The following table describes the information elements you can retrieve with the `getinfo` action:

Element	Example	Description
Application	Acrobat PDFMaker 7.0.7 for Word	Application used to create the PDF document. This value is read-only.
Author	Harper Lee	Author of the PDF document. You can specify a text string with the `setInfo` action.
CenterWindowOnScreen	[empty string]	Display setting for initial view of the PDF document. To change this setting, use the `processddx` action with the `InitialViewProfile` DDX element.
ChangingDocument	Not Allowed	Permissions assigned for editing the PDF content. To change this setting, use the `permissions` attribute with the `protect` action.
Commenting	Allowed	Permissions assigned for adding comments to the PDF document. To change this setting, use the `permissions` attribute with the `protect` action.
ContentExtraction	Allowed	Permissions assigned for extracting content from the PDF document. To change this setting, use the `permissions` attribute with the `protect` action.
CopyContent	Allowed	Permissions assigned for copying content from the PDF document. To change this setting, use the `permissions` attribute with the `protect` action.
Created	D:20061121155226-05'00'	System-generated creation date of the PDF document. You can specify a text string with the `setInfo` action.
DocumentAssembly	Not Allowed	Permissions assigned for merging the PDF document with other PDF documents. To change this setting, use the `permissions` attribute with the `protect` action.
Encryption	Password Security	Specifies whether the PDF file is password-protected. To change the encryption algorithm, or add a password, use the `protect` action.
FilePath	C:\ColdFusion\wwwroot\lion\myDoc.pdf	Absolute pathname for the PDF file. This value is read-only.
FillingForm	Allowed	Permissions assigned for entering data in form fields. To change this setting, use the `permissions` attribute with the `protect` action.
FitToWindow	[empty string]	Display setting for initial view of the PDF document. To change this setting use the `processddx` action with the `InitialViewProfile` DDX element.
HideMenubar	[empty string]	Display setting for initial view of the PDF document. To change this setting, use the `processddx` action with the `InitialViewProfile` DDX element.
HideToolbar	[empty string]	Display setting for initial view of the PDF document. To change this setting, use the `processddx` action with the `InitialViewProfile` DDX element.
HideWindowUI	[empty string]	Display setting for initial view of the PDF document. To change this setting, use the `processddx` action with the `InitialViewProfile` DDX element.

Element	Example	Description
Keywords	marketing,sales,production	Keywords specified for searches in the PDF document. You can specify a comma-separated list of keywords with the `setInfo` action.
Language	EN-US	Language version used to create the source file for the PDF document. This value is read-only.
Modified	D:20061121155226-06'00'	System-generated timestamp for when the PDF file was last modified. You can specify a text string with the `setInfo` action
PageLayout	OneColumn	Display setting for the initial view of the PDF document. To change this setting, use the `processddx` action with the `InitialViewProfile` DDX element.
Printing	Allowed	Permissions assigned for printing the document. To change this setting, use the `permissions` attribute with the `protect` action.
Producer	Acrobat Distiller 7.0.5 (Windows)	Version of Acrobat Distiller used to generate the PDF document. This value is read-only.
Properties	[empty string]	This value is read-only.
Secure	Not Allowed	Display setting that shows whether the PDF document is password protected.
ShowDocumentsOption	[empty string]	Display setting for initial view of the PDF document. To change this setting, use the `processddx` action with the `InitialViewProfile` DDX element.
ShowWindowsOption	[empty string]	Display setting for initial view of the PDF document. To change this setting, use the `processddx` action with the `InitialViewProfile` DDX element.
Signing	Allowed	Permissions for allowing electronic signatures to the PDF document. To change this setting, use the `permissions` attribute with the `protect` action.
Subject	Product Marketing	The subject assigned to the PDF document. You can specify a text string with the `setInfo` action.
Title	Chapter 1: Getting Started	The title assigned to the PDF document. You can specify a text string with the `setInfo` action.
TotalPages	25	Total pages in the PDF document. This value is read-only.
Trapped	[empty string]	Indicates whether trapping is applied to the PDF document. Trapping is used in printing to eliminate gaps between two adjoining ink colors. You can specify a text string with the `setInfo` action.
Version	1.6	Version of the Adobe PDF generator used to create the PDF document. To change this setting use the `version` attribute with the `write` action. For more information, see "PDF versions" on page 447.

merge action Use the `merge` action to assemble PDF documents or pages from PDF source files into one output file. The following code shows how to merge all the PDF files in a directory:

```
<cfpdf action="merge" directory="c:\myPDFfiles" destination="oneBigFile.pdf"
    overwrite="yes">
```

By default, ColdFusion adds the files in descending order by timestamp. The following code merges the source files in ascending order by filename:

```
<cfpdf action="merge" directory="c:\book" order="name" ascending="yes"
    destination="c:\book\output1.pdf" overwrite="yes">
```

This is useful if the source files have logical names, such as Chap0.pdf, Chap1.pdf, Chap2.pdf, and so on.

By default, ColdFusion continues the merge process even if it encounters a file in the specified directory that is not a valid PDF document. To stop the merge process if the directory contains files other than valid PDF documents, set the `stopOnError` attribute to `yes`:

```
<cfpdf action="merge" directory="c:\bookfiles" destination="book.pdf" overwrite="yes"
    order="name" ascending="yes" keepBookmark="yes" stopOnError="yes">
```

To create a PDF file from specific pages in a document, use the `source` attribute with the `pages` attribute. The following code creates a file from pages 1–5 of the source document:

```
<cfpdf action="merge" source="myBigBook.pdf" pages="1-5" destination="myShortBook.pdf"
    overwrite="yes">
```

To merge several files into one document, specify the absolute pathnames of the files in a comma-separated list, as the following code shows:

```
<cfpdf action="merge" source="c:\PDFdocs\myBook\Chap1.pdf,
    c:\PDFdocs\myBook\Chap2.pdf,c:\PDFdocs\myBook\Chap3,pdf" destination="myBook.pdf"
    overwrite="yes">
```

For more control over the order of files, to assemble files in different locations, and to extract pages from multiple PDF files, use the `cfpdfparam` tag with the `merge` action. For more information on merging PDF files, see "Assembling PDF Documents" on page 741 in the *ColdFusion Developer's Guide*.

processddx action Use the `processddx` action to assemble PDF files by processing Document Description XML (DDX) instructions. DDX is a declarative markup language used by Adobe® LiveCycle® Assembler. You can use DDX instructions to perform advanced tasks, such as adding table of contents pages, headers and footers, automatic page numbers, and text-string watermarks to PDF documents.

ColdFusion provides a subset of LiveCycle Assembler functionality. To determine whether you can perform the your tasks in ColdFusion or whether you have to purchase LiveCycle Assembler, see the tables in the following sections.

For complete DDX syntax, see the *Adobe LiveCycle Assembler Document Description XML Reference*.

Supported DDX elements

The following table lists the DDX elements that ColdFusion supports:

About	Author	Background
Center	DatePattern	DDX
DocumentInformation	DocumentText	Footer
Header	InitialViewProfile	Keyword
Keywords	Left	MasterPassword
Metadata	NoBookmarks	OpenPassword
PageLabel	Password	PasswordAccessProfile
PasswordEncryptionProfile	PDF (see Note)	PDFGroup
Permissions	Right	StyledText
StyleProfile	Subject	TableOfContents
TableOfContentsEntryPattern	TableOfContentsPagePattern	Title
Watermark		

Note: *ColdFusion does not support the* `certification` *and* `mergeLayers` *attributes of the* `PDF` *element.*

Restricted DDX elements

The following table lists the DDX elements that ColdFusion excludes:

ArtBox	AttachmentAppearance	Bookmarks
BlankPage	BleedBox	Comments
Description	FileAttachments	FilenameEncoding
LinkAlias	Links	NoBackgrounds
NoComments	NoFileAttchments	NoFooters
NoForms	NoHeaders	NoLinks
NoPageLabels	NoThumbnails	NoWatermarks
NoXFA	PageMargins	PageSize
PageRotation	PageOverlay	PageUnderlay
PDFsFromBookmarks	Transform	TrimBox

Simple DDX instructions

You can create DDX instructions in any text editor and save the file with a DDX extension. The following example shows the DDX instructions for merging several documents and generating a table of contents with bookmarks from the source PDF documents:

```
<?xml version="1.0" encoding="UTF-8"?>
<DDX xmlns="http://ns.adobe.com/DDX/1.0/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ns.adobe.com/DDX/1.0/ coldfusion_ddx.xsd">
<PDF result="Out1">
    <PDF source="Title"/>
    <TableOfContents/>
    <PDF source="Doc1"/>
    <PDF source="Doc2"/>
    <PDF source="Doc3"/>
</PDF>
</DDX>
```

Processing DDX instructions in ColdFusion

The following code processes the DDX instructions in ColdFusion:

```
<!--- The following code verifies that the DDX file exists and the DDX instructions are
valid. --->
<cfif IsDDX("Book.ddx")>

    <!--- The following code maps the PDF source files to the PDF source variables in the
    DDX file. --->
    <cfset inputStruct=StructNew()>
    <cfset inputStruct.Title="Title.pdf">
    <cfset inputStruct.Doc1="Chap1.pdf">
    <cfset inputStruct.Doc2="Chap2.pdf">
    <cfset inputStruct.Doc3="Chap3.pdf">

    <!--- The following code maps the PDF output file to the PDF result variable in the DDX
    file. --->
    <cfset outputStruct=StructNew()>
```

```
    <cfset outputStruct.Out1="output.pdf">

    <!--- The following code process the DDX instructions in the Book.ddx file to generate
    a merged document. --->
    <cfpdf action="processddx" ddxfile="Book.ddx" inputfiles="#inputStruct#"
        outputfiles="#outputStruct#" name="ddxVar">
<cfelse>
    <p>The DDX instructions are not valid.</p>
</cfif>

<!--- The following code displays a success or failure message. --->
<cfoutput>#ddxVar.Out1#</cfoutput>
```

The `name` attribute defines a variable that you use to determine the success or failure of the process. Use the `cfoutput` tag to display the success or failure message, as the previous example shows, or use the `cfdump` tag to display a structure:

```
<cfdump var="#ddxVar#">
```

This code returns the following information for each output file in the structure:

- "Successful", if the file is assembled successfully.
- "Reason for failure", if the file is not assembled successfully and the reason for failure is known.
- "Failure", if the file is not assembled successfully and the reason for failure is not known.

Use the `IsDDX` function to determine whether a DDX file or set of instructions is valid.

For detailed examples, see "Assembling PDF Documents" on page 741 in the *ColdFusion Developer's Guide*.

protect action Use the `protect` action to password-protect PDF output files, set permissions, and encrypt PDF output files.

When you use the `protect` action, you must set a `newUserPassword` or a `newOwnerPassword`. (You can set both, as long as the passwords differ.) When you assign a user password to a document, all users must use this password to open the PDF document. The following code adds a user password to a PDF document:

```
<cfpdf action="protect" source="Finances.pdf" destination="myFinances.pdf"
    newUserPassword="keepOut">
```

To set the permissions on the output file, you must set the `newOwnerPassword`. A user who enters the owner password when accessing the PDF file is considered the owner of file. The following example shows how to set a new owner password:

```
<cfpdf action="protect" encrypt="AES_128"source="Book.pdf" destination="MysteryBook.pdf"
    overwrite="yes" newOwnerPassword="pssst" permissions="AllowDegradedPrinting">
```

Because the permissions are set to `AllowDegradedPrinting` in this example, ColdFusion lets users print the document at 150 DPI, but prohibits all other actions. If a user tries to delete the file, for example, ColdFusion generates an error message indicates that the password was entered incorrectly or the permissions do not allow the action to be performed.

ColdFusion does not retain permissions: if you add a `newUserPassword` attribute, you also must set the permission explicitly.

To work with `myVar`, you specify `newownerpw` as the password.

PDF document passwords

A PDF document can have two kinds of passwords: a *user* password and an *owner* password. The following table describes the two types of ColdFusion passwords and their equivalents in Acrobat:

ColdFusion password	Acrobat equivalent	Description
User password	Document Open password, user password	Anyone who tries to open the PDF document must enter the password that you specify. A user password does not allow a user to change restricted features in the PDF document.
Owner password	Permissions password, master password	Lets the person who enters the password restrict access to features in a PDF document.

When you protect a PDF, your password changes to the one you provide. ColdFusion updates the variable's saved password to the one you provide. However, if you provide both passwords, ColdFusion uses the owner password.

The following protects a PDF:

```
<cfpdf action="protect" source="myVar" password="oldpassword"
    permissions="none" newuserpassword="newuserpw"
    newownerpassword="newownerpw">
```

To get all the properties of the PDF, you do the following:

```
<cfpdf action="info" source="myVar" name="info">
```

To get only the properties allowed for the user, you do the following:

```
<cfpdf action="info" source="myVar" password=" newuserpw" name="info">
```

Permissions for PDF documents

The following table lists the permissions an owner can set for PDF documents:

Permissions	Description
`All`	There are no restrictions on the PDF document.
`AllowAssembly`	Users can add the PDF document to a merged document.
`AllowCopy`	Users can copy text, images, and other file content. This setting is required to generate thumbnail images with the `thumbnail` action.
`AllowDegradedPrinting`	Users can print the document at low-resolution (150 DPI).
`AllowFillIn`	Users can enter data into PDF form fields. Users can sign PDF forms electronically.
`AllowModifyAnnotations`	Users can add or change comments in the PDF document.
`AllowModifyContents`	Users can change the file content. Users can add the PDF document to a merged document.
`AllowPrinting`	Users can print the document at high-resolution (print-production quality). This setting is required for use with the `cfprint` tag.
`AllowScreenReaders`	Users can extract content from the PDF document.
`AllowSecure`	Users can sign the PDF document (with an electronic signature).
`None`	Users can view the document only.

Encryption for PDF documents

The `encrypt` attribute sets the type of encryption used for opening a password-protected document. By default, ColdFusion uses the RC4 128-bit encryption algorithm to encrypt PDF files. To change the encryption algorithm, use the `encrypt` attribute with the `protect` action. The following code encrypts the PDF output file with the AES algorithm:

```
<cfpdf action="protect" encrypt="AES_128" source="Book.pdf" destination="MysteryBook.pdf"
    overwrite="yes" newOwnerPassword="pssst" permissions="AllowDegradedPrinting">
```

ColdFusion supports the following encryption algorithms:

Encryption algorithm	Compatibility	Description
AES_128	Adobe Acrobat 7.0 and later	Advanced Encryption Standard (AES) specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits. This is the highest encryption level. This encryption algorithm lets users do the following: • Encrypt all document contents. • Encrypt all document contents except for the metadata. • Encrypt only the file attachments.
RC4_128M	Adobe Acrobat 6.0 and later	RC4 specifies the RSA Security software stream cipher for algorithms such as Secure Sockets Layer (SSL), to protect Internet traffic, and WEP, to secure wireless networks. This encryption algorithm lets users do the following: • Encrypt all document contents. • Encrypt all document contents except for the metadata.
RC4_128	Adobe Acrobat 5.0 and later	RC4 128-bit encryption. This encryption algorithm lets users encrypt the document contents, but not the document metadata.
RC4_40	Adobe Acrobat 3.0 and later	RC4 40-bit encryption. This is the lowest encryption level.
None		The document is not encrypted.

Note: *Document metadata is used in Internet searches. If the metadata is encrypted, search engines cannot search the PDF document. Users running an earlier version of Acrobat cannot open a PDF document with a higher encryption setting. For example, if you specify AES 128 encryption, a user cannot open the document in Acrobat 6.0 or earlier.*

read action Use the `read` action to read the source PDF document into the `name` variable, as the following code shows:

```
<cfif IsPDFFile("Book.pdf")>
    <cfpdf action="read" source="Book.pdf" name="myBook">
...
</cfif>
```

removeWatermark action Use the `removewatermark` action to remove a watermark from a PDF document or specified pages in a document. The following example removes a watermark from the first page of a PDF document and writes the output to a new file:

```
<cfpdf action="removeWatermark" source="Book.pdf" pages="1" destination="newBook.pdf"
overwrite="yes">
```

setInfo action Use the `setinfo` action to specify information associated with a PDF document to be saved with it. Create a structure that contains the relevant information. Use the `info` attribute of the `cfpdf` tag to refer to the structure. The following code shows the elements that you can modify by using the `setInfo` action:

```
<cfset PDFinfo=StructNew()>
<cfset PDFinfo.Title="Make Way for Ducklings">
<cfset PDFinfo.Author="Donald Duck">
<cfset PDFinfo.Keywords="Huey,Dewy,Louie">
<cfset PDFinfo.Subject="Ducks">

<cfpdf action="setInfo" source="chap1.pdf" info="#PDFinfo#" destination="meta1.pdf"
overwrite="yes">
```

thumbnail action Use the `thumbnail` action to generate thumbnail images from the source PDF document.

If you do not specify a destination directory for the thumbnail files, ColdFusion creates a directory for the thumbnails in the directory where the CFM page is located. If you specify a filename as the source, the thumbnail directory name is a concatenation of the name of the source file and _thumbnails. For example, the following code generates a thumbnail image for each page in myBook.pdf and stores them in a directory called myBook_thumbnails:

```
<cfpdf action="thumbnail" source="myBook.pdf">
```

If the CFM page is located in the directory c:\myProject\genThumbnails.cfm, the pathname for the thumbnails directory is c:\myProject\myBook_thumbnails.

By default, ColdFusion generates thumbnail files in JPEG format; the images are scaled to 25% of the original.

You can specify individual pages within the source document to generate thumbnails. Also, you can change the size of the thumbnail; the resolution, the output format (JPEG, PNG, or TIFF); and the prefix used for the thumbnail filenames. The following code generates a low-resolution thumbnail from the first page of the source document that is scaled at 50% of the original size:

```
<cfpdf action="thumbnail" source="myBook.pdf" pages="1" destination="c:\myBook\images"
    imagePrefix="Cover" format="png" scale="50" resolution="low">
```

The full output file pathname is as follows:

```
c:\myBook\images\Cover_page_1.png
```

Note: *To generate thumbnail images, the permissions of the source document must include* `AllowCopy`*. For more information, see* *"Permissions for PDF documents" on page 445**.*

write action se the `write` action to write the source PDF document, or the PDF document stored in memory as a variable, to a file. The following code converts a PDF file stored in memory to a different PDF version and writes the output to a new file:

```
<cfpdf action="read" source="Book.pdf" name="myBook">
<cfpdf action="write" source="myBook" destination="myBook1.pdf"
    version="1.4">
```

PDF versions

Change the PDF version so that users running an older version of Acrobat or Adobe Reader can open the file. The following table shows the compatibility between the PDF version and the corresponding Acrobat and Adobe Reader versions:

PDF version	Compatibility
1.1	Acrobat and Adobe Reader 2
1.2	Acrobat and Adobe Reader 3
1.3	Acrobat and Adobe Reader 4
1.4	Acrobat and Adobe Reader 5
1.5	Acrobat and Adobe Reader 6
1.6	Acrobat and Adobe Reader 7

To linearize PDF documents for faster web display, set the `saveOption` attribute to `linear`, as the following code shows:

```
<cfpdf action="write" source="myBook" destination="myBook1.pdf" saveOption="linear"
    overwrite="yes">
```

Do not use the linear save option if you have to maintain interactivity in PDF forms or if the PDF document is enabled for electronic signatures. To allow for electronic signatures, set the `saveOption` attribute to `incremental`, as the following code shows:

```
<cfpdf action="write" source="myDraft" destination="mySignedDoc.pdf"
    saveOption="incremental" overwrite="yes">
```

Use the `flatten` attribute to flatten forms created in Acrobat:

```
<cfpdf action="write" source="myAcrobatForm.pdf"
    destination="myFlatForm.pdf" flatten="yes" overwrite="yes">
```

Note: *ColdFusion does not support flattening forms created in Adobe® LiveCycle®. For more information about forms created in LiveCycle and Acrobat, see "Manipulating PDF Forms in ColdFusion" on page 725 in the ColdFusion Developer's Guide*

Example

The following example generates thumbnail images from pages in a PDF document and links the thumbnail images to the pages in the PDF document:

```
<h3>PDF Thumbnail Demo</h3>

<!--- Create a variable for the name of the PDF document. --->
<cfset mypdf="myBook">
<cfset thisPath=ExpandPath(".")>
<!--- Use the getInfo action to retrieve the total page count for the
    PDF document. --->
<cfpdf action="getInfo" source="#mypdf#.pdf" name="PDFInfo">
<cfset pageCount="#PDFInfo.TotalPages#">

<!--- Generate a thumbnail image for each page in the PDF source document,
    create a directory (if it doesn't already exist) in the web root that is
    a concatenation of the PDF source name and the word "thumbnails", and
    save the thumbnail images in that directory. --->
<cfpdf action="thumbnail" source="#mypdf#.pdf" overwrite="yes"
    destination="#mypdf#_thumbnails" scale=60>

<!--- Loop through the images in the thumbnail directory and generate a link
    from each image to the corresponding page in the PDF document. --->
<cfloop index="LoopCount" from ="1" to="#pageCount#" step="1">
    <cfoutput>
        <!--- Click the thumbnail image to navigate to the page in the PDF
                document. --->
        <a href="#mypdf#.pdf##page=#LoopCount#" target="_blank">
        <img src="#mypdf#_thumbnails/#mypdf#_page_#LoopCount#.jpg"></a>
    </cfoutput>
</cfloop>
```

cfpdfform

Description

Manipulates existing forms created in Adobe® Acrobat® and Adobe® LiveCycle® Designer. The following list describes some of the tasks you can perform with the `cfpdfform` tag:

- Embed an interactive form created in Acrobat LiveCycle in a PDF document. You use the `cfpdfform` tag to embed the PDF form in a `cfdocument` tag.
- Render an existing form created in Acrobat or LiveCycle. This includes prefilling fields from a database or an XML data file and processing form data from an HTTP post or PDF submission.
- Extract or prefill values in stored PDF forms and save the output to a file or use it to update a data source.

History

ColdFusion 8: Added this tag.

Category

Forms tags

Syntax

```
populate
<cfpdfform
    required
    action = "populate"
    source = "PDF file pathname|byte array"
    optional
    XMLdata = "XML object|XML string|XML data filename|
        URL that returns XML data"
    destination = "output file pathname"
    overwrite = "yes|no"/>

read
<cfpdfform
    required
    action = "read"
    source = "pathname|byte array"
        at least one of the following:
    XMLdata = "variable name for XML data"
    result = "structure containing form field values"
    optional
    overwrite = "yes|no"/>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfdocument`, `cfdocumentsection`, `cfform`, `cfinput`, `cfpdf`, `cfpdfformparam`, `Usagecfpdfparam`, `cfpdf-subform`, `cfprint`, `IsPDFFile`, `IsPDFObject`, "Manipulating PDF Forms in ColdFusion" on page 725 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Action	Req/Opt	Default	Description
action	NA	Required		Action to perform on the source: • populate • read
destination	populate	Optional	write to browser	Pathname for the output file. You can specify an absolute pathname or a pathname relative to the context root. The file extension must be PDF or XDP. The file extension determines the format of the file. (The XDP format applies only to LiveCycle forms.) If you do not specify the destination, ColdFusion displays the form in the browser. Do not specify the destination when you embed a form in a PDF document.
overwrite	populate read	Optional	no	Specifies whether to overwrite the destination file (if action="populate") or the data file (if action="read"): • yes • no
overwriteData	populate	Optional	no	Specifies whether to overwrite existing data in PDF form fields with data from the data source: • yes: Overwrite existing data in the form fields with that from the data source. • no: Retain existing data in form fields and populate only those fields without data. This attribute applies to data supplied from an XML data source and from the cfpdfparam and cfpdfsubform tags.
result	read	Optional (see Description)		ColdFusion structure that contains the form field values. You must specify the XMLdata attribute or the result attribute; you can specify both.
source	populate read	Required		Pathname of the source PDF (absolute path or path relative to the context root) or byte array representing a PDF.
XMLdata	populate read	Optional (see Description)		Pathname for the XML data file. • If action="populate", the data from this file, XML object, or XML string populates the form fields. You can specify a pathname relative to the context root or a relative pathname. • If action="read", ColdFusion writes the data to the variable. You must specify either the XMLdata attribute or the result attribute for the read action; you can specify both.

Usage

ColdFusion supports two types of interactive forms: forms created in Adobe Acrobat 6.0 or earlier, and forms created in Adobe LiveCycle. In Adobe Acrobat Professional and Standard 7.0, Adobe introduced Adobe® LiveCycle® Designer for creating PDF forms. ColdFusion supports forms created in LiveCycle Designer 7.0 and later.

Forms created in Acrobat have a flat structure: a list of fields at the same level. Forms created in LiveCycle Designer are hierarchical, often composed of nested subforms. To map the data to the form field, you use `cfpdfsubform` tags to recreate the structure of the form in ColdFusion. For examples, see the Usage section of the `cfpdfsubform` tag, and ""Manipulating PDF Forms in ColdFusion" on page 725 in the *ColdFusion Developer's Guide*.

populate action Use the `populate` action to populate PDF form fields from the specified data file. You can specify a destination to write the output to a file or write the populated form directly to the browser. To display the interactive PDF form in the browser, do not specify a destination.

The following example shows how to populate a PDF form with an XML data file and display the completed form in a browser:

```
<cfpdfform source="c:\payslipTemplate.pdf" action="populate" XMLdata="c:\formdata.xml"/>
```

This example shows how to populate a PDF form with an XML data file and write the completed form to a new PDF file:

```
<!--- Specify an XML file to populate a PDF form. --->
<cfpdfform source="c:\payslipTemplate.pdf"
    destination="c:\employeeid123.pdf" action="populate"
    XMLdata="c:\formdata.xml"/>
```

Also, you can specify a URL that returns XML data. In the following example, `"http://test1.com/xyz"` returns XML content:

```
<cfpdfform source= "#sourcefile#" action="populate" XMLdata=
    "http://test1.com/xyz" destination="#resultfile#" overwrite="true"/>
```

For forms created in Acrobat, you can write the output to a PDF file only. For forms created in LiveCycle, you have the option to write the output to an XML Data Package (XDP) file. An XDP file is an XML representation of a PDF file.

Note: *Supplied values in form fields created in Acrobat or LiveCycle Designer are case-sensitive. For example, if a check box in a form requires a "Yes" value, the value "yes" does not populate that field.*

The file extension determines the file format: to save the output in XDP format, use an XDP extension in the destination filename:

```
<!--- Specify a an XML file to populate a PDF form. --->
<cfpdfform source="c:\payslipTemplate.pdf"
    destination="c:\employeeid123.xdp" action="populate"
    XMLdata="c:\formdata.xml"/>
```

You can use one or more `cfpdfformparam` tags within a `cfpdfform` tag to populate individual fields in a PDF form.

The following example shows how to populate an existing form created in Acrobat (payslipTemplate.pdf) and create a PDF form (employeeid123.pdf) with the employeeID and salary fields filled in:

```
<!--- This example shows how to populate two fields in a form created in Acrobat. --->
<cfpdfform source="c:\payslipTemplate.pdf"
    destination="c:\employeeid123.pdf" action="populate">
        <cfpdfformparam name="employeeId" value="123">
        <cfpdfformparam name="salary" value="$85,000">
</cfpdfform>
```

ColdFusion requires that you reproduce the exact structure of the source PDF form to populate fields. To verify the structure of a PDF form in ColdFusion, use the `read` action of `cfpdfform` tag, and then use the `cfdump` tag to display the result structure. Use a `cfpdfsubform` tag for each level within the structure. For more information, see "Manipulating PDF Forms in ColdFusion" on page 725 in the *ColdFusion Developer's Guide*.

The following example shows how to populate a form created in LiveCycle. Many forms created from templates in LiveCycle contain a subform called form1. Use the `cfpdfsubform` tag to create a subform in ColdFusion.

```
<!--- This example shows how to populate two fields in a LiveCycle form.
    --->
<cfpdfform source="c:\payslipTemplate.pdf"
    destination="c:\employeeid123.pdf" action="populate">
    <cfpdfsubform name="form1">
        <cfpdfformparam name="employeeId" value="123">
        <cfpdfformparam name="salary" value="$85,000">
    </cfpdfsubform>
</cfpdfform>
```

read action Use the `read` action to read the data from the source PDF form and generate a result structure that contains the form fields and their values. Also, you can use the read action to generate an XML data file from a PDF source file.

The following example shows how to read a PDF file and generate a result structure from the data:

```
<!--- Use the read action to retrieve the values from the saved PDF. --->
<cfpdfform source="c:\employeeid123.pdf" result="resultStruct" action="read"/>
```

You can use the `cfdump` tag to display the result structure:

```
<cfdump var="#resultStruct#">
```

You can use the result fields in ColdFusion, for example, `#resultStruct.employeeId#` and `#resultStruct.salary#`.

The following example shows how to read a PDF file and write the data to an XML file:

```
<cfpdfform source="c:\employeeid123.pdf" result="c:\employeeid123.xml" overwrite="yes"
    action="read"/>
```

The following example shows how to read a PDF file into a variable that contains XML data:

```
<cfpdfform source="c:\employeeid123.pdf" XMLdata="myXMLdata" action="read"/>
```

The following example shows how to read a PDF file into an XML data variable and generate a result structure. The `cffile` tag writes the data to an XML file:

```
<cfset sourcefile = "Grant Application Updated.pdf">
<cfset resultfile = "Expandpath('datafile_result1.xml')">
<!--- Use the cfpdfform tag to read data extracted from a form into an XML data variable and
    generate a result structure. --->
<cfpdfform source= "#sourcefile#" action="read" xmldata="xmldata" result="resultstruct"/>
<!--- Use the cffile tag to write the XML data to a file. --->
<cffile action="write"file="#resultfile#" output="#xmldata#">
<!---- Use the cfdump tag to display the result structure. --->
<cfdump var="#resultstruct#">
```

Extracting data from a PDF submission

Use the following code to extract data from a PDF submission and write it to a structure called fields:

```
<!--- The following code reads the submitted PDF file and generates a result structure called
    fields. --->
<cfpdfform source="#PDF.content#" action="read" result="fields"/>
```

Use the `cfdump` tag to display the data structure, as follows:

```
<cfdump var="#fields#">
```

Note: *When you extract data from a PDF submission, always specify "`#PDF.content#`" as the source.*

You can set the form fields to a variable, as the following code shows:

```
<cfset empForm="#fields.form1#">
```

Use the populate action of the cfpdfform tag to write the output to a file. Specify "#PDF.content#" as the source. In the following example, the unique filename is generated from a field on the PDF form:

```
<cfpdfform action="populate" source="#PDF.content#"
    destination="timesheets\#empForm.txtsheet#.pdf" overwrite="yes"/>
```

Extracting data from an HTTP post submission

An HTTP post submission transmits the data from the PDF form, but not the form itself. You can extract data from the PDF form fields, but you cannot write the output directly to a file. To extract the data and update a database, for example, you must map the fields in the database to the structure and HTTP post data exactly.

Note: *The structure of the HTTP post data (after submission) is not the same as the structure of the PDF form (before data submission). For examples of both, see "Manipulating PDF Forms in ColdFusion" on page 725 in the ColdFusion Developer's Guide.*

To determine the structure of the HTTP post data, use the cfdump tag with the form name as the variable to display the data structure, as follows:

```
<cfdump var="#FORM.form1#">
```

Note: *When you extract data from an HTTP post submission, always specify the form name as the source. For example, specify "#FORM.form1#" for a form generated from a template in LiveCycle Designer. When data extraction that uses the cfpdfform tag results in more than one page, instead of returning one structure, ColdFusion returns one structure per page.*

Embedding PDF forms within a PDF document

You can use the cfpdfform tag inside the cfdocument tag to embed an existing interactive PDF form within a PDF document. Use at least one cfdocumentsection tag with the cfpdfform tag, but do not place the cfpdfform tag within the cfdocumentsection tag. For more information about embedding PDF forms, see "Manipulating PDF Forms in ColdFusion" on page 725 in the *ColdFusion Developer's Guide*.

Flattening forms created in Acrobat

You use the cfpdf tag to flatten forms created in Acrobat. ColdFusion does not support flattening forms created in LiveCycle. For more information, see "Assembling PDF Documents" on page 741 in the *ColdFusion Developer's Guide*.

Printing forms

Use the cfprint tag to print forms created in Acrobat. Markups, such as sticky notes, comments, and editorial revisions, are not printed with the form. You cannot use the cfprint tag to print forms created in LiveCycle Designer.

Example

The following example shows how to embed an interactive PDF form in a PDF document created with the cfdocument tag:

```
<!--- The following code extracts data from the cfdocexamples database based
    on a username entered in a login form. --->
<cfquery name="getEmpInfo" datasource="cfdocexamples">
    SELECT * FROM EMPLOYEES
    WHERE EMAIL = <cfqueryparam value="#form.username#">
</cfquery>
```

```
<!--- The following code creates a PDF document with headers
    and footers. --->
<cfdocument format="pdf">
    <cfdocumentitem type="header">
    <font size="-1" align="center"><i>Nondisclosure Agreement</i></font>
    </cfdocumentitem>
    <cfdocumentitem type="footer">
    <font size="-1"><i>Page <cfoutput>#cfdocument.currentpagenumber# of
        #cfdocument.totalpagecount#</cfoutput></i></font>
    </cfdocumentitem>

<!--- The following code creates the first section in the PDF document. --->
    <cfdocumentsection>
    <h3>Employee Nondisclosure Agreement</h3>
    <p>Please verify the information in the enclosed form. Make any of the
    necessary changes in the online form and click the <b>Print</b> button.
    Sign and date the last page. Staple the pages together and return the
    completed form to your manager.</p>
    </cfdocumentsection>

<!--- The following code embeds an interactive PDF form within the PDF
    document with fields populated by the database query. The cfpdpfform tag
    automatically creates a section in the PDF document. Do not embed the
    cfpdfform within cfdocumentsection tags. --->

    <cfpdfform action="populate" source="c:\forms\embed.pdf">
        <cfpdfsubform name="form1">
            <cfpdfformparam name="txtEmpName" value="#getEmpInfo.FIRSTNAME#
                #getEmpInfo.LASTNAME#">
            <cfpdfformparam name="txtDeptName" value="#getEmpInfo.DEPARTMENT#">
            <cfpdfformparam name="txtEmail" value="#getEmpInfo.IM_ID#">
            <cfpdfformparam name="txtPhoneNum" value="#getEmpInfo.PHONE#">
            <cfpdfformparam name="txtManagerName" value="Randy Nielsen">
        </cfpdfsubform>
    </cfpdfform>

<!--- The following code creates the last document section. Page numbering
    resumes in this section. --->
    <cfdocumentsection>
    <p>I, <cfoutput>#getEmpInfo.FIRSTNAME# #getEmpInfo.LASTNAME#</cfoutput>,
        hereby attest that the information in this document is accurate and complete.</p>
    <br/><br/>
    <table border="0" cellpadding="20">
    <tr><td width="300">
    <hr />
    <p><i>Signature</i></p></td>
    <td width="150"><hr />
    <p><i>Today's Date</i></p></td></tr>
    </cfdocumentsection>
</cfdocument>
```

cfpdfformparam

Description

Provides additional information to the cfpdfform tag.

The cfpdfformparam tag is always a child tag of the cfpdfform or cfpdfsubform tag. Use the cfpdfformparam tag to populate fields in a PDF form.

History

ColdFusion 8: Added this tag.

Category

Forms tags

Syntax

```
<cfpdfform ...>
    <cfpdfformparam
    name = "field name"
    value = "ColdFusion variable"
    index = "integer">
</cfpdfform>
```

Note: *You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.*

See also

cfdocument, cfdocumentsection, cfform, cfinput, cfpdf, cfpdfform Usage, Usagecfpdfparam, cfpdf-subform, cfprint, IsPDFFile, IsPDFObject

Attributes

Attribute	Req/Opt	Default	Description
index	Optional	1	Index associated with the field name. If multiple fields have the same name, use the index value to locate one of them. Applies to forms created in LiveCycle only.
name	Required		Field name on the PDF form.
value	Required		Value associated with the field name. For interactive fields, specify a ColdFusion variable.

Usage

Use the cfpdfformparam tag inside the cfpdfform tag or the cfpdfsubform tag to populate fields in a PDF form.

Use the index attribute of the cfpdfformparam tag to specify fields with the same name and different values, as the following code shows:

```
<!--- This example shows how to use multiple cfpdfformparam tags with the same name and
    different index values for a PDF form that contains fields with same name. --->
<cfpdfform source="c:\payslipTemplate.pdf"
    destination="c:\employeeid123.pdf" action="populate">
        <cfpdfformparam name="phone" value="781-869-1234" index="1"/>
        <cfpdfformparam name="phone" value="617-273-9021" index="2"/>
</cfpdfform>
```

Note: *Use the index attribute with forms created in LiveCycle only. Forms created in Acrobat cannot contain more than one field with the same name; therefore the index attribute is not valid.*

Example

See the cfpdfform tag examples.

cfpdfparam

Description

Provides additional information for the cfpdf tag. The cfpdfparam tag applies only to the merge action of the cfpdf tag and is always a child tag of the cfpdf tag.

History

ColdFusion 8: Added this tag.

Category

Forms tags

Syntax

```
<cfpdf action = "merge" ..>
    <cfpdfparam
    pages = "page number|page range|comma-separated page numbers"
    password = "user or owner password"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable">
</cfpdf>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfdocument, cfdocumentsection, cfpdf, Usagecfpdfform, Usagecfpdfformparam, cfpdfparam, cfpdf-subform, cfprint, IsPDFFile, IsPDFObject

Attributes

Attribute	Req/Opt	Default	Description
pages	Optional		Page or pages of the PDF source file to merge. You can specify a range of pages, for example, "1—5 ", or a comma-separated list of pages, for example, "1-5,9—10,18".
password	Optional		User or owner password, if the source PDF file is password-protected.
source	Required		Source PDF file to merge. You can specify a PDF variable, a cfdocument variable, or the path-name to a file.

Usage

Use the cfpdfparam tag to merge several PDF documents into one file. The cfpdfparam tag lets you specify the order of source files explicitly. You can use this tag to merge pages from multiple PDF document source files in different locations.

The following code creates a single PDF document called combined.pdf that contains pages 1–3 and page 5 of the file abc.pdf, followed by all of the pages in xyz.pdf, a file in memory with the variable name myPDFvariable, and lastly pages 10–90 from the file abc.pdf. The password attribute applies only if the source file is password-protected:

```
<cfpdf action="merge" destination="combined.pdf" overwrite="yes">
    <cfpdfparam source="c:\abc.pdf" pages="1-3,5" password="adobe">
    <cfpdfparam source="c:\new\xyz.pdf">
    <cfpdfparam source="myPDFvariable">
    <cfpdfparam source="abc.pdf" pages="10-90" password="adobe">
</cfpdf>
```

Note: *When you use the* `cfpdfparam` *tag with the* `cfpdf merge` *action, you must specify either the* `destination` *attribute or the* `name` *attribute for the* `cfpdf` *tag.*

Example

The following ColdFusion page creates a form for downloading tax forms and tax information booklets:

```
<h3>Downloading Federal Tax Documents</h3>
<p>Please choose the your type of business.</p>
<!--- Create the ColdFusion form to determine which PDF documents to merge. --->
<table>
<cfform action="cfpdfMergeAction.cfm" method="post">
    <tr><td><cfinput type="radio" name="businessType" Value="SoleP">
        Sole Proprieter</td></tr>
    <tr><td><cfinput type="radio" name="businessType"
        Value="Partner">Partnership</td></tr>
    <tr><td><cfinput type="radio" name="businessType" Value="SCorp">S Corporation</td></tr>
    <cfinput type = "hidden" name = "selection required" value = "must make a selection">
    <tr><td><cfinput type="Submit" name="OK" label="OK"></td></tr>
    </tr>
</cfform>
</table>
```

The ColdFusion action page merges PDF files in different locations based on the selection in the form:

```
<!--- Create a merged PDF document based on the selection in the form. --->
<cfpdf action="merge" name="taxDoc">
    <cfif #form.businessType# is "SoleP">
        <cfpdfparam source="taxForms\f2106ez.pdf">
        <cfpdfparam source="taxForms\f1040.pdf">
        <cfpdfparam source="taxForms\f1040sc.pdf">
        <cfpdfparam source="taxInfo\i1040sc.pdf">
        <cfpdfparam source="taxInfo\i2106.pdf">
        <cfpdfparam source="taxInfo\i1040sc.pdf">
        <cfpdfparam source="taxInfo\p535.pdf">
        <cfpdfparam source="taxInfo\p560.pdf">
        <cfpdfparam source="taxInfo\p334.pdf">
    <cfelseif #form.businessType# is "Partner">
        <cfpdfparam source="taxForms\f1065.pdf">
        <cfpdfparam source="taxForms\f1065b.pdf">
        <cfpdfparam source="taxForms\f1065bsk.pdf">
        <cfpdfparam source="taxForms\f8804.pdf">
        <cfpdfparam source="taxForms\f8825.pdf">
        <cfpdfparam source="taxInfo\p535.pdf">
        <cfpdfparam source="taxInfo\p560.pdf">
        <cfpdfparam source="taxInfo\i1065bsk.pdf">
    <cfelseif #form.businessType# is "SCorp">
        <cfpdfparam source="taxForms\f1120s.pdf">
        <cfpdfparam source="taxForms\f2553.pdf">
        <cfpdfparam source="taxForms\f8453s.pdf">
        <cfpdfparam source="taxForms\f8825.pdf">
        <cfpdfparam source="taxInfo\i1120s.pdf">
        <cfpdfparam source="taxInfo\p542.pdf">
        <cfpdfparam source="taxInfo\p535.pdf">
        <cfpdfparam source="taxInfo\p560.pdf">
    </cfif>
</cfpdf>

<cfpdf action="write" source="taxDoc" destination="c:\taxDoc.PDF"
    overwrite="yes"/>
```

Note: *ColdFusion automatically flattens form fields when you use the* `merge` *action of the* `cfpdf` *tag.*

cfpdfsubform

Description

Populates a subform within the cfpdfform tag.

The cfpdfsubform tag can be a child tag of the cfpdfform tag or nested in another cfpdfsubform tag.

History

ColdFusion 8: Added this tag.

Category

Forms tags

Syntax

```
<cfpdfform ..>
    <cfpdfsubform
    name = "field name"
    index = "integer">
    </cfpdfsubform>
</cfpdfform>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfdocument, cfdocumentsection, cfform, cfinput, cfpdf, cfpdfformUsage, cfpdfformparam, cfpdfparam, cfprint, IsPDFFile, IsPDFObject

Attributes

Attribute	Req/Opt	Default	Description
index	Optional	1	Index associated with the field name. If multiple fields have the same name, ColdFusion uses the index value to locate one of them.
name	Required		Name of the subform corresponding to subform name in the PDF form.

Usage

Use the cfpdfsubform tag with the cfpdfform tag to populate one or more subforms within a PDF form. The cfpdfsubform tag can contain multiple cfpdfformparam tags. Also, you can nest subforms, as the following example shows:

```
<!--- This example shows how to nest cfpdfsubform tags. --->
<cfpdfform source="c:\payslipTemplate.pdf"
            destination="c:\employeeid123.pdf" action="populate">
    <cfpdfsubform name="employeeDetail">
        <cfpdfsubform name="address">
            <cfpdfformparam name="txtAddLine1" value="572 Evergreen Terrace">
            <cfpdfformparam name="txtCity" value="Springfield">
            <cfpdfformparam name="txtState" value="Oregon">
            <cfpdfformparam name="txtZip" value="65412">
            <cfpdfformparam name="txtCountry" value="United States">
        </cfpdfsubform>
        <cfpdfformparam name="txtEmployeeId" value="879104">
        <cfpdfformparam name="numSalary" value="$85,000">
    </cfpdfsubform>
</cfpdfform>
```

Use subforms to match the exact structure of the source PDF form. If you do not, ColdFusion cannot prefill the form with data and generates an error. Many of the forms generated from templates in LiveCycle contain a subform called form1. You must specify this as a subform in your code, as the following example shows:

```
<cfpdfform source="c:\forms\timesheetForm.pdf" action="populate">
    <cfpdfsubform name="form1">
        <cfpdfformparam name="txtCompanyName" value="Adobe">
        <cfpdfformparam name="txtManager" value="Randy Nielsen">
    </cfpdfsubform>
</cfpdfform>
```

To verify the structure of a PDF form in ColdFusion, use the `read` action of the `cfpdfform` tag, as the following example shows:

```
<cfpdfform source="c:\forms\timesheetForm.pdf" result="resultStruct" action="read"/>
```

Then use the `cfdump` tag to display the structure:

```
<cfdump var="#resultStruct#">
```

Example

See the `cfpdfform` tag examples.

cfpod

Description

Creates a pod, an area of the browser window or layout area with an optional title bar and a body that contains display elements.

Category

Display management tags

Syntax

```
<cfpod
    source = "path"
    bodyStyle = "CSS style specification"
    headerStyle = "CSS style specification"
    height = "number of pixels"
    name = "string"
    onBindError = "JavaScript function name"
    title = "string"
    width = "number of pixels"/>

OR

<cfpod
    bodyStyle = "CSS style specification"
    headerStyle = "CSS style specification"
    height = "number of pixels"
    name = "string"
    onBindError = "JavaScript function name"
    title = "string"
    width = "number of pixels">
        pod contents
</pod>
```

If the tag does not have a body and end tag, you must close it with /> character combination.

Note: *You can specify this tag's attribute in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute name as structure key.*

See also

`cfajaximport`, `cfdiv`, `cflayout`, `cfwindow`

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
bodyStyle	Optional		A CSS style specification for the pod body. As a general rule, use this attribute to set color and font styles. Using this attribute to set the height and width, for example, can result in distorted output.
headerStyle	Optional		A CSS style specification for the pod header. As a general rule, use this attribute to set color and font styles. Using this attribute to set the height and width, for example, can result in distorted output.
height	Optional	100	Height if the control, including the title bar and borders, in pixels

Attribute	Req/Opt	Default	Description
name	Optional		Name of the pod control.
onBindError	Optional	See Description	The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and have specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up displays.
overflow	Optional	auto	Specifies how to display child content whose size would cause the control to overflow the pod boundaries. The following values are valid: • auto: shows scrollbars when necessary. • hidden: does not allow access to overflowing content. • scroll: always shows horizontal and vertical scroll bars, even if they are not needed. • visible: content can display outside the bounds of the pod. **Note:** In Internet Explorer, pods with the visible setting expand to fit the size of the contents, rather than having the contents extend beyond the layout area.
source	Required if the tag does not have a body		A URL that returns the pod contents. ColdFusion uses standard page path resolution rules. If you specify this attribute and the cfpod tag has a body, ColdFusion ignores the body contents. You can use a bind expression with dependencies in this attribute; for more information see Usage. **Note:** If a CFML page specified in this attribute contains tags that use AJAX features, such as cfform, cfgrid, and cfwindow, you must use a cfajaximport tag on the page with the cfpod tag. For more information, see cfajaximport.
title	Optional		Text to display in the pod's title bar. You can use HTML mark-up to control the title appearance, of example to show the text in red italic font. If you omit this attribute, the pod does not have a title bar.
width	Optional	500	Width if the control, including borders, in pixels.

Usage

You use a source attribute or a tag body to specify the pod contents; if you specify both, ColdFusion uses the contents specified by the source attribute and ignores the tag body. If you use a source attribute, an animated icon and the text "Loading..." appears while the contents is being fetched.

If the source attribute specifies a page that defines JavaScript functions, the function definitions on that page must have the following format:

```
functionName = function(arguments) {function body}
```

Function definitions that use the following format may not work:

```
function functionName (arguments) {function body}
```

However, Adobe recommends that you include all custom JavaScript in external JavaScript files and import them on the application's main page, and not write them inline in code that you get using the source attribute. Imported pages do not have this function definition format restriction.

If you use the source attribute, you can use a *bind expression* to include form field values or other form control attributes as part of the source specification. You can bind to HTML format form controls only.

To use a bind expression, specify a URL and pass one or more URL parameters the page, including *bind parameters*. In its most basic form, a bind parameter consists of the `name` or `id` attribute of the control to which you are binding in braces ({ }). To include the value of the `city` control as a bind parameter, for example, use the following format:

```
source="/myapplication/cityPod.cfm?cityname={city}"
```

For detailed information about using bind expressions, see "Binding data to form fields" on page 650 in the *ColdFusion Developer's Guide*.

Example

The following CFML page displays two pods in a vertical layout. Each pod gets its contents from a displayforpod.cfm page that uses the `cffeed` tag to get an Atom feed.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Untitled Document</title>
</head>

<body>
<cflayout type="hbox" style="background-color:##CCffFF; color:red;">
    <cflayoutarea>
        <cfpod name="pod01" source="displayforpod.cfm?start=1" height="500" width="300"
            title="Comment 1"/>
    </cflayoutarea>
    <cflayoutarea>
        <cfpod name="pod02" source="displayforpod.cfm?start=2" height="500" width="450"
            title="Comment 2"/>
    </cflayoutarea>
</cflayout>
</body>
</html>
```

The following code shows the contents of the displayforpod.cfm page:

```
<cffeed action="read" source="http://googleblog.blogspot.com/atom.xml"
            query="feedQuery" properties="feedMetadata" >

<cfloop query = "feedQuery"
        startRow = "#url.start#" endRow = "#url.start#">
    <cfoutput>#feedQuery.content#<br />
    ========================================<br/>
    </cfoutput>
</cfloop>
```

cfpop

Description

Retrieves or deletes e-mail messages from a POP mail server.

Category

Communications tags, Internet protocol tags

Syntax

```
<cfpop
    server = "server name"
    action = "getHeaderOnly|getAll|delete"
    attachmentPath = "path"
    debug = "yes|no">
    generateUniqueFilenames = "yes|no"
    maxRows = "number"
    messageNumber = "number"
    name = "query name"
    password = "password"
    port = "port number"
    startRow = "number"
    timeout = "seconds"
    uid = "number"
    username = "user name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfftp`, `cfhttp`, `cfldap`, `cfmail`, `cfmailparam`, `SetLocale`; "Sending and Receiving E-Mail" on page 998 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7.01: Added cids query variable.

ColdFusion MX 6.1:

- Added support for multipart mail messages with Text and HTML parts.
- Changed the attachment name separator: the TAB character is now the separator between attachment names in the `attachments` and `attachmentfiles` query fields if a message has multiple attachments. This behavior is identical to ColdFusion 5 and earlier versions.

ColdFusion MX: Changed the attachment name separator: the comma separates names in the `attachments` and `attachmentfiles` query fields if a message has multiple attachments.

Attributes

Attribute	Req/Opt	Default	Description
server	Required		POP server identifier: • A host name, for example, "biff.upperlip.com". • An IP address, for example, "192.1.2.225".
action	Optional	getHeaderOnly	• getHeaderOnly: returns message header information only • getAll: returns message header information, message text, and attachments if attachmentPath is specified • delete: deletes messages on POP server
attachmentPath	Optional		If action="getAll", specifies a directory in which to save any attachments. If the directory does not exist, ColdFusion creates it. If you omit this attribute, ColdFusion does not save any attachments. If you specify a relative path, the path root is the ColdFusion temporary directory, which is returned by the GetTempDirectory function.
debug	Optional	no	• yes: sends debugging output to standard output. By default, if the console window is unavailable on server configurations, ColdFusion sends output to *cf_root*/runtime/logs/coldfusion-out.log. On J2EE configurations, with JRun, the default location is *jrun_home*/logs/*servername*-out.log. *Caution*: If you set this option to Yes, ColdFusion writes detailed debugging information to the log, including all retrieved message contents, and can generate large logs quickly. • no: does not generate debugging output.
generateUniqueFilenames	Optional	no	• yes: generate unique filenames for files attached to an e-mail message, to avoid naming conflicts when files are saved. • no
maxRows	Optional	Retrieves all available rows	Number of messages to return or delete, starting with the number in startRow. Ignored if messageNumber or uid is specified.
messageNumber			Message number or comma-separated list of message numbers to get or delete. Invalid message numbers are ignored. Ignored if uid is specified.
name	Required if action = "getAll" or "getHeaderOnly"		Name for query object that contains the retrieved message information.
password	Optional		Password that corresponds to username.
port	Optional	110	POP port.
startRow	Optional	1	First row number to get or delete. Ignored if messageNumber or uid is specified.
timeout	Optional	60	Maximum time, in seconds, to wait for mail processing.

Attribute	Req/Opt	Default	Description
uid			UID or a comma-separated list of UIDs to get or delete. Invalid UIDs are ignored.
username	Optional		A user name.

Usage

The cfpop tag retrieves one or more mail messages from a POP server and populates a ColdFusion query object with the resulting messages, one message per row. Alternatively, it deletes one or more messages from the POP server.

Note: *When the cfpop tag encounters malformed mail messages, it does not generate errors; instead, it returns empty fields.*

To optimize performance, two retrieve options are available. Message header information is typically short, and therefore quick to transfer. Message text and attachments can be very long, and therefore take longer to process.

The cfpop query variables

The following table describes the variables that provide information about the query that is returned by cfpop:

Variable names	Description
queryname.recordCount	Number of records returned by query.
queryname.currentRow	Current row that cfoutput is processing.
queryname.columnList	List of column names in query.
queryname.UID	Unique identifier for the e-mail message file.
queryname.cids	Structure that contains key-value pairs. The keys are the names of image files that are embedded in the e-mail message; the values are their cids. You can use the cid to find the correct place of an image in an HTML e-mail message that the cfpop tag retrieves. If the e-mail message contains more than one embedded image, only the last embedded image is available.

Query message header and body columns

The following table lists the message header and body columns that are returned if action = "getHeaderOnly" or "getAll":

Column name	getHeaderOnly returns	getAll returns
queryname.date	yes	yes
queryname.from	yes	yes
queryname.messagenumber	yes	yes
queryname.messageid	yes	yes
queryname.replyto	yes	yes
queryname.subject	yes	yes
queryname.cc	yes	yes
queryname.to	yes	yes
queryname.body	no	yes
queryname.textBody	no	yes
queryname.HTMLBody	no	yes

Column name	getHeaderOnly returns	getAll returns
queryname.header	yes	yes
queryname.attachments	no	yes
queryname.attachmentfiles	no	yes

If the mail message includes a part with a Content-Type of text/plain, the queryname.textBody column contains the part's message content. If the mail message includes a part with a Content-Type of text/HTML, the queryname.HTMLBody column contains the part's message content. If no Content-Type matches these types, the columns are empty. The queryname.Body column always contains the first message body found.

The queryname.attachments column contains a tab-separated list of all the attachment names. The queryname.attachmentfiles column contains a tab-separated list of the locations of the attachment files. Use the `cffile` tag to delete these temporary files when you have processed them.

To create a ColdFusion date/time object from the date-time string that is extracted from a mail message in the queryname.date column, use the following table:

Locale	How to create a ColdFusion date/time object from queryname.date
English (US)	Use the `ParseDateTime` function. If you specify the `pop-conversion` attribute, the function adjusts the date/time object to UTC.
Other	Extract the date part of string; pass it to the `LSParseDateTime` function.

***Note:** To set the default display format of date, time, number, and currency values, use the `SetLocale` function.*

For more information on `cfpop`, see "Sending and Receiving E-Mail" on page 998 in the *ColdFusion Developer's Guide*.

Example

```
<!--- This view-only example shows the use of cfpop. --->
<h3>cfpop Example</h3>
<p>cfpop lets you retrieve and manipulate mail in a POP3 mailbox.
    This view-only example shows how to create one feature of
    a mail client, to display the mail headers in a POP3 mailbox.
<p>To execute this, un-comment this code and run with a mail-enabled CF Server.
<!---
<cfif IsDefined("form.server")>
    <!--- Make sure server, username are not empty. --->
    <cfif form.server is not "" and form.username is not "">
        <cfpop server = "#form.popserver# " username = #form.username# password = #form.pwd#
        action = "getHeaderOnly" name = "GetHeaders ">
        <h3>Message Headers in Your Inbox</h3>
        <p>Number of Records:
        <cfoutput>#GetHeaders.recordCount#</cfoutput></p>

        <ul>
            <cfoutput query = "GetHeaders">
            <li>Row: #currentRow#: From: #From# -- Subject: #Subject#
            </cfoutput>
        </ul>
    </cfif>
</cfif>

<form action = "cfpop.cfm " method = "post">
    <p>Enter your mail server:</p>
```

```
    <p><input type = "Text" name = "popserver"></p>
    <p>Enter your username:</p>
    <p><input type = "Text" name = "username"></p>
    <p>Enter your password:</p>
    <p><input type = "password" name = "pwd"></p>
    <p><input type = "Submit" name = "get message headers"></p>
</form>
--->
```

cfpresentation

Description

Defines the look of a dynamic slide presentation and determines whether to write the presentation files to disk. The `cfpresentation` tag is the parent tag for one or more `cfpresentationslide` tags, where you define the content for the presentation, and the `cfpresenter` tags, which provide information about the people presenting the slides.

History

ColdFusion 8: Added this tag.

Category

Data output tags

Syntax

```
<cfpresentation
    title = "text string"
    authPassword = "authentication password"
    authUser = "authentication user name"
    autoPlay = "yes|no"
    backgroundColor = "hexadecimal color|HTML named color"
    control = "normal|brief"
    controlLocation = "right|left"
    directory = "pathname"
    glowColor = "hexadecimal color|HTML named color"
    initialTab = "outline|search|notes"
    lightColor = "hexadecimal color|HTML named color"
    loop = "yes|no"
    overwrite = "yes|no"
    primaryColor = "hexadecimal color|HTML named color"
    proxyHost = "IP address or server name for proxy host"
    proxyPassword = "password for the proxy host"
    proxyPort = "port of the proxy host"
    proxyUser = "user name for the proxy host"
    shadowColor = "hexadecimal color|HTML named color"
    showNotes = "yes|no"
    showOutline = "yes|no"
    showSearch = "yes|no"
    textColor = "hexadecimal color|HTML named color"
    userAgent = "HTTP user agent identifier">
    presentation content...
</cfpresentation>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfchart`, `cfpresentationslide`, `cfpresenter`, `cfreport`, "Creating Slide Presentations" on page 856 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
authPassword	Optional		Sends a password to the target URL for Basic Authentication. Combined with username to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
authUser	Optional		Sends a user name to the target URL for Basic Authentication. Combined with password to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
autoPlay	Optional	yes	Specifies whether to play the presentation automatically: • yes: the presentation automatically runs through the entire presentation at startup. • no: the user must click the Play button to start the presentation and click the Next button to advance to the next slide in the presentation.
backgroundColor	Optional	727971	Background color of the presentation. The value is hexadecimal: use the form "##xxxxxx" or "##xxxxxxxx", where x = 0—9 or A—F; use two number signs or none. Also, you can use a subset of HTML named colors. For more information, see "Named colors" on page 472.
control	Optional	normal	Presentation control: • normal • brief
controlLocation	Optional	right	Specifies the location of the presentation control: • right • left
directory	Optional		Directory where the presentation is saved. This can be an absolute path or a path relative to the CFM page. Also, ColdFusion creates a subdirectory called data that contains: • A SWF file for each slide • srchdata.xml (which creates the search interface) • vconfig.xml • viewer.xml • images, video clips, and SWF files referenced by the cfpresentationslide tags If you do not specify a directory, ColdFusion writes the files to a temp directory and runs the presentation in the client browser.
glowColor	Optional	35D334	Color used for glow effects on the buttons. The value is hexadecimal: use the form "##xxxxxx" or "##xxxxxxxx", where x = 0—9 or A—F; use two number signs or none. Also, you can use a subset of HTML named colors. For more information, see "Named colors" on page 472.
initialTab	Optional	outline	Specifies which tab displays on top when the presentation is run. This applies only when the control value is normal: • outline • search • notes

Attribute	Req/Opt	Default	Description
lightColor	Optional	4E5D60	Light color used for light-and-shadow effects. The value is hexadecimal: use the form "##xxxxxx" or "##xxxxxxxxx", where x = 0—9 or A—F; use two number signs or none. Also, you can use a subset of HTML named colors. For more information, see "Named colors" on page 472.
loop	Optional	no	Specifies whether the presentation runs in a loop: • yes: the presentation restarts automatically after it ends. • no: the user must click the Play button to restart the presentation.
overwrite	Optional	yes	Specifies whether files in the directory are overwritten. Valid only when the directory attribute is specified. • yes: overwrites files if they are already present • no: creates new files
primaryColor	Optional	6F8488	Primary color of the presentation. The value is hexadecimal: use the form "##xxxxxx" or "##xxxxxxxxx", where x = 0—9 or A—F; use two number signs or none. Also, you can use a subset of HTML named colors. For more information, see "Named colors" on page 472.
proxyHost	Optional		Host name or IP address of a proxy server to which to send the request.
proxyPassword	Optional		Password required by the proxy server.
proxyPort	Optional	80	The port to connect to on the proxy server.
proxyUser	Optional		User name to provide to the proxy server.
shadowColor	Optional	000000	Shadow color used for light-and-shadow effects. The value is hexadecimal: use the form "##xxxxxx" or "##xxxxxxxxx", where x = 0—9 or A—F; use two number signs or none. Also, you can use a subset of HTML named colors. For more information, see "Named colors" on page 472.
showNotes	Optional	no	Specifies whether the Notes tab is present in the presentation control panel: • yes • no
showOutline	Optional	yes	Specifies whether the Outline is present in the presentation control panel: • yes • no
showSearch	Optional	yes	Specifies whether the Search tab is present in the presentation control panel: • yes • no
textColor	Optional	FFFFFF	Color for all the text in the presentation user interface. The value is hexadecimal: use the form "##xxxxxx" or "##xxxxxxxxx", where x = 0—9 or A—F; use two number signs or none. Also, you can use a subset of HTML named colors. For more information, see "Named colors" on page 472.
title	Required		Title of the presentation
userAgent	Optional	ColdFusion	Text to put in the HTTP User-Agent request header field. Used to identify the request client software.

Usage

Use the cfpresentation tag to create the container for a slide presentation. You can define the position and appearance of the presentation controls, the background color, and the text for the presentation. Also, use this tag to determine whether to write the presentation to files or to run it directly in the client browser.

The settings in the cfpresentation tag do not affect the appearance of the content defined in the cfpresentationslide tags.

Named colors

The cfpresentation tag supports the following named colors for use with the backgroundColor, glowColor, lightColor, primaryColor, shadowColor, and textColor attributes:

Named color	Hexadecimal value
red	FF0000
green	008000
blue	0000FF
black	000000
white	FFFFFF
yellow	FFFF00
gray	808080
darkgray	A9A9A9
lightgray	D3D3D3
cyan	00FFFF
magenta	FF00FF
orange	FFA500
pink	FFC0CB

Example

```
<!--- This example shows how to create a slide presentation from --->
<!--- an HTML file and from HTML code on the CFM page and write --->
<!--- the presentation files to a directory called myPresentation, --->
<!--- which is relative to the CFM page. --->
<cfpresentation title="Sales Presentation" directory="myPresenation">
    <cfpresenter name="Shyam" title="Vice President" email="shyam@somecompany.com"
image="shyam.jpg">
    <cfpresenter name="Ram" title="Sr. Vice President" email="ram@somecompany.com">

<!--- The following code creates a slide from an HTML file --->
<!--- located on the ColdFusion server. --->
    <cfpresentationslide src="introduction.htm" title="Introduction" presenter="Shyam"
        audio="myAudio.mp3" duration="36"/>

<!--- The following code creates a slide from HTML code in the CFM file. --->
    <cfpresentationslide>
        <h3>Sales</h3>
        <ul>
            <li>Overview</li>
            <li>Q1 Sales Figures</li>
            <li>Projected Sales</li>
            <li>Competition</li>
            <li>Advantages</li>
            <li>Long Term Growth</li>
        </ul>
    </cfpresentationslide>

<!--- The following code creates a slide from HTML and CFML code. --->
    <cfpresentationslide Title="Q1 Sales Figures" duration="14" presenter="Ram"
```

```
            audio="myAudio2.mp3">
        <h3>Q1 Sales Figures</h3>
        <cfchart format="png" showborder="yes" chartheight="250" chartwidth="300"
            pieslicestyle="sliced">
        <cfchartseries type="pie">
            <cfchartdata item="Europe" value="9">
            <cfchartdata item="Asia" value="20">
            <cfchartdata item="North America" value="50">
            <cfchartdata item="South America" value="21">
        </cfchartseries>
        </cfchart>
    </cfpresentationslide>
</cfpresentation>
```

cfpresentationslide

Description

Creates a slide dynamically from a source file or HTML and CFML code on the ColdFusion page. The `cfpresentationslide` is a child tag of the `cfpresentation` tag.

History

ColdFusion 8: Added this tag.

Category

Data output tags

Syntax

```
<cfpresentation ...>
    <cfpresentationslide
    advance = "auto|never|click"
    audio = "pathname relative to the CFM page or the web root for audio file"
    authPassword = "authentication password"
    authUser = "authentication user name"
    duration = "duration of slide in seconds"
    marginBottom = "margin in pixels"
    marginLeft = "margin in pixels"
    marginRight = "margin in pixels"
    marginTop = "margin in pixels"
    notes = "text string"
    presenter = "presenter name"
    scale = "decimal"
    src = "absolute path|URL|path relative to CFM page"
    title = "text string"
    userAgent = "HTTP user agent identifier"
    video = "pathname relative to the CFM page or the web root
        for video file"/>
</cfpresentation>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfchart`, `cfpresentation`, `cfpresenter`, `cfreport`, "Creating Slide Presentations" on page 856 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
advance	Optional	See Description	Overrides the cfpresentation tag autoPlay attribute for the slide: • auto: after the slide plays, the presentation advances to the next slide automatically. This is the default value if cfpresentation autoPlay="yes". • never: after the slide plays, the presentation does not advance to the next slide until the user clicks the Next button. This is the default value if cfpresentation autoPlay="no". • click: after the slide plays, the presentation advances to the next slide if the user clicks anywhere in the main presentation area.
audio	Optional		Pathname of the audio file relative to the CFM page or the web root. The audio file must be an MP3 file. You cannot specify both audio and video for a slide.
authPassword	Optional		Use to pass a password to the target URL for Basic Authentication. Combined with username to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
authUser	Optional		Use to pass a user name to the target URL for Basic Authentication. Combined with password to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.
duration	Optional		Duration in seconds that the slide is played. If you do not specify a duration, the slide plays for the duration of the audio clip associated with the slide.
marginBottom	Optional	0	Bottom margin of the slide.
marginLeft	Optional	0	Left margin of the slide.
marginRight	Optional	0	Right margin of the slide
marginTop	Optional	0	Top margin of the slide
notes	Optional		Notes used for the slide. Notes are displayed only if the showNotes attribute of the cfpresentationslide tag is set to yes.
presenter	Optional		Presenter of the slide. A slide can have only one presenter. This name must match one of the presenter names in the cfpresenter tag.
scale	Optional	1.0	Scale used for the HTML content in the slide presentation. If you do not specify the scale, ColdFusion automatically scales the content to fit in the slide.
src	Optional		HTML or SWF source files used as a slide. You can specify the following as the slide source: • An absolute path • A path relative to the CFM page • A URL: Specify if the source returns HTML content SWF files must be present on the system running ColdFusion and the path must be either an absolute path or a path relative to the CFM page. If you do not specify a source file, you must include HTML/CFML code as the body. If you specify a source file and HTML /CFML, ColdFusion ignores the source file and displays the HTML/CFML content in the slide.
title	Optional		Title of the slide

Attribute	Req/Opt	Default	Description
userAgent	Optional	ColdFusion	Text to put in the HTTP User-Agent request header field. Identifies the request client software.
video	Optional		Video file used for the presenter of the slide. If you specify video for the slide and an image for the presenter, the video is used instead of the image for the slide. You cannot specify both audio and video for a slide. The video must be an FLV or SWF file. The video file pathname must be relative to the CFM page or the web root.

Usage

Use the cfpresentationslide tag within the cfpresentation tag to create a slide presentation from individual SWF or HTML source files. If you do not specify a source file, you must include the HMTL or CFML code for the body of the slide within the cfpresentationslide tag. You can assign one presenter to each slide. Use the cfpresenter tag to define presenters referenced by the cfpresentationslide tags.

The following code shows how to create a slide presentation from existing SWF files:

```
<!--- The following example shows how to create a slide presentation --->
<!--- from individual SWF files located on the ColdFusion server. --->
<!--- Because no directory is specified, the presentation runs in --->
<!--- the browser. --->
<cfpresentation title="myPresentation">
    <cfpresentationslide title="1st slide" src="slide1.swf" duration="10"/>
    <cfpresentationslide title="2nd slide" src="slide2.swf"
        audio="audio1.mp3" duration="20"/>
    <cfpresentationslide title="3rd slide" src="slide3.swf"
        audio="audio2.mp3" duration="218"/>
</cfpresentation>
```

Note: *The cfpresentationslide tag requires an end tag. If you specify a source file as the content for the slide instead of CFML and HTML code within start and end tags, use the end slash as a shortcut for the end tag.*

You can reference source files from a URL as long as they return HTML content. The following code shows how to create a slide presentation from HTML files located on an external website:

```
<!--- The following example shows how to create a slide presentation --->
<!--- from HTML files located on an external site. --->
<cfpresentation title="USGS Naming Conventions" directory="myPresenation">
    <cfpresenter name="Robert L. Payne" title="Executive Secretary">
    <cfpresenter name="Trent Palmer" title="Executive Secretary Foreign Names">
    </cfpresentationslide>
    <cfpresentationslide src="http://geonames.usgs.gov/index.html"
        duration="10" presenter="Robert L. Payne"/>
    <cfpresentationslide src="http://geonames.usgs.gov/domestic/index.html"
        duration="15" presenter="Robert L. Payne"/>
    <cfpresentationslide src="http://geonames.usgs.gov/foreign/index.html"
        duration="15" presenter="Trent Palmer"/>
    </cfpresentation>
```

Note: *The links within slides created from HTML files are not active.*

Also, you can enter HTML and CFML code as the body for a slide. Within the code, you can include charts, graphs, and images, as the following code shows:

```
<!--- This example shows how to create a slide presentation dynamically --->
<!--- from HTML code and CFML code. Because no directory is specified, --->
<!--- the presenation runs in the client browser. --->
<cfpresentation title="Sales Presentation">
    <cfpresenter name="Shyam" title="Vice President" email="shyam@somecompany.com">
```

```
    <cfpresenter name="Ram" title="Sr. Vice President" email="ram@somecompany.com">
    <cfpresentationslide title="Introduction" presenter="Shyam" audio="myAudio3.mp3"
            duration="10">
        <h3>Introduction</h3>
        <table>
            <tr>
                <td>
                    <ul>
                        <li>Overview</li>
                        <li>Q1 Sales Figures</li>
                        <li>Projected Sales</li>
                        <li>Competition</li>
                        <li>Advantages</li>
                        <li>Long Term Growth</li>
                    </ul>
                </td>
                <td><img src="../cfdocs/images/artgallery/paul01.jpg"/></td>
            </tr>
        </table>
    </cfpresentationslide>
    <cfpresentationslide Title="Q1 Sales Figures" duration="14" presenter="Ram"
            audio="myAudio1.mp3">
        <h3>Q1 Sales Figures</h3>
        <cfchart format="png" showborder="yes" chartheight="250" chartwidth="300"
                pieslicestyle="sliced">
            <cfchartseries type="pie">
                <cfchartdata item="Europe" value="9">
                <cfchartdata item="Asia" value="20">
                <cfchartdata item="North America" value="50">
                <cfchartdata item="South America" value="21">
            </cfchartseries>
        </cfchart>
    </cfpresentationslide>
</cfpresentation>
```

Example

```
<!--- The following example shows how to create a slide presentation --->
<!--- dynamically from HTML in ColdFusion and from HTML files located --->
<!--- on an external site. ColdFusion writes the presentation files --->
<!--- to a directory relative to the CFM page. --->
<cfpresentation title="USGS Naming Conventions" directory="namingConventions">
    <cfpresenter name="Robert L. Payne" title="Executive Secretary">
    <cfpresenter name="Trent Palmer" title="Executive Secretary Foreign Names">
    <cfpresentationslide presenter="Robert L. Payne">
    <h3>USGS Naming Conventions</h3>
    <ul>
        <li>Overview</li>
        <li>General Naming Conventions</li>
        <li>Domestic Naming Conventions</li>
        <li>Foreign Naming Conventions</li>
    </ul>
    <p></p>
    </cfpresentationslide>
        duration="10" presenter="Robert L. Payne"/>
    <cfpresentationslide src="http://geonames.usgs.gov/domestic/index.html"
        duration="15" presenter="Robert L. Payne"/>
    <cfpresentationslide src="http://geonames.usgs.gov/foreign/index.html"
        duration="15" presenter="Trent Palmer"/>
</cfpresentation>
```

cfpresenter

Description

Describes a presenter in a slide presentation. A slide presentation can have multiple presenters. The presenters must be referenced from the slides defined by the `cfpresentationslide` tag.

History

ColdFusion 8: Added this tag.

Category

Data output tags

Syntax

```
<cfpresenter
    biography = "text string"
    email = "e-mail address of the presenter"
    image = "relative pathname for JPG"
    name = "text string"
    logo = "relative pathname for JPG"
    title = "text string">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfchart`, `cfpresentation`, `cfpresentationslide`

Attributes

Attribute	Req/Opt	Default	Description
`biography`	Optional		A text string that provides information about the presenter; for example, "Sally Maverick has been a top seller of Adobe products for the last five years."
`email`	Optional		E-mail address of the presenter. This attribute activates the Contact link in the presentation control panel, which opens an e-mail message when you click on it.
`image`	Optional		Pathname for the presenter's image in JPEG format. The JPEG file must be relative to the CFM page. If you specify a video for the `cfpresentationslide` tag, the video clip overrides this value for that slide.
`name`	Required		Name of the presenter. Use this value in the `presenter` attribute of the `cfpresentationslide` tag to associate the presenter with the slide.
`logo`	Optional		Pathname of the image file that represents the presenter's logo or the logo of the presenter's organization. The logo must be in JPEG format. The file must be relative to the CFM file website.
`title`	Optional		Title of the presenter, for example, "VP of Sales".

Usage

Use the `cfpresenter` tag to define the presenters that you specify for each slide. The presenter information appears in the control panel for the slide to which it is assigned. To specify a presenter for a slide, use the `presenter` attribute of the `cfpresentationslide` tag.

You can specify an image of the presenter and the presenter's company logo by using the `image` and `logo` attributes of the `presenter` tag, respectively. To display a video clip in place of the presenter's image, you can specify an FLV or SWF file for `video` attribute of the `cfpresentationslide` tag.

Example

```
<!--- This example shows how to specify presenters for a slide --->
<!--- presentation and assign a presenter to each slide in the presentation. --->
<cfpresentation title="myPresentation" directory="presentation" overwrite="yes">

<!--- The following code defines three presenters. --->
<cfpresenter name="Shyam" title="President" email="shyam@somecompany.com"
    image="images\shyam01.jpg">
<cfpresenter name="Ram" title="V.P. Sales" email="ram@somecompany.com"
    image="images\ram01.jpg">
<cfpresenter name="Michelle" title="V.P. Engineering"
    email="mhatter@adobe.com" image="images\michelle01.jpg">

<!--- The following code assigns a presenter to each of three slides in the presentation. --->
<cfpresentationslide title="myFirstSlide" src="slide1.swf" duration="10"
    presenter="Shyam"/>
<cfpresentationslide title="mySecondSlide" src="slide2.swf" duration="15"
    presenter="Michelle"/>
<cfpresentationslide title="myThirdSlide" src="slide3.swf" duration="2"
    presenter="Ram"/>

<!--- In the following slide, ColdFusion uses a video clip --->
<!--- instead of the JPEG image for the presenter. --->
<cfpresentationslide title="myFourthSlide" src="slide4.swf" duration="5"
    presenter="Shyam" video="video\video1.flv"/>
</cfpresentation>
```

cfprint

Description

Prints specified pages from a PDF file. Use this tag to perform automated batch print jobs. Use the `cfprint` tag to print any PDF document, including those generated by the `cfdocument`, `cfpdf`, and `cfpdfform` tag. Also, you use this tag to print Report Builder reports exported in PDF format.

History

ColdFusion 8: Added this tag.

Category

Data output tags

Syntax

```
<cfprint
    source = "absolute or relative pathname to a PDF file|PDF document variable"
    type = "PDF"
    attributeStruct = "ColdFusion structure that contains standard print request
        key-value pairs"
    color = "yes|no"
    copies = "number of copies"
    fidelity = "yes|no"
    pages = "page or pages to print"
    password = "PDF source file owner or user password"
    paper = "letter|legal|A4|A5|B4|B5|B4-JIS|B5-JIS|any media supported by the printer"
    printer = "string that specifies the printer name">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfdocument`, `cfpdf`, `cfpdfform`, `cfpdfformparam`, `cfpdfparam`, `cfpdfsubform`, `GetPrinterInfo`, `IsPDFFile`, `IsPDFObject`

Attributes

Attribute	Req/Opt	Default	Description
`attributeStruct`	Optional		ColdFusion structure used to specify additional print instructions. Individually named attributes take precedence over the key-value pairs in the attribute structure. For information about key-value pairs, see the table in the section "attributeStruct" on page 483.
`color`	Optional		Color or monochrome printing: • `yes`: print in color • `no`: print in black and white, with colors in shades of gray
`copies`	Optional		Number of copies to print. The value must be greater than or equal to 1.
`fidelity`	Optional	`no`	Whether to print a job based on the print requirements specified. Valid values are: • `yes`: if the job cannot be printed exactly as specified in the print requirements, the job is rejected. • `no`: a reasonable attempt to print the job is acceptable

Attribute	Req/Opt	Default	Description
pages	Optional	all	Pages in the source file to print. Duplicate pages and pages beyond the total count of pages in the document are ignored as long as there is at least one page between 1 and the total number of pages in the document. You can combine individual page numbers and page ranges, for example, 1—3,6,10—20. If you do not specify a value for the pages attribute, ColdFusion prints the entire document.
paper	Optional		Paper used for the print job. The value can be any returned by the GetPrinterInfo function. The following values are valid: • na-letter • na-legal • iso-a4 • iso-a5 • iso-b4 • iso-b5 • jis-b4 • jis-b5 For more information, see "Supported paper types" on page 482.
password	Optional		The owner or user password for the PDF source file. If the PDF file is password-protected, you must specify this attribute for the file to print.
printer	Optional		The name of a printer. An example in Windows is \\s1001prn02\NTN-2W-HP_BW02. The default name is the default printer for the account where the ColdFusion server is running. Printer names are case-sensitive and must be entered exactly as they appear in the System Information page of the ColdFusion Administrator. For more information, see Usage.
source	Required		Source document to print. Specify one of the following: • An absolute or relative pathname to a PDF document, for example, c:\work\myPDF.pdf or myPDF.pdf. The default directory is the template directory. • A PDF document variable in memory that is generated by the cfdocument tag or the cfpdf tag, for example, "myPDFdoc".
type	Required	PDF	The file type of the document being printed. The only valid value is PDF.

Usage

Use the cfprint tag for automated batch printing of PDF documents. For example, you can run a batch job each evening that generates a report in PDF format and then prints either the entire report or selected pages for review the next morning without user intervention.

Most of the cfprint tag attributes are printer-dependent. If a printer does not support a specified attribute, it ignores the instruction. The default settings for the attributes also are printer-dependent. If you set a default printer, only specify the PDF file source and the password, if the file is password-protected.

Note: *The particular printer attributes supported are dependent on the operating system, the network printer server, if there is one, and the printer. The cfprint tag is dependent on the Java Print Service (JPS). Many printers support attributes that are not accessible from JPS. For example, the JPS for a Macintosh OSX running JDK 1.5 supports the fewest printer attributes. Upgrading to JDK 1.6 adds some functionality, but finishing attributes are still not supported.*

If the fidelity attribute is set to yes, the job does not print if any of the specified attributes are not supported by the printer. If the fidelity attribute is set to no, the printer accepts the print job and either ignores any attribute it does not support or substitutes a reasonable alternative for the attribute.

To determine which attributes are supported on a specified printer, use the GetPrinterInfo function.

Supported paper types

You can use the equivalent page types supported by the cfdocument tag, but they are not returned by the GetPrinterInfo function:

cfdocument	cfprint
• letter	• na-letter
• legal	• na-legal
• A4	• iso-a4
• A5	• iso-a5
• B4	• iso-b4
• B5	• iso-b5
• B4-JIS	• jis-b4
• B5-JIS	• jis-b5

View a list of configured printers

1 Log on to the ColdFusion Administrator.

2 Click on the System Information icon located at the top right of the Administrator Console window. (The icon has an “i” on it.)

3 Scroll to the bottom of the System Information page. Under Printer Details is the Default Printer. Below the default printer is Printers, which lists the configured printers available to ColdFusion, including the default printer.

Printer configuration is operating system-dependent. Configure printers outside of ColdFusion.

View the print log

1 Log on to the ColdFusion Administrator.

2 Expand the Debugging and Logging topic.

3 Click the Log Files link. The print.log file appears in the list of log files.

Permissions for printing

If the PDF file is encrypted, the permissions for the file must be set to AllowPrinting, or you must specify the owner password to print the file. Use the protect action of the cfpdf tag to set permissions and passwords on PDF files. For more information, see “Permissions for PDF documents” on page 445.

If a Security Manager is installed, the following permission is required in the coldfusion.policy file to initiate a print job request:

```
grant { permission java.lang.RuntimePermission "queuePrintJob"; };
```

In Windows systems, the account running the ColdFusion server must have PRINTER_ACCESS_USE access rights for each printer it uses. Even if the printer is configured locally on the system, the printer is not available if the account in which ColdFusion is running does not have the proper permissions.

Note: *By default, ColdFusion installs and runs as the Local System account, which may not have printer queue access rights. For information on running ColdFusion as a specific user, see the following Tech Note:*
http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=tn_17279

attributeStruct

The following table lists the optional `attributeStruct` key-value pairs that you use to specify print requests:

Element	Description
`autoRotateAndCenter`	Adjusts the document's orientation to match the orientation specified in the printer attributes and centers the page in the imaging area: • `yes`: the orientation, if specified, is ignored (default) • `no`: the orientation, if specified, is applied to the document
`collate` or `sheetCollate`	Specifies whether the sheets of each copy of each printed document in a job are in sequence when multiple copies of the document are specified by the `copies` attribute: • `yes` • `no`
`color` or `chromaticity`	Specifies color or monochrome printing. Monochrome printing displays colors in shades of gray: • `yes`: print in color. • `no`: print in monochrome.
`copies`	Number of copies of the source document to print. Valid values are integers greater than or equal to 1.
`coverPage` or `jobSheets`	Specifies which job start and end sheets, if any, are printed with a job: • `none` • `standard`
`fidelity`	Specifies whether to print a job based on the print requirements specified. The following values are valid values: • `yes`: If the job cannot be printed exactly as specified in the print requirements, the job is rejected. • `no`: A reasonable attempt to print the job is acceptable (default).
`finishings`	Finishing operation to perform after each copy of a document is printed: • `staple-top-left` • `staple-bottom-left` • `staple-top-right` • `staple-bottom-right` • `edge-stitch-left` • `edge-stitch-right` • `edge-stitch-top` • `edge-stitch-bottom` • `dual-right` • `dual-top` • `dual-bottom` • `dual-left`

Element	Description
jobHoldUntil	Date-time attribute for the exact date and time at which the job is available for printing. Valid values are ColdFusion date and time variables.
jobName	The name of a print job.
jobPriority	Integer value that represents a print job's priority. Among those jobs that are ready to print, a printer must print all jobs with a priority value of *n* before printing those with a priority value of *n*-1 for all *n*. Valid values are integers from 1 (lowest priority) through 100 (highest priority).
numberUp	Number of pages to print on a single side of paper. The value must be a number greater than or equal to 1.
orientation or orientationRequested	Orientation of the page to be printed. The only valid value for PDF documents is portrait. To change the orientation to landscape, set the autoRotateAndCenter to yes (which is the default value). The autoRotateAndCenter instruction overrides the orientation instruction.
pages	Pages in the source file to print. Duplicate pages and pages beyond the total count of pages in the document are ignored as long as there is at least one page between 1 and the total number of pages in the document. You can combine individual page numbers and page ranges,for example, 1—3,6,10—20. If you do not specify a value for the pages attribute, ColdFusion prints the entire document.
pageScaling	Specifies how pages are scaled on the paper: • fit-to-printer-margins: Reduces or enlarges each page to fit the printable area of the currently selected paper size (Default). • reduce-to-printer-margins: Shrinks large pages to fit the currently selected paper size but does not enlarge small pages. If an area is selected and is larger than the printable area of the currently selected paper, the page is scaled to fit the printable area. • none: Prints the upper left or center of a page (if autorotated and centered) without scaling. Pages that don't fit on the paper are cropped.
pageSubset	Prints a subset of pages in specified by the pages attribute: • all: Prints all the pages in the specified page range (Default). • odd: Prints only the odd pages in the specified page range. • even: Prints only the even pages in the specified page range.
paper	Paper used for the print job. The value can be any returned by the GetPrinterInfo function. The following values are the most common: • na-letter • iso-a4
presentationDirection	Used in conjunction with the numberUp attribute to indicate the layout of multiple document pages on one side of the paper.
printer	The name of a printer. An example in Windows is \\s1001prn02\NTN-2W-HP_BW02. The default name is the default printer for the account where the ColdFusion server is running. Printer names are case-sensitive and you must enter the names exactly as they appear in the System Information page of the ColdFusion Administrator. For more information on viewing print logs, see Usage.
quality	Print quality for the print job: • draft • high • normal
requestingUserName	A string that specifies the name of the end user that submitted the print job.

Element	Description
reversePages	Prints pages in reverse order. If page ranges are entered, the pages print opposite of the order in which they were entered. For example, if the Pages box shows 3-5, 7-10, selecting Reverse Pages prints pages 10-7, and then 5-3. • yes • no (default)
sides	Sides of the paper on which the pages are printed: • one-sided (default) • duplex *or* two-sided-long-edge • tumble *or* two-sided-short-edge
usePdfPageSize	Uses the PDF page size to determine the area of the paper printed rather than the paper size. This is useful for printing PDF documents that contain multiple page sizes: • yes • no (default)

Example

The following example shows how to use the attributeStruct attribute and the cfprint tag to print five, double-sided copies of a letter-sized PDF document, which are stapled on the top-left corner and collated:

```
<cfset aset=StructNew()>
<cfset aset["sides"] = "duplex">
<cfprint type="pdf" source="myfile.pdf"
    printer="\\s1001prn02\NTN-2W-HP_BW02" copies="5" paper="letter"
    attributeStruct="#aset#">
```

The following example shows how to specify all of the print attributes with the attributeStruct attribute:

```
<cfset aset=StructNew()>
<cfset aset["paper"] = "letter">
<cfset aset["sides"] = "duplex">
<cfset aset["copies"] = "5">
<cfset aset["printer"] = "\\s1001prn02\NTN-2W-HP_BW02">

<cfprint type="pdf" source="myfile.pdf" attributeStruct="#aset#">
```

Printers have a setting called autoRotateAndCenter, which is set to yes by default. The following example shows how to override the default autoRotatateAndCenter setting and use the orientation setting instead:

```
<cfset aset=StructNew()>
    <cfset aset["autoRotateAndCenter"] = "no">
    <cfset aset["orientation"] = "portrait">

    <cfprint printer="myprinter" source="_mydoc.pdf" attributeStruct="#aset#">
```

To run a print job asynchronously, start a print job in a thread. Do not wait for the print job to be sent to the printer before proceeding. To start a print job in a thread, enclose the cfprint tag within cfthread start and end tags, as the following example shows:

```
<cfthread name="mythread" action="run">
    <cfprint type="pdf" source="myfile.pdf" printer="\\s1001prn02\NTN-2W-HP_BW02">
</cfthread>
....
```

For more information, see "cfthread" on page 581.

cfprocessingdirective

Description

Provides the following information to ColdFusion about how to process the current page:

- Specifies whether to remove excess whitespace character from ColdFusion generated content in the tag body.
- Identifies the character encoding (character set) of the page contents.

Category

Data output tags, Page processing tags

Syntax

```
<cfprocessingdirective
    pageencoding = "page-encoding literal string"/>

OR

<cfprocessingdirective
    pageEncoding = "page-encoding literal string"
    suppressWhiteSpace = "yes|no">
    CFML tags
</cfprocessingdirective>
```

See also

cfcol, cfcontent, cfoutput, cfsetting, cfsilent, cftable, SetEncoding; "Developing Globalized Applications" on page 337 in the *ColdFusion Developer's Guide*

History

ColdFusion MX:

- Changed `suppresswhitespace` attribute value behavior: you can specify the `suppresswhitespace` attribute value as a string variable. (ColdFusion 5 supported setting it only as a constant.)
- Added the `pageEncoding` attribute.

Attributes

Attribute	Req/Opt	Default	Description
pageEncoding	Optional	Character encoding identified by the page byte order mark, if any; otherwise, system default encoding	A string literal; cannot be a variable. Identifies the character encoding of the current CFML page. This attribute affects the entire page, not just the cfprocessing tag body. The value may be enclosed in single- or double-quotation marks, or none. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information on character encodings, see www.w3.org/International/O-charset.html.
suppressWhiteSpace	Optional		Boolean; whether to suppress white space characters within the cfprocessingdirective block that are generated by CFML tags and often do not affect HTML appearance. Does not affect any white space in HTML code.

Usage

The cfprocssingdirective tag has limitations that depend on the attribute you use. For this reason, Adobe recommends that you include either the pageencoding or suppresswhitespace attribute in a cfprocessingdirective tag, not both. To specify both values, use separate tags.

In a ColdFusion component (.cfc file), the cfprocessingdirective tag must follow the cfcomponent tag.

If you use the pageEncoding attribute, the following rules apply:

- You must put the tag within the first 4096 bytes of a page. It can be positioned after a cfsetting or cfsilent tag.
- If you use the tag on a page that includes other pages by using the cfinclude or cfmodule tags, custom tag invocation, and so on, the tag has no effect on the included pages.
- You cannot embed the tag within conditional logic, because the pageEncoding attribute is evaluated when ColdFusion compiles a page (not when it executes the page). For example, the following code has no effect at execution time, because the cfprocessingdirective tag has already been evaluated:

```
<cfif dynEncoding is not "dynamic encoding is not possible">
    <cfprocessingdirective pageencoding=#dynEncoding#>
</cfif>
```

- If you have multiple cfprocessingdirective tags in one page that specify the pageEncoding attribute, they must all specify the same value; if not, ColdFusion throws an error.
- If you specify only the pageencoding attribute, do not use a separate end tag.

• ColdFusion accepts character encoding names that are supported by the Java platform. If an invalid name is specified, ColdFusion throws an InvalidEncodingSpecification exception.

• If a page has a byte order mark (BOM), and a `pageencoding` attribute specifies an encoding that differs from the BOM, ColdFusion generates an error.

The following rules apply to the `suppressWhiteSpace` attribute:

• You can specify the `suppresswhitespace` attribute value as a constant or a variable. To use a variable: define the variable (for example, `whitespaceSetting`), assign it the value yes or no, and code a statement such as the following:

```
<!--- ColdFusion allows suppression option to be set at runtime --->
<cfprocessingdirective suppresswhitespace=#whitespaceSetting#>
code to whose output the setting is applied
</cfprocessingdirective>
```

• The `suppresswhitespace` attribute only affects code that you put between the `<cfprocessingdirective>` begin tag and the `</cfprocessingdirective>` end tag.

The following example shows the use of a nested `cfprocessingdirective` tag. The outer tag suppresses unnecessary whitespace during computation of a large table; the inner tag retains whitespace, to output a preformatted table.

Example

```
<cfprocessingdirective suppressWhiteSpace = "Yes">
    <!--- CFML code --->
    <cfprocessingdirective suppressWhiteSpace = "No">
        <cfoutput>#table_data#
        </cfoutput>
    </cfprocessingdirective>
</cfprocessingdirective>
```

The following example shows the use of the `pageencoding` attribute:

```
<cfprocessingdirective pageencoding = "shift_jis">
```

cfprocparam

Description

Defines stored procedure parameters. This tag is nested within a `cfstoredproc` tag.

Category

Database manipulation tags

Syntax

```
<cfprocparam
    CFSQLType = "parameter data type"
    maxLength = "length"
    null = "yes|no"
    scale = "decimal places"
    type = "in|out|inout"
    value = "parameter value"
    variable = "variable name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfinsert`, `cfprocresult`, `cfquery`, `cfqueryparam`, `cfstoredproc`, `cftransaction`, `cfupdate`; "Optimizing ColdFusion applications" on page 239 in "Designing and Optimizing a ColdFusion Application" on page 219 in the *ColdFusion Developer's Guide*

History

ColdFusion MX:

- The `maxrows` attribute is obsolete.
- Changed the `dbvarname` attribute behavior: it is now ignored for all drivers. ColdFusion uses JDBC 2.2 and does not support named parameters.
- Changed the `maxLength` attribute behavior: it now applies to IN and INOUT parameter values.

Attributes

Attribute	Req/Opt	Default	Description
CFSQLType	Required		SQL type to which the parameter (any type) is bound. ColdFusion supports the following values, where the last element of the name corresponds to the SQL data type. Different database systems might support different subsets of this list. For information on supported parameter types, see your DBMS documentation. • CF_SQL_BIGINT • CF_SQL_BIT • CF_SQL_BLOB • CF_SQL_CHAR • CF_SQL_CLOB • 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_REFCURSOR • CF_SQL_SMALLINT • CF_SQL_TIME • CF_SQL_TIMESTAMP • CF_SQL_TINYINT • CF_SQL_VARCHAR For a mapping of ColdFusion SQL data types to JDBC data types, see cfquery-param.
maxLength	Optional	0	Maximum length of a string or character IN or INOUT value attribute. A maxLength of 0 allows any length. The maxLength attribute is not required when specifying type=out.
null	Optional	no	Whether the parameter is passed in as a null value. Not used with OUT type parameters. • yes: tag ignores the value attribute. • no
scale	Optional	0	Number of decimal places in numeric parameter. A scale of 0 limits the value to an integer.

Attribute	Req/Opt	Default	Description
type	Optional	in	• in: the parameter is used to send data to the database system only. Passes the parameter by value. • out: the parameter is used to receive data from the database system only. Passes the parameter as a bound variable. • inout: the parameter is used to send and receive data. Passes the parameter as a bound variable.
value	Required if type = "IN"		Value that ColdFusion passes to the stored procedure.This is optional for inout parameters.
variable	Required if type = "OUT" or "INOUT"		ColdFusion variable name; references the value that the output parameter has after the stored procedure is called. This is ignored for in parameters.

Usage

Use this tag to identify stored procedure parameters and their data types. Code one cfprocparam tag for each parameter. The parameters that you code vary based on parameter type and DBMS. ColdFusion supports both positional and named parameters. If you use positional parameters, you must code cfprocparam tags in the same order as the associated parameters in the stored procedure definition.

Output variables are stored in the ColdFusion variable specified by the variable attribute.

You cannot use the cfprocparam tag for Oracle 8 and 9 reference cursors. Instead, use the cfprocresult tag.

Example

The following examples lists the equivalent Oracle and Microsoft SQL Server stored procedures that insert data into the database. The CFML to invoke either stored procedure is the same.

The following example shows the Oracle stored procedure:

```
CREATE OR REPLACE PROCEDURE Insert_Book (
    arg_Title Books.Title%type,
    arg_Price Books.Price%type,
    arg_PublishDate Books.PublishDate%type,
    arg_BookID OUT Books.BookID%type)
AS
    num_BookID NUMBER;
BEGIN
    SELECT seq_Books.NEXTVAL
    INTO num_BookID
    FROM DUAL;

    INSERT INTO
        Books (
            BookID,
            Title,
            Price,
            PublishDate )
    VALUES (
        num_BookID,
        arg_Title,
        arg_Price,
        arg_PublishDate );

    arg_BookID := num_BookID;
END;
/
```

The following example shows the SQL Server stored procedure:

```
CREATE PROCEDURE Insert_Book (
    @arg_Title VARCHAR(255),
    @arg_Price SMALLMONEY,
    @arg_PublishDate DATETIME,
    @arg_BookID INT OUT)
AS
BEGIN
INSERT INTO
    Books (
        Title,
        Price,
        PublishDate )
    VALUES (
        @arg_Title,
        @arg_Price,
        @arg_PublishDate );

    SELECT @arg_BookID = @@IDENTITY;
END;
```

You use the following CFML code to call either stored procedure:

```
<cfset ds = "sqltst">
<!--- <cfset ds = "oratst"> --->

<!--- If submitting a new book, insert the record and display confirmation --->
<cfif isDefined("form.title")>
    <cfstoredproc procedure="Insert_Book" datasource="#ds#">
        <cfprocparam cfsqltype="cf_sql_varchar" value="#form.title#">
        <cfprocparam cfsqltype="cf_sql_numeric" value="#form.price#">
        <cfprocparam cfsqltype="cf_sql_date" value="#form.price#">
        <cfprocparam cfsqltype="cf_sql_numeric" type="out" variable="bookId">
    </cfstoredproc>

<cfoutput>
    <h3>'#form.title#' inserted into database.The ID is #bookId#.</h3>
</cfoutput>

</cfif>
<cfform action="#CGI.SCRIPT_NAME#" method="post">
    <h3>Insert a new book</h3>

    Title:
    <cfinput type="text" size="20" required="yes" name="title"/>
    <br/>

    Price:
    <cfinput type="text" size="20" required="yes" name="price" validate="float"/>
    <br/>

    Publish Date:
    <cfinput type="text" size="5" required="yes" name="publishDate" validate="date"/>
    <br/>

    <input type="submit" value="Insert Book"/>

</cfform>
```

cfprocresult

Description

Associates a query object with a result set returned by a stored procedure. Other ColdFusion tags, such as `cfoutput` and `cftable`, use this query object to access the result set. This tag is nested within a `cfstoredproc` tag.

Category

Database manipulation tags

Syntax

```
<cfprocresult
    name = "query name"
    maxRows = "number"
    resultSet = "1-n">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfinsert`, `cfprocparam`, `cfquery`, `cfqueryparam`, `cfstoredproc`, `cftransaction`, `cfupdate`; "Optimizing database use" on page 243 in "Designing and Optimizing a ColdFusion Application" on page 219 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		Name for the query result set.
`maxRows`	Optional	-1 (All)	Maximum number of rows returned in result set.
`resultSet`	Optional	1	Names one result set, if stored procedure returns more than one.

Usage

To enable access to data returned by the stored procedure, specify one or more `cfprocresult` tags. If the stored procedure returns more than one result set, use the `resultSet` attribute to specify which of the stored procedure's result sets to return.

The `resultSet` attribute must be unique within the scope of the `cfstoredproc` tag. If you specify a result set twice, the second occurrence overwrites the first.

CFML supports Oracle 8 and 9 Reference Cursor type, which passes a parameter by reference. Parameters that are passed this way can be allocated and deallocated from memory within the execution of one application. To use reference cursors in packages or stored procedures, use the `cfprocresult` tag. This causes the ColdFusion JDBC database driver to put Oracle reference cursors into a result set. (You cannot use this method with Oracle's ThinClient JDBC drivers.)

Example

```
<!--- This example executes a Sybase stored procedure that returns three result sets, two
    of which we want. The stored procedure returns status code and one output parameter, which
    we display. We use named notation for parameters. --->
<!--- cfstoredproc tag --->
<cfstoredproc procedure = "foo_proc"
    dataSource = "MY_SYBASE_TEST" username = "sa"
    password = "" dbServer = "scup" dbName = "pubs2"
    returnCode = "Yes" debug = "Yes">
```

```
    <!--- cfprocresult tags --->
    <cfprocresult name = RS1>
    <cfprocresult name = RS3 resultSet = 3>
    <!--- cfprocparam tags --->
    <cfprocparam type = "IN" CFSQLType = CF_SQL_INTEGER value = "1">
    <cfprocparam type = "OUT" CFSQLType = CF_SQL_DATE variable = FOO>
    <!--- Close the cfstoredproc tag. --->
</cfstoredproc>
<cfoutput>
    The output param value: '#foo#'<br>
</cfoutput>
<h3>The Results Information</h3>
<cfoutput query = RS1>#name#,#DATE_COL#<br>
</cfoutput>
<p></p>
<cfoutput>
    <hr>
    <p>Record Count: #RS1.recordCount# <p>Columns: #RS1.columnList#</p>
    <hr>
</cfoutput>
<cfoutput query = RS3>#col1#,#col2#,#col3#<br>
</cfoutput>
<p></p>
<cfoutput>
    <hr>
    <p>Record Count: #RS3.recordCount# <p>Columns: #RS3.columnList#</p>
    <hr>
    The return code for the stored procedure is:
    '#cfstoredproc.statusCode#'<br>
</cfoutput>
...
```

cfproperty

Description

Defines properties of a ColdFusion component (CFC). Used to create complex data types for web services. The attributes of this tag are exposed as component metadata and are subject to inheritance rules.

Category

Extensibility tags

Syntax

```
<cfproperty
    name="name"
    default="default value"
    displayname="descriptive name"
    hint="extended description"
    required="boolean"
    type="type">
```

See also

cfargument, cfcomponent, cffunction, cfinvoke, cfinvokeargument, cfobject, cfreturn; "Documenting CFCs" on page 168 in "Building and Using ColdFusion Components" on page 158 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
name	Required		A string; a property name. Must be a static value.
default	Optional		If no property value is set when the component is used for a web service, specifies a default value. If this attribute is present, the required attribute must be set to no or not specified.
displayname	Optional		A value to be displayed when using introspection to show information about the CFC. The value appears in parentheses following the property name.
hint	Optional		Text to be displayed when using introspection to show information about the CFC. This attribute can be useful for describing the purpose of the parameter.

Attribute	Req/Opt	Default	Description
required	Optional	no	Whether the parameter is required: • yes • no
type	Optional	any	A string; identifies the property data type: • any • array • binary • boolean • date • guid: the argument must be a UUID or GUID of the form *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* where each *x* is a character representing a hexadecimal number (0-9A-F). • numeric • query • string • struct • uuid: The argument must be a ColdFusion UUID of the form *xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx* where each *x* is a character representing a hexadecimal number (0-9A-F). • variableName: a string formatted according to ColdFusion variable naming conventions. • a component name: if the type attribute value is not one of the preceding items, ColdFusion treats it as the name of a ColdFusion component. When the function executes, it generates an error if the argument that is passed in is not a CFC with the specified name.

Usage

You must position cfproperty tags at the beginning of a component, above executable code and function definitions.

If a component is not used as a web service, The cfproperty only provides metadata information when the component is viewed using introspection, for example, by opening the CFC file directly in the browser. It does not define variables or set values that you can then use in your component.

For web services that you create in ColdFusion, the cfproperty tag defines complex variables used by the web service.

Example

The following code defines a component in the file address.cfc that contains properties that represent a street address:

```
<cfcomponent>
    <cfproperty name="Number" type="numeric">
    <cfproperty name="Street" type="string">
    <cfproperty name="City" type="string">
    <cfproperty name="State" type="string">
    <cfproperty name="Country" type="string">
</cfcomponent>
```

This component represents a complex data type that can be used in a component that is exported as a web service, such as the following:

```
<cfcomponent>
    <cffunction name="echoAddress" returnType="address" access="remote">
        <cfargument name="input" type="address">
        <cfreturn arguments.input>
    </cffunction>
</cfcomponent>
```

cfquery

Description

Passes queries or SQL statements to a data source.

Adobe recommends that you use the `cfqueryparam` tag within every `cfquery` tag, to help secure your databases from unauthorized users. For more information, see Security Bulletin ASB99-04, "Multiple SQL Statements in Dynamic Queries," in the Security Zone, www.adobe.com/devnet/security/security_zone/asb99-04.html, and “Accessing and Retrieving Data” on page 393 in the *ColdFusion Developer's Guide*.

Category

Database manipulation tags

Syntax

```
<cfquery
    name = "query name"
    blockFactor = "block size"
    cachedAfter = "date"
    cachedWithin = "timespan"
    dataSource = "data source name"
    dbtype = "query"
    debug = "yes|no"
    maxRows = "number"
    password = "password"
    result = "result name"
    timeout = "seconds"
    username = "user name">
</cfquery>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfdbinfo`, `cfinsert`, `cfprocparam`, `cfprocresult`, `cfqueryparam`, `cfstoredproc`, `cftransaction`, `cfupdate`; “Optimizing database use” on page 243 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the result variable that specifies the ID of a row.

ColdFusion MX 7:

- Added the `result` attribute for specifying an alternate name for the structure that holds the result variables.
- Added result variables for the SQL statement executed (`sql`), the number of records returned (`recordcount`), whether the query was cached (`cached`), an array of `cfqueryparam` values (`sqlparameters`), and the list of columns in the returned query (`columnlist`).

ColdFusion MX:

- Changed Query of Queries behavior: it now supports a larger subset of standard SQL.
- Changed dot notation support: ColdFusion now supports dot notation within a record set name. ColdFusion interprets such a name as a structure.
- Deprecated the `connectString`, `dbName`, `dbServer`, `provider`, `providerDSN`, and `sql` attributes, and all values of the `dbtype` attribute except `query`. They do not work, and might cause an error, in releases later than ColdFusion 5.

- New query object variable: `cfquery.ExecutionTime`.
- No longer supports native drivers. It now uses JDBC (and ODBC-JDBC bridge) for database connectivity.

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		Name of query. Used in page to reference query record set. Must begin with a letter. Can include letters, numbers, and underscores.
`blockFactor`	Optional	1	Maximum rows to get at a time from server. Range: 1 - 100. Might not be supported by some database systems.
`cachedAfter`	Optional		Date value (for example, April 16, 1999, 4-16-99). If date of original query is after this date, ColdFusion uses cached query data. To use cached data, current query must use same SQL statement, data source, query name, user name, password. A date/time object is in the range 100 AD–9999 AD. When specifying a date value as a string, you must enclose it in quotation marks.
`cachedWithin`	Optional		Timespan, using the `CreateTimeSpan` function. If original query date falls within the time span, cached query data is used. `CreateTimeSpan` defines a period from the present, back. Takes effect only if query caching is enabled in the Administrator. To use cached data, the current query must use the same SQL statement, data source, query name, user name, and password.
`dataSource`	Required unless `dbtype=query`.		Name of data source from which query gets data. You must specify either `dbtype` or `dataSource`.
`dbtype`	Optional		Results of a query as input. You must specify either `dbtype` or `dataSource`.
`debug`	Optional; value and equals sign may be omitted		• `yes`, or if omitted: if debugging is enabled, but the Administrator Database Activity option is not enabled, displays SQL submitted to the data source and number of records returned by query. • `no`: if the Administrator Database Activity option is enabled, suppresses display.
`maxRows`	Optional	-1 (All)	Maximum number of rows to return in record set.
`password`	Optional		Overrides the password in the data source setup.
`result`	Optional		Name for the structure in which `cfquery` returns the result variables. For more information, see Usage.
`timeout`			Maximum number of seconds that each action of a query is permitted to execute before returning an error. The cumulative time may exceed this value. For JDBC statements, ColdFusion sets this attribute. For other drivers, see the driver documentation.
`username`	Optional		Overrides user name in the data source setup.

Usage

Use this tag to execute a SQL statement against a ColdFusion data source. Although you can use the `cfquery` tag to execute any SQL Data Definition Language (DDL) or Data Manipulation Language (DML) statement, you typically use it to execute a SQL SELECT statement.

Note: *To call a stored procedure, use the `cfstoredproc` tag.*

This tag creates a query object, providing this information in query variables:

Variable name	Description
query_name.currentRow	Current row of query that cfoutput is processing.
query_name.columnList	Comma-separated list of the query columns.
query_name.RecordCount	Number of records (rows) returned from the query.

The cfquery tag also returns the following result variables in a structure. You can access these variables with a prefix of the name you specified in the result attribute. For example, if you assign the name myResult to the result attribute, you would retrieve the name of the SQL statement that was executed by accessing #myResult.sql#. The result attribute provides a way for functions or CFCs that are called from multiple pages, possibly at the same time, to avoid overwriting results of one call with another. The result variable of INSERT queries contains a key-value pair that is the automatically generated ID of the inserted row; this is available only for databases that support this feature. If more than one record was inserted, the value can be a list of IDs. The key name is database-specific.

Variable name	Description
result_name.sql	The SQL statement that was executed.
result_name.recordcount	Number of records (rows) returned from the query.
result_name.cached	True if the query was cached; False otherwise.
result_name.sqlparameters	An ordered Array of cfqueryparam values.
result_name.columnList	Comma-separated list of the query columns.
result_name.ExecutionTime	Cumulative time required to process the query.
result_name.IDENTITYCOL	SQL Server only. The ID of an inserted row.
result_name.ROWID	Oracle only. The ID of an inserted row. This is not the primary key of the row, although you can retrieve rows based on this ID.
result_name.SYB_IDENTITY	Sybase only. The ID of an inserted row.
result_name.SERIAL_COL	Informix only. The ID of an inserted row.
result_name.GENERATED_KEY	MySQL only. The ID of an inserted row. MySQL 3 does not support this feature.

You can cache query results and execute stored procedures. For information about this and about displaying cfquery output, see the *ColdFusion Developer's Guide*.

Because the timeout attribute only affects the maximum time for each suboperation of a query, the cumulative time may exceed its value. To set a timeout for a page that might get a very large result set, set the Administrator > Server Settings > Timeout Requests option to an appropriate value or use the RequestTimeout attribute of the cfsetting tag (for example, <cfsetting requestTimeout="300">).

The Caching page of the ColdFusion Administrator specifies the maximum number of cached queries. Setting this value to 0 disables query caching.

You cannot use ColdFusion reserved words as query names.

You cannot use SQL reserved words as variable or column names in a Query of Queries, unless they are escaped. The escape character is the bracket []; for example:

```
SELECT [count] FROM MYTABLE.
```

For a list of reserved keywords in ColdFusion, see "Escaping reserved keywords" on page 431 in the *ColdFusion Developer's Guide*.

Example

```
<!--- This example shows the use of CreateTimeSpan with CFQUERY ------>
<!--- to cache a record set. Define startrow and maxrows to ---->
<!--- facilitate 'next N' style browsing. ---->
<cfparam name="MaxRows" default="10">
<cfparam name="StartRow" default="1">
<!---------------------------------------------------------------------
Query database for information if cached database information has
not been updated in the last six hours; otherwise, use cached data.
---------------------------------------------------------------------->
<cfquery
    name="GetParks" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT PARKNAME, REGION, STATE
    FROM Parks
    ORDER BY ParkName, State
</cfquery>
<!--- Build HTML table to display query. ------------------------->
<table cellpadding="1" cellspacing="1">
    <tr>
        <td bgcolor="f0f0f0">
             
        </td>
        <td bgcolor="f0f0f0">
            <b><i>Park Name</i></b>
        </td>
        <td bgcolor="f0f0f0">
            <b><i>Region</i></b>
        </td>
        <td bgcolor="f0f0f0">
            <b><i>State</i></b>
        </td>
    </tr>
<!--- Output the query and define the startrow and maxrows parameters.
Use the query variable CurrentCount to keep track of the row you are displaying. ------>
<cfoutput query="GetParks" startrow="#StartRow#" maxrows="#MaxRows#">
    <tr>
        <td valign="top" bgcolor="ffffed">
            <b>#GetParks.CurrentRow#</b>
        </td>
        <td valign="top">
            <font size="-1">#ParkName#</font>
        </td>
        <td valign="top">
            <font size="-1">#Region#</font>
        </td>
        <td valign="top">
            <font size="-1">#State#</font>
        </td>
    </tr>
</cfoutput>
<!--- If the total number of records is less than or equal to the total number of rows,
then offer a link to the same page, with the startrow value incremented by maxrows
(in the case of this example, incremented by 10). --------->
    <tr>
        <td colspan="4">
        <cfif (StartRow + MaxRows) LTE GetParks.RecordCount>
```

```
            <cfoutput><a href="#CGI.SCRIPT_NAME#?startrow=#Evaluate(StartRow + MaxRows)#">
            See next #MaxRows# rows</a></cfoutput>
        </cfif>
        </td>
    </tr>
</table>
```

cfqueryparam

Description

Verifies the data type of a query parameter and, for DBMSs that support bind variables, enables ColdFusion to use bind variables in the SQL statement. Bind variable usage enhances performance when executing a `cfquery` statement multiple times.

This tag is nested within a `cfquery` tag, embedded in a query SQL statement. If you specify optional parameters, this tag performs data validation.

Adobe recommends that you use the `cfqueryparam` tag within every `cfquery` tag, to help secure your databases from unauthorized users. For more information, see Security Bulletin ASB99-04, "*Multiple SQL Statements in Dynamic Queries*," at www.adobe.com/devnet/security/security_zone/asb99-04.html, and "Accessing and Retrieving Data" on page 393 in the *ColdFusion Developer's Guide*.

Category

Database manipulation tags

Syntax

```
<cfquery
    name = "query name"
    dataSource = "data source name"
    ...other attributes...
    SQL STATEMENT column_name =
    <cfqueryparam value = "parameter value"
        CFSQLType = "parameter type"
        list = "yes|no"
        maxLength = "maximum parameter length"
        null = "yes|no"
        scale = "number of decimal places"
        separator = "separator character">
    AND/OR ...additional criteria of the WHERE clause...>
</cfquery>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfinsert`, `cfprocparam`, `cfprocresult`, `cfquery`, `cfstoredproc`, `cftransaction`, `cfupdate`; "Enhancing security with cfqueryparam" on page 399 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
value	Required		Value that ColdFusion passes to the right of the comparison operator in a where clause. If CFSQLType is a date or time option, ensure that the date value uses your DBMS-specific date format. Use the CreateODBCDateTime or DateFormat and TimeFormat functions to format the date value.
CFSQLType	Optional	CF_SQL_CHAR	SQL type that parameter (any type) is bound to: • CF_SQL_BIGINT • CF_SQL_BIT • CF_SQL_CHAR • CF_SQL_BLOB • CF_SQL_CLOB • 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_REFCURSOR • CF_SQL_SMALLINT • CF_SQL_TIME • CF_SQL_TIMESTAMP • CF_SQL_TINYINT • CF_SQL_VARCHAR
list	Optional	no	• yes: the value attribute value is a delimited list. • no
maxLength	Optional	Length of string in value attribute	Maximum length of parameter. Ensures that the length check is done by ColdFusion before the string is sent to the DBMS, thereby helping to prevent the submission of malicious strings.
null	Optional	no	Whether parameter is passed as a null value: • yes: tag ignores the value attribute. • no

Attribute	Req/Opt	Default	Description
scale	Optional	0	Number of decimal places in parameter. Applies to CF_SQL_NUMERIC and CF_SQL_DECIMAL.
separator	Required, if you specify a list in value attribute	, (comma)	Character that separates values in list, in value attribute.

Usage

Use the cfqueryparam tag in any SQL statement (for example, SELECT, INSERT, UPDATE, and DELETE) that uses ColdFusion variables.

You cannot use the cfquery cachedAfter or cachedWithin attributes with cfqueryparam.

For maximum validation of string data, specify the maxlength attribute.

This tag does the following:

- Allows the use of SQL bind parameters, which improves performance.
- Ensures that variable data matches the specified SQL type.
- Allows long text fields to be updated from a SQL statement.
- Escapes string variables in single-quotation marks.

To benefit from the enhanced performance of bind variables, you must use cfqueryparam for all ColdFusion variables, and your DBMS must support bind variables. If a DBMS does not support bind parameters, ColdFusion validates and substitutes the validated parameter value back into the string. If validation fails, it returns an error message.

The validation rules are as follows:

- For these types, a data value can be converted to a numeric value: CF_SQL_SMALLINT, CF_SQL_INTEGER, CF_SQL_REAL, CF_SQL_FLOAT, CF_SQL_DOUBLE, CF_SQL_TINYINT, CF_SQL_MONEY, CF_SQL_MONEY4, CF_SQL_DECIMAL, CF_SQL_NUMERIC, and CF_SQL_BIGINT
- For these types, a data value can be converted to a date supported by the target data source: CF_SQL_DATE, CF_SQL_TIME, CF_SQL_TIMESTAMP
- For all other types, if the maxLength attribute is used, a data value cannot exceed the maximum length specified.

ColdFusion debug output shows the bind variables as question marks and lists the values beneath the query, in order of usage.

Note: *To insert an empty string into a Microsoft Access table using the SequelLink ODBC Socket or SequelLink Access driver, the CFSQLType attribute must specify CF_SQL_LONGVARCHAR.*

The following table shows the mapping of ColdFusion SQL data types with JDBC SQL types and those of the listed database management systems:

ColdFusion	JDBC	DB2	Informix	Oracle	MSSQL
CF_SQL_ARRAY	ARRAY				
CF_SQL_BIGINT	BIGINT	Bigint	int8, serial8		
CF_SQL_BINARY	BINARY	Char for Bit Data			binary
					timestamp

ColdFusion	JDBC	DB2	Informix	Oracle	MSSQL
CF_SQL_BIT	BIT		boolean		bit
CF_SQL_BLOB	BLOB	Blob	blob	blob, bfile	
CF_SQL_CHAR	CHAR	Char	char, nchar	char, nchar	char, nchar, unique identifier
CF_SQL_CLOB	CLOB	Clob	clob	clob,nclob	
CF_SQL_DATE	DATE	Date	date, datetime, year to day		
CF_SQL_DECIMAL	DECIMAL	Decimal	decimal, money	number	decimal, money, small-money
CF_SQL_DISTINCT	DISTINCT				
CF_SQL_DOUBLE	DOUBLE	Double			
CF_SQL_FLOAT	FLOAT	Float	float	number	float
CF_SQL_IDSTAMP	CHAR	Char	char, nchar	char, nchar	char, nchar, uniqueidentifier
CF_SQL_INTEGER	INTEGER	Integer	integer, serial		int
CF_SQL_LONGVARBINARY	LONGVARBINARY	Long Varchar for Bit Data	byte	long raw	image
CF_SQL_LONGVARCHAR	LONGVARCHAR	Long Varchar	text	long	text, ntext
CF_SQL_MONEY	DOUBLE	Double			
CF_SQL_MONEY4	DOUBLE	Double			
CF_SQL_NULL	NULL				
CF_SQL_NUMERIC	NUMERIC	Numeric			numeric
CF_SQL_OTHER	OTHER				
CF_SQL_REAL	REAL	Real	smallfloat		real
CF_SQL_REFCURSOR	REF				
CF_SQL_SMALLINT	SMALLINT	Smallint	smallint		smallint
CF_SQL_STRUCT	STRUCT				
CF_SQL_TIME	TIME	Time	datetime hour to second		
CF_SQL_TIMESTAMP	TIMESTAMP	Timestamp	datetime year to fraction(5), datetime year to second	date	datetime, smalldatetime
CF_SQL_TINYINT	TINYINT				tinyint
CF_SQL_VARBINARY	VARBINARY	Rowid		raw	varbinary
CF_SQL_VARCHAR	VARCHAR	Varchar	varchar, nvarchar, lvarchar	varchar2, nvarchar2	varchar, nvarchar, sysname

Example

```
<!--- This example shows cfqueryparam with VALID input in Course_ID. --->
<h3>cfqueryparam Example</h3>
<cfset Course_ID = 12>
<cfquery name = "getFirst" dataSource = "cfdocexamples">
    SELECT *
    FROM courses
    WHERE Course_ID = <cfqueryPARAM value = "#Course_ID#"
    CFSQLType = 'CF_SQL_INTEGER'>
```

```
</cfquery>
<cfoutput query = "getFirst">
    <p>Course Number: #Course_ID#<br> Description: #descript#</p>
</cfoutput>

<!--- This example shows the use of CFQUERYPARAM when INVALID string data is
    in Course_ID. ---->
<p>This example throws an error because the value passed in the CFQUERYPARAM tag exceeds the
    MAXLENGTH attribute</p>

<cfset LastName="Peterson; DELETE employees WHERE LastName='Peterson'">
<!------- Note that for string input you must specify the MAXLENGTH attribute
    for validation. ------------------------------------------------->
<cfquery
    name="getFirst" datasource="cfdocexamples">
    SELECT *
    FROM employees
    WHERE LastName=<cfqueryparam
                value="#LastName#"
                cfsqltype="CF_SQL_VARCHAR"
                maxlength="17">
</cfquery>
<cfoutput
    query="getFirst"> <p>
        Course Number: #FirstName# #LastName#
        Description: #Department# </p>
</cfoutput>
```

cfregistry

Description

This tag is deprecated for the UNIX platform.

Reads, writes, and deletes keys and values in the system registry. Provides persistent storage of client variables.

Note: *For this tag to execute, it must be enabled in the ColdFusion Administrator. For more information, see Configuring and Administering ColdFusion.*

Category

Other tags, Variable manipulation tags

Syntax

The tag syntax depends on the `action` attribute value. See the following sections:

- "cfregistry action = "getAll"" on page 509
- "cfregistry action = "get"" on page 511
- "cfregistry action = "set"" on page 512
- "cfregistry action = "delete"" on page 513

See also

`cfcookie`, `cfparam`, `cfsavecontent`, `cfschedule`, `cfset`; "About resource and sandbox security" on page 313 in "Securing Applications" on page 312 and "Using Persistent Data and Locking" on page 273 in the *ColdFusion Developer's Guide*

History

ColdFusion MX:

- Deprecated this tag on the UNIX platform. It might not work, and might cause an error, in later releases.
- Changed how persistent data is stored: ColdFusion now stores most persistent data outside the system registry, in XML files.

cfregistry action = "getAll"

Description

Returns all registry keys and values defined in a branch. You can access the values as you would any record set.

Syntax

```
<cfregistry
    action = "getAll"
    branch = "branch"
    name = "query name"
    sort = "asc|desc"
    type = "string|dWord|key|any">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

"Using Persistent Data and Locking" on page 273 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Always `getall`.
`branch`	Required		Name of a registry branch.
`name`	Required		Name of record set to contain returned keys and values.
`sort`	Optional	`asc`	Sorts query column data (case-insensitive). Sorts on Entry, Type, and Value columns as text. Specify a combination of columns from query output, in a comma-delimited list. For example: `sort = "value desc, entry asc"` • `asc`: ascending (a to z) sort order. • `desc`: descending (z to a) sort order.
`type`	Optional	`string`	• `string`: returns string values. • `dWord`: returns DWord values. • `key`: returns keys. • `any`: returns keys and values.

Usage

This tag returns `#entry#`, `#type#`, and `#value#` in a record set that you can access through tags such as `cfoutput`. To fully qualify these variables, use the record set name, as specified in the `name` attribute.

If `#type#` is a key, `#value#` is an empty string.

If you specify `type= "any"`, `getAll` also returns binary registry values. For binary values, the `#type#` variable contains UNSUPPORTED and `#value#` is blank.

Example

```
<!--- This example uses cfregistry with the getAll action. --->
<cfregistry action = "getAll"
    branch = "HKEY_LOCAL_MACHINE\Software\Microsoft\Java VM"
    type = "Any" name = "RegQuery">
<h1>cfregistry action = "getAll"</h1>
```

```
<cftable query = "RegQuery" colHeaders HTMLTable border = "yes">
    <cfcol header = "<b>Entry</b>" width = "35" text = "#RegQuery.Entry#">
    <cfcol header = "<b>Type</b>" width = "10" text = "#RegQuery.type#">
    <cfcol header = "<b>Value</b>" width = "35" text = "#RegQuery.Value#">
</cftable>
```

cfregistry action = "get"

Description

Accesses a registry value and stores it in a ColdFusion variable.

Syntax

```
<cfregistry
    action = "get"
    branch = "branch"
    entry = "key or value"
    variable = "variable"
    type = "string|dWord|key">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

"Using Persistent Data and Locking" on page 273 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Always `get`.
`branch`	Required		Name of a registry branch.
`entry`	Required		Registry value to access.
`variable`	Required		Variable into which to put value.
`type`	Optional	*`string`*	• `string`: returns string value. • `dWord`: returns DWord value. • `key`: returns key's default value.

Usage

If the value does not exist, the `cfregistry` tag does not create an entry.

Example

```
<!--- This example uses cfregistry with the get action. --->
<cfregistry action = "get"
    branch = "HKEY_LOCAL_MACHINE\Software\Microsoft\Java VM"
    entry = "ClassPath" type = "String" variable = "RegValue">
<h1>cfregistry action = "get"</h1>
<cfoutput>
    Java ClassPath value is #RegValue#
</cfoutput>
```

cfregistry action = "set"

Description

Adds a registry key, adds a value, or updates a value.

Syntax

```
<cfregistry
    action = "set"
    branch = "branch"
    entry = "key or value"
    type = "string|dWord|key"
    value = "data">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

"Using Persistent Data and Locking" on page 273 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Always `set`.
`branch`	Required		Name of a registry branch.
`entry`	Required		Key or value to set.
`type`	Optional		• `string`: sets a string value (default). • `dWord`: sets a DWord value. • `key`: creates a key.
`value`	Optional		Value data to set. If you omit this attribute, the `cfregistry` tag creates default value, as follows: • `string`: creates an empty string: `" "`. • `dWord`: creates a value of `0` (zero).

Usage

If it does not exist, the `cfregistry` tag creates the key or value.

Example

```
<!--- This example uses the cfregistry set action to modify registry value data. --->
<!--- Normally you pass in a filename instead of setting one here. --->
<cfset FileName = "dummy.cfm">
<cfregistry action = "set"
    branch = "HKEY_LOCAL_MACHINE\Software\cflangref"
    entry = "LastCFM01" type = "String" value = "#FileName#">
<h1>cfregistry action = "set"</h1>
```

cfregistry action = "delete"

Description

Deletes a registry key or value.

Syntax

```
<cfregistry
    action = "delete"
    branch = "branch"
    entry = "key or value">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

"Using Persistent Data and Locking" on page 273 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		Always `delete`.
`branch`	Required		• For key deletion: name of registry key to delete. Do not specify the `entry` attribute. • For value deletion: name of registry branch that contains value to delete. You must specify the `entry` attribute.
`entry`	Required for value deletion		Value to delete.

Usage

If you delete a key, the `cfregistry` tag also deletes values and subkeys defined beneath it.

Example

```
<cfregistry action = "delete"
    branch = "HKEY_LOCAL_MACHINE\Software\cflangref\tempkey"
    entry = "LastCFM01">
<h1>cfregistry action = "delete"</h1>
```

cfreport

Description

Used to do either of the following:

- Execute a report created with the ColdFusion Report Builder, displaying it in PDF, Adobe® FlashPaper®, RTF, HTML, XML or Excel format. Optionally, you can save this report to a file.
- Run a predefined Crystal Reports report. Applies only to Windows systems.

Category

Data output tags

Syntax

```
ColdFusion Report Builder syntax:
<cfreport
    format = "PDF|FlashPaper|Excel|RTF|HTML|XML"
    template = "absolute pathname or pathname relative to the report file"
    encryption = "128-bit|40-bit|none"
    filename = "output filename"
    name = "ColdFusion variable"
    ownerpassword = "password"
    overwrite = "yes}no"
    permissions = "permission list"
    query = "query variable"
    resourceTimespan = #CreateTimeSpan (days, hours, minutes, seconds)#
    style = "CSS style definition or css file pathname"
    userpassword = "password">
    <cfreportparam ...>
</cfreport>

Crystal Reports syntax:
<cfreport
    report = "report path"
    dataSource = "data source name"
    formula = "formula"
    orderBy = "result order"
    password = "password"
    timeout = "number of seconds"
    type = "standard|netscape|microsoft"
    username = "username">
</cfreport>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcollection`, `cfdocument`, `cfdocumentitem`, `cfdocumentsection`, `cfexecute`, `cfindex`, `cfobject`, `cfreportparam`, `cfsearch`, `cfwddx`; "Creating Reports with Report Builder" on page 820 in the *ColdFusion Developer's Guide*; Report Builder online Help

History

ColdFusion 8: Added the `style` and `resourceTimespan` attributes. Added the `HTML` and `XML` values to the `format` attribute.

ColdFusion MX 7.0.1: Added the `RTF` value to the `format` attribute, to let you generate reports in RTF format.

ColdFusion MX 7: Added support for the ColdFusion Report Builder.

ColdFusion MX: Changed data source connection behavior: Crystal Reports now establishes an independent connection to the data source. The connection is not subject to any ColdFusion data source-specific restrictions. For example, the Crystal Reports server can access a data source, regardless of whether it is disabled in the ColdFusion Administrator.

Attributes

Attribute	Applies to	Req/Opt	Default	Description
`datasource`	Crystal Reports	Optional		Name of registered or native data source.
`encryption`	Report Builder	Optional	`none`	(`format="PDF"` only) Type of encryption for the report output. Valid values are: • `128-bit` • `40-bit` • `none`
`filename`	Report Builder	Optional		Filename to contain the report. You cannot specify both the `name` and `filename` attributes. The filename extension must match the value of the `format` attribute. If you write report output to an HTML file, ColdFusion automatically creates a directory relative to the output file in the format *`filename`*_files. Also, it generates PNG files for any charts in the report and copies of any image files imported into the report and stores them in this directory.
`format`	Report Builder	Required		Format of the report output: • `PDF` • `FlashPaper` • `Excel` • `RTF` • `XML` • `HTML` When you write report output directly to the browser in HTML format, ColdFusion generates a temporary directory and files for the images in the report. The location of the temporary directory that contains the image files is: C:\ColdFusion8\tmpCache\CFFileServlet_cfreport_report[*unique_identifier*] To determine when the images are removed from the browser, use the `resourceTimespan` attribute.

`formula`	Crystal Reports	Optional		One or more named formulas. Terminate each formula with a semicolon. Use the format:
				`formula = "formulaname1 = 'formula1';formulaname2 = 'formula2';"`
				If you use a semicolon in a formula, you must escape it by typing it twice (;;). For example:
				`formula = "Name1 = 'Val_1a;;Val_1b';Name2 = 'Val2';"`

`name`	Report Builder	Optional		Name of the ColdFusion variable that contains the report output. You cannot specify both `name` and `filename`. This attribute is not valid when `format="HTML"`.

Attribute	Applies to	Req/Opt	Default	Description
orderBy	Crystal Reports	Optional		Orders results according to your specifications.
overwrite	Report Builder	Optional	no	Specifies whether to overwrite files that have the same name as that specified in the filename attribute: • yes • no
ownerPassword	Report Builder	Optional		(format="PDF" only) Owner password for the report,
password	Crystal Reports	Optional		Password that corresponds to username required for database access. Overrides default settings for data source in the ColdFusion Administrator.
permissions	Report Builder	Optional		(format="PDF" only) Specifies one or more of the following permissions: • AllowPrinting • AllowModifyContents • AllowCopy • AllowModifyAnnotations • AllowFillIn • AllowScreenReaders • AllowAssembly • AllowDegradedPrinting Separate multiple permissions with commas.
query	Report Builder	Optional		Name of the query that contains input data for the report. This query overrides the query in the Report Builder report. The ColdFusion query must contain at least all of the columns included in the Report Builder query; however, the WHERE clause can differ. If you omit this parameter, Report Builder uses the data from the internal SQL statement or from cfreportparam items.
report	Crystal Reports	Required		Report pathname. Store Crystal Reports files in the same directories as ColdFusion page files.
resourceTimespan	Report Builder	Optional	5 minutes	(format="HTML" only) Life span of the resource directory. When you export a Report Builder report in HTML format, ColdFusion writes any images or other resource files in the report to a temporary resource directory. Use this attribute to determine when the resource directory is deleted. For the value, use the CreateTimeSpan function and specify the timespan in days, hours, minutes, and seconds, separated by commas; for example, to delete the resource directory after one hour, specify: #CreateTimeSpan(0,1,0,0)# If the value is set to 0, the resource directory persists until the next server restart. ColdFusion deletes the resource directory only when format="HTML" and neither the name nor filename attribute is specified.The default setting is 5 minutes: #CreateTimeSpan(0,0,5,0)#
style	Report Builder	Optional		Style in CSS format that overrides a style defined in the Report Builder report at run time. You can specify an absolute file path, a file path relative to the report, or a string in valid CSS format. For the styles to take effect, the style names must match Style Name attributes assigned to elements in the Report Builder report. You can generate a CSS file in Report Builder and export it or you can create a CSS file with a text editor. For a list of supported CSS styles, see "Style properties" on page 518.
template	Report Builder	Required		Specifies the pathname to the Report Builder (CFR) file, relative to the web root.

Attribute	Applies to	Req/Opt	Default	Description
timeout	Crystal Reports	Optional		Specifies the maximum time, in seconds, in which a connection must be made to a Crystal Reports file.
type	Crystal Reports	Optional	standard	Type of report: • standard (not valid for Crystal Reports 8.0) • netscape • microsoft
userName	Crystal Reports	Optional		Username required for entry into a database from which the report is created. Overrides default settings for data source in ColdFusion Administrator.
userPassword	Report Builder	Optional		(format="PDF" only) User password.

Usage

Use this tag to generate a report using a report definition created in either ColdFusion Report Builder or in Crystal Reports. (For more information on using the ColdFusion Report Builder, display the online help by opening the Report Builder and pressing F1.)

Note: *The Excel report output format type provides limited support for the formatting options available in ColdFusion Reporting. Images and charts are not supported and numeric data containing formatting (commas, percents, currency, and so on) appear as plain text in Excel. The Excel output format supports simple reports only and Adobe recommends that you give careful design and layout consideration to reports designed for Excel output.*

This tag requires an end tag.

Using Cascading Style Sheets

You can override Cascading Style Sheets (CSS) in Report Builder reports at run time by using the style attribute of the cfreport tag in ColdFusion.

You can create CSS files in one of two ways: by exporting styles with the Export Report Styles icon in Report Builder or by creating a CSS file in any text editor. For the CSS styles to take effect, however, you must use Report Builder to assign the style names to the elements in the report. (The exception is the default style: you can use the style attribute to define the default style in ColdFusion and apply it to the report even if the default style is not defined in Report Builder.)

After you assign the style names in Report Builder, you can update the style definitions in the CSS file at any time and apply them at run time by using the cfreport and cfreportparam tags. If your report contains subreports, the default style applies to the master report and to all of the subreports. If the master report uses CSS styles other than the default style, the CSS styles do not apply to the subreports unless you specify them explicitly.

The following code shows how to apply three different style sheets to the main report and two subreports at run time:

```
<cfreport template="myreport.cfr" style="mystyle.css" format="PDF">
    <cfreportParam subreport="subreport1" style="subreport1-style.css">
    <cfreportParam subreport="subreport2" style="subreport2-style.css">
</cfreport>
```

The following code shows how to apply a CSS style as a value of the style attribute:

```
<cfreport template="myreport.cfr" style='mystyle { defaultStyle: true; font-family:"Comic
Sans MS"; color: ##00FF00; }' format="FlashPaper">
</cfreport>
```

The following code shows how to create a variable called *myStyle* and use it as a value of the style attribute:

```
<cfset mystyle='mystyle { defaultStyle: false; font-family: "Comic Sans MS"; }'>
```

```
<cfreport template="myreport.cfr" style="#mystyle#" format="HTML">
</cfreport>
```

Style attribute syntax

The style file or string must be valid CSS syntax. For more information, see http://www.w3.org/Style/CSS/. The style must contain one or more rule sets. Each rule set consists of a simple selector, which is the Report Builder style name, followed by a declaration block, which consists of a series of declarations separated by semicolons. A declaration is a property:value pair.

If a selector contains invalid syntax, ColdFusion ignores the selector and its declaration block. Selectors and properties not supported by this feature are ignored. Styles are case-insensitive, except parts not under the control of CSS (such as font names).

The following example shows the CSS definition for the default style:

```
DefaultStyle
    {
        default-style: true;
        color: black;
        font-family: Arial, "Comic Sans MS";
        font-size: 16;
        text-decoration: underline;
    }
```

The following example shows the CSS definition for a custom style called PurpleBoldItalicText:

```
PurpleBoldItalicText
    {
        color: purple;
        font: italic bold 20px 30px Arial;
    }
```

Identifiers for styles must be CSS2-compliant. For example, CSS1 allows '_' in identifiers, but CSS2 does not.

In CSS2, identifiers, including element names, classes, and IDs in selectors, can contain only the characters A-Z, a-z, and 0-9. Also, they can include ISO 10646 characters 161 and higher and the hyphen character (-); however, identifiers cannot start with a hyphen or a digit. They can contain escaped characters and any ISO 10646 character as a numeric code. For example, you can write the identifier "B&W?" as "B\&W\?" or "B\26 W\3F".

Style properties

The following table shows the style properties exported by Report Builder:

Property name	Report Builder only	Valid values	Description
background-color	No	See "Valid color values" on page 523	Background color of the specified report element, if the element is not transparent. The default background color is white.

border	No	`[border-width]`
		`[border-style]`
		`[border-color]`
			A shorthand property that specifies the `border-width`, `border-style`, and `border-color` properties for all of the borders in an element.

border-color	No	See "Valid color values" on page 523	Border color for text, images, and charts. You can customize each side of the border. The default color is white.

Property name	Report Builder only	Valid values	Description

border-style	No	`solid`
		`dashed`
		`none`
			A shorthand property that specifies the `border-top-style`, `border-right-style`, `border-bottom-style`, and `border-left-style` (the comma-separated values must be in this order). If one or more values are not specified, the value for the opposite side is used. If only one value is listed, that value applies to all sides.
			The `none` value overrides the `border-width` value. Any other values, including `hidden`, `dotted`, `groove`, `ridge`, `inset`, `outset`, and `double`, are displayed as solid.

border-top-color border-left-color border-bottom-color border-right-color	No	See "Valid color values" on page 523.	Color of the element's top, left, bottom, and right border. See "Border and margin styles" on page 523. If no `border-color` property is specified, the value of the `color` property is used instead.
border-top-style border-left-style border-bottom-style border-right-style	No	`solid` `dashed`	Line style of the element's top, left, bottom, and right border. See "Border and margin styles" on page 523. Any value other than `dashed` displays as solid.

border-top-width
border-left-width
border-bottom-width
border-right-width
	No	`thin`
		`medium`
		`thick`
		`1px`
		`2px`
		`4px`
			Thickness of the top, left, bottom, and right border of an element. Negative values are not valid. See "Border and margin styles" on page 523:
			• `thin = 1/2 pt`
			• `medium = 2px`
			• `thick = 4px`

border-width	No	`thin`
		`medium`
		`thick`
		`1px`
		`2px`
		`4px`
			A shorthand property that specifies the `border-top-width, border-right-width`, and `border-bottom-width` properties with a single property and value notation (the comma-separated values must be in this order). If one or more values are not specified, the value for the opposite side is used. If only one value is listed, it applies to all sides:
			• `thin = 1/2 pt`
			• `medium = 2px`
			• `thick = 4px`

Property name	Report Builder only	Valid values	Description

clip	No	`auto`
		`stretch`
		`ratio`
			Specifies how images are resized:
			• `auto`: If the dimensions of the source image differ from the element in the report, this attribute crops the image to fit within the element boundaries. The image is not resized. Only the part of the image that fits in the boundaries is displayed.
			• `stretch`: If the dimensions of the source image differ from the element in the report, this attribute resizes the image so that it fits within the element boundaries. If the dimensions differ, the image is distorted.
			• `ratio`: If the dimensions of the source image differ from the element in the report, this attribute resizes the image to fit within the element boundaries but maintains the aspect ratio of the source so that the image is not distorted.

color	No	See "Valid color values" on page 523	Color of the text (in text elements) and the border (in graphic elements). The default color is black.
corner-radius	Yes	`integer`	Radius for arcs used to draw the corner of rectangles. The default is 0 (square corners). Values less than 0 are interpreted as 0.
default-style	Yes	`true` `false`	Default style for elements that do not or cannot have a style applied. A subreport inherits its parent's default-style if it does not have one of its own.
embed-pdf-font	Yes	`true` `false`	Specifies whether fonts are embedded in the PDF document. Embedded fonts insure that the fonts display properly even if the font is not installed on the system where the report is viewed.
empty-cells	No	`show` `hide`	Shows or hides a null value for text expressions: • `show`: If the text expression returns a null value, the string "null" is displayed. • `hide`: If the text expression returns a null value, the null value is replaced with an empty string. This is the default.

font	No	`[font-style]`
		`[font-weight]`
		`[font-size]`
		`[line-height]`
		`[font-family]`
			Font characteristic specifications. Use this as a shorthand to specifying multiple property values; for example:
			`font: italic 20px Arial;`
			Default values for this property match those used for the individual properties. Default values for the individual properties are applied to the values omitted from the font property.

font-family	No	Comma-separated list of font names.	Group of fonts to apply to the element. The first font found in the list is applied to the element. The default is:
			`font-family: Arial, Helvetica, sansserif;`
			If a font name contains spaces, enclose the name in quotation marks, for example:
			`font-family: Courier, "Courier New", Arial;`

Property name	Report Builder only	Valid values	Description
font-size	No	[*length*]	Font size measured in points or pixels. Negative values are not valid. The default value is 10 points. You can specify points or pixels. 1 pixel = 0.75. points. This property also is a component of the font property. Standard CSS supports other types of values not supported by Report Builder.

font-style	No	`normal`
		`italic`
		`oblique`
			Font style. The `italic` and `oblique` values are similar. The default value is normal. Also, this property is a component of the `font` property.

font-weight	No	`normal`
		`bold`
		`bolder`
		`lighter`
		`100`
		`200`
		`300`
		`400`
		`500`
		`600`
		`700`
		`800`
		`900`
			Font weight. Report Builder does not support varying degrees of boldness or lightness; therefore, `normal` and `lighter` appear as `normal`; all other values appear as `bold`. The default is `normal`. Also, this property is a component of the `font` property.

line-height	No	`normal`
		`[number]`
		`[length]`
		`[percentage]`
			Amount of space between consecutive lines of text:
			
			• `normal`: Sets the line-height to single-spacing (default).
			
			• `number`: A multiplier that determines the line height as a factor of the element's font size. To determine the line height from this number, multiply the current element font-size by the number. Negative values are not valid.
			
			• `length`: Sets the line height to an explicit length. You can specify points (for example, "20") or pixels (for example, "20px"). 1 pixel = 0.75 points. Negative values are not valid. Standard CSS allows other units of length not supported by Report Builder.
			
			• `percentage`: Defines the line-height as a percentage. The percent symbol is required (for example, 150%). Negative values are invalid.

Property name	Report Builder only	Valid values	Description

line-size	Yes	`none`
		`thin`
		`1px`
		`2px`
		`4px`
		`dashed`
			Type of the border around a graphic element or the type and the thickness of line elements. By default, lines and rectangles have a 1-pixel border; `thin` is 0.05 pixels.

margin	No	`[top-integer]`
		`[right-integer]`
		`[bottom-integer]`
		`[left-integer]`
			Amount of blank space within the bounding box of an element. This is a shorthand property that specifies the `margin-top`, `margin-right`, `margin-bottom`, and `margin-left` properties with a single property and value notation (the values must be in this order separated by commas.) If one or more values are not specified, the value for the opposite side is used. If only one value is listed, it applies to all sides. See "Border and margin styles" on page 523.
			CSS margins are transparent, which reveals the background of the parent element. Negative values are valid; this allows for text overlays.
			Examples:
			`margin: 10,20,30,40;`
			`margin: 10;`
			`margin: 10,20,30;`

margin-top margin-left margin-bottom margin-right	No	`integer`	See margin.
parent-style	Yes	`styleName`	Name of the parent report style from which this style inherits some or all of its properties. The style name must be defined in either the report or the before this style definition in the CSS file or text.

text-align	No	`left`
		`center`
		`right`
		`justify`
			Alignment of text and images on the horizontal axis. The default value is `left`.

text-decoration	No	`underline`
		`line-through`
		`underline line-through`
			Text characteristics not defined with the `font-style` and `font-weight` properties. The color of the text-decoration is determined by the color property for the element. Unknown values are ignored.

text-rotation	Yes	`none`
		`left`
		`right`
			Rotation of text elements. Use it to change the text direction by rotating it 90 degrees to the right or left.

transparency-mode	Yes	`opaque` `transparent`	Transparency of elements. Graphic elements, such as rectangles and lines, are opaque by default, but images are transparent. Subreport elements, static text, and text fields are transparent by default.

Property name	Report Builder only	Valid values	Description

vertical-align	No	`top`
		`middle`
		`bottom`
			Alignment of text and images on the vertical axis. The default value is `top`.

xhtml-formatted-text	Yes	`true` `false`	Specifies whether the text element contains XHTML-compatible instructions: • `true`: The text element contains xhtml-compatible instructions, for example, My Text Label. In this example, the text within the tags displays in bold (**My Text Label**), and the tags ( ) are not displayed. • `false`: The text element does not contain xhtml-compatible instructions; therefore, all the text within the text element is displayed. This is the default.

Styles or values that are not supported by Report Builder are ignored in the report, in which case, if a default-style is defined, Report Builder applies the default style to the element.

Valid color values

You can specify a color as #RRGGBB. This represents a color that uses a triplet of hexadecimal values concatenated together. The values represent the red, green, and blue components for a given color. The range of each component value is 00-FF in hexadecimal. Also, you can use one of the 140 X11 color names (see http://www.blooberry.com/indexdot/color/x11makerFrameNS.htm). The color name is case-insensitive. This set of names assigns names to specific RGB values. Also, a color name can also be specified as ##RGB, rgb(r,g,b), or rgb(r%,g%,b%). See CSS Color Units for syntax details (see http://www.blooberry.com/indexdot/css/syntax/units/color.htm). UI Name is not supported.

The following example shows the different ways you can represent the color lime:

```
color:lime
color:#00FF00
color:#0F0
color:rgb(0,255,0)
color:rgb(0%,50%,0%)
```

If you specify a color in hexadecimal format as part of the `style` attribute for the `cfreport` tag, you must use the format `##00FF00`. For example:

```
<cfreport template="myreport.cfr" style='mystyle { defaultStyle: true;
    font-family:"Comic Sans MS"; color: ##00FF00; }' format="HTML"/>
```

Border and margin styles

Use the `border-width`, `border-style`, `border-color`, and `margin` properties when all four sides of the element have the same value. You can specify from one to four parameters for these properties:

Number of parameters	Example	Result
1	border-width: thick	Parameter applied to all four sides of the element's border.
2	border-width: thick, thin	First parameter (`thick`) applied to the top and bottom sides; second parameter (`thin`) applied to the left and right sides.

Number of parameters	Example	Result
3	border-width: thick, thin, medium	First parameter (`thick`) applied to the top; second parameter (`thin`) applied to the left and right sides; third parameter (`medium`) applied to the bottom.
4	border-width: thick, thin, medium, thick	First parameter (`thick`) applied to the top; second parameter (`thin`) applied to the right side; third parameter (`medium`) applied to the bottom; fourth parameter (`thick`) applied to the left side.

You can use the properties for each side of a border to override the style specified by the `border-width`, `border-style`, `border-color`, and `margin` properties.

Example

Example 1: This example shows the use of cfreport for the ColdFusion Report Builder.

```
<cfquery name="northwindemployees" datasource="localnorthwind">
    SELECT EmployeeID, LastName, FirstName, Title, City, Region, Country
    FROM Employees
    ORDER BY Country, City
</cfquery>

<CFREPORT format="PDF" template="FifthReport.cfr"
 query="#northwindemployees#"/>
```

Example 2: This view-only example shows the use of cfreport for Crystal Reports.

```
<h3>cfreport Tag<h3>
<p>cfreport lets reports from the Crystal Reports Professional report writer display through
a ColdFusion interface. To run, the tag requires the name of the report. cfreport can also
pass information to the report file displayed, to change the output conditions.</p>
<p>This example would run a report called "monthlysales.rpt " and pass it an
    optional filter condition to show only the information for a subset
    of the report.</p>

<cfreport report = '/reports/monthlysales.rpt'>
 {Departments.Department} = 'International'
</cfreport>

<p>Substitute your report files and filters for this code. cfreport can put Crystal Reports
into web pages.</p>
```

cfreportparam

Description

The cfreportparam tag lets you perform the following tasks:

- Pass input parameters to a ColdFusion Report Builder report definition.
- Override query data in subreports and charts defined in Report Builder reports.
- Override styles defined in Report Builder subreports.

The cfreportparam tag is always a child tag of the cfreport tag.

Category

Data output tags

Syntax

```
<cfreport template = ...>
    <cfreportparam
        chart = "name of the chart contained in the report or subreport"
        name = "data name"
        query = "query value passed to the chart or subreport"
        series = "ordinal number of a chart series"
        style = "CSS style definition or CSS file pathname"
        subreport = "name of the subreport"
        value = "data value">
</cfreport>
```

Note: *You can specify this tag's attributes in an* attributeCollection *attribute whose value is a structure. Specify the structure name in the* attributeCollection *attribute and use the tag's attribute names as structure keys.*

See also

cfreport; "Creating Reports with Report Builder" on page 820 in the *ColdFusion Developer's Guide*; Report Builder online Help

History

ColdFusion 8: Added the chart, query, series, subreport, and style attributes.

ColdFusion MX 7: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
chart	Optional		Name of the chart contained in a report or subreport. The value of this attribute must match Name property of a chart defined in the Report Builder report. If you specify the chart attribute, you cannot specify the subreport or name attribute.
name	Optional		Variable name for data that is passed. The value of this attribute must match the name of an input parameter defined in the Report Builder report. If you specify the name attribute, you cannot specify the chart or subreport attribute.
query	Optional		Query value to pass to a subreport or chart. The ColdFusion query must contain at least all of the columns included in the Report Builder query. Charts and subreports require this attribute.
series	Optional	1	Ordinal number of a chart series to use for the query. This attribute is valid only when the chart attribute is specified.

Attribute	Req/Opt	Default	Description
subreport	Optional		Name of the subreport. The value of this attribute must match the Name property of the subreport in Report Builder. Subreport names within a report must be unique. If you specify the subreport attribute, you cannot specify the chart or name attribute.
style	Optional		Style in CSS format for a subreport. The value can be an absolute file path, a file path relative to the report, or a string in valid CSS format. For the styles to take effect, the style names must match Style Name attributes assigned to elements in the Report Builder report. You can generate the CSS file in Report Builder and exported or created with a text editor. For a list of supported CSS styles, see "Style properties" on page 518.
value	Optional (see Description)		Value of the data that is sent. You must specify the value attribute with the name attribute. You cannot specify this attribute when a chart or subreport attribute is specified. The value can be a string or a variable.

Usage

You can specify only one of the following attributes in a cfreportparam tag:

- name
- subreport
- chart

You can use the query, subreport, and chart attributes to override Report Builder queries and chart information at run time. This way you can customize subreport and chart data from the CFM page without having to change the queries built into your report.

For example, in Report Builder, you can create a master report that contains several subreports and populate each subreport with a different query. Instead of modifying the queries in Report Builder, you can customize your reports by creating modified queries on the ColdFusion calling page. The ColdFusion query must contain at least all of the columns included in the Report Builder query.

Note: *You cannot specify a subreport query that depends on arguments from the master report. Instead, you can define a CFML function or CFC method that returns the subreport query given the arguments from the master report. ColdFusion calls this code when it executes the subreport.*

On the calling CFM page, you can specify a cfreportparam tag for any subreport and chart in the Report Builder report. The value of the subreport or chart attribute must match the Name property of the subreport or chart defined in the Report Builder report. (Charts are treated like subreports.)

The following code shows a master report that contains two subreports and a chart with two chart series:

```
<cfreport template="myreport.cfr" query="master" format="RTF">
    <cfreportParam subreport="subreport1" query="subquery1">
    <cfreportParam subreport="subreport2" query="subquery2">
    <cfreportParam chart="chart1" series="1" query="chartquery1">
    <cfreportParam chart="chart1" series="2" query="chartquery2">
    <cfreportParam name="ReportDate" value="#DateFormat(Now())#, #TimeFormat(Now())#">
</cfreport>
```

The cfreportparam tag also lets you override CSS styles assigned to subreports in Report Builder. Use the style attribute with the subreport attribute; the value of the subreport attribute must match the name of the subreport in Report Builder. The following code applies a style sheet to the master report and two different style sheets to the subreports:

```
<cfreport template="myreport.cfr" style="myStyle.css" format="PDF">
    <cfreportParam subreport="subreport1" style="subreport-style.css">
    <cfreportParam subreport="subreport2" style="subreport-style.css">
</cfreport>
```

For more information, see "Using Cascading Style Sheets" on page 517.

Example

```
<!--- The following example shows how to override a query in a Report Builder report from
the CFM page. The cfreportparam tag adds the current date and time to the report.--->
<cfquery name="coursedept" datasource="cfdocexamples">
    SELECT Departments.Dept_ID as dDept_ID, Departments.Dept_Name,
     CourseList.Course_ID, CourseList.Dept_ID as cDept_ID,
     CourseList.CorNumber, CourseList.CorName,
     CourseList.CorLevel
    FROM Departments, CourseList
    WHERE Departments.Dept_ID = CourseList.Dept_ID
    ORDER BY CourseList.Dept_ID
</cfquery>

<cfreport format="PDF" template="FourthReport.cfr" query="#coursedept#" overwrite="yes">
<cfreportparam name="ReportTime" value="#DateFormat(Now())#, #TimeFormat(Now())#">
</cfreport>
```

cfrethrow

Description

Rethrows the currently active exception. Preserves the exception's `cfcatch.type` and `cfcatch.tagContext` variable values.

Category

Exception handling tags, Extensibility tags

Syntax

```
<cfrethrow>
```

See also

`cferror`, `cfthrow`, `cftry`; "Handling runtime exceptions with ColdFusion tags" on page 259 in the *ColdFusion Developer's Guide*

Usage

Use this tag within a `cfcatch` block. This tag is useful in error handling code, if the error handler cannot handle an error that it catches. For example, if `cfcatch type = "any"` gets a DATABASE exception, and the code is designed to handle only CFX exceptions, the handler raises the exceptions again, with details intact, so that a higher-level handler can process the error information. If you used the `cfthrow` tag, the type and details of the original exception would be lost.

Example

```
<h3>cfrethrow Example</h3>
<!--- Rethrow a DATABASE exception. --->
<cftry>
    <cftry>
        <cfquery name = "GetMessages" dataSource = "cfdocexamples">
            SELECT*
            FROM Messages
        </cfquery>
    <cfcatch type = "DATABASE">
        <!--- If database signalled a 50555 error, ignore; otherwise, rethrow
            exception. --->
        <cfif cfcatch.sqlstate neq 50555>
            <cfrethrow>
        </cfif>
    </cfcatch>
    </cftry>
<cfcatch>
    <h3>Sorry, this request can't be completed</h3>
    <h4>Catch variables</h4>
    <cfoutput>
        <cfloop collection = #cfcatch# item = "c">
            <br>
            <cfif IsSimpleValue(cfcatch[c])>#c# = #cfcatch[c]#
            </cfif>
        </cfloop>
    </cfoutput>
</cfcatch>
</cftry>
```

cfreturn

Description
Returns result values from a component method. Contains an expression returned as result of the function.

Return value
An expression; the result of the function from which this tag is called.

Category
Extensibility tags

Syntax
```
<cfreturn
    expr>
```

See also
cfargument, cfcomponent, cffunction, cfinvoke, cfinvokeargument, cfobject, cfproperty; "Building and Using ColdFusion Components" on page 158 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
expr	Required		Function result; value of any type.

Usage
This tag is equivalent to a `return` statement within a `cfscript` tag. It accepts one return variable argument. To return more than one value, populate a structure with name-value-pairs, and return the structure with this tag.

To access the result value from this tag, you use the variable scope that is the value of the `cfinvoke` tag `returnVariable` attribute.

You can code a maximum of one `cfreturn` tag within a function.

For example code, see "Building and Using ColdFusion Components" on page 158 in the *ColdFusion Developer's Guide*.

Example
```
<cfcomponent>
    <cffunction name="getEmp">
         <cfquery name="empQuery" datasource="ExampleApps" >
            SELECT FIRSTNAME, LASTNAME, EMAIL
            FROM tblEmployees
         </cfquery>
         <cfreturn empQuery>
    </cffunction>
    <cffunction name="getDept">
        <cfquery name="deptQuery" datasource="ExampleApps" >
            SELECT *
            FROM tblDepartments
         </cfquery>
         <cfreturn deptQuery>
```

```
    </cffunction>
</cfcomponent>
```

cfsavecontent

Description

Saves the generated content of the `cfsavecontent` tag, including the results of evaluating expressions and executing custom tags, in the specified variable.

Category

Variable manipulation tags

Syntax

```
<cfsavecontent
    variable = "variable name">
    the content
</cfsavecontent>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

"Caching parts of ColdFusion pages" on page 241 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`variable`	Required		Name of the variable in which to save the generated content of the tag.

Usage

This tag requires an end tag.

You cannot use this tag to suppress output from a tag library.

Example

The following example uses a custom tag to generate a report and saves the report in the variable CONTENT. It replaces all instances of the word "report" with the phrase "MyCompany Quarterly Report" and outputs the result.

```
<cfsavecontent variable="content">
    <CF_OutputBigReport>
</cfsavecontent>
<cfoutput>
    #replace(content, "report", "MyCompany Quarterly Report", "all")#
</cfoutput>
```

cfschedule

Description

Provides a programmatic interface to the ColdFusion scheduling engine. Can run a CFML page at scheduled intervals, with the option to write the page output to a static HTML page. This feature enables you to schedule pages that publish data, such as reports, without waiting while a database transaction is performed to populate the page.

Category

Variable manipulation tags

Syntax

```
<cfschedule
    action = "run|update|pause|resume|delete"
    task = "task name"
    endDate = "date"
    endTime = "time"
    file = "filename"
    interval = "seconds"
    operation = "HTTPRequest"
    password = "password"
    path = "path to file"
    port = "port number"
    proxyPassword = "password"
    proxyPort = "port number"
    proxyServer = "host name"
    proxyUser = "user name"
    publish = "yes|no"
    requestTimeOut = "seconds"
    resolveURL = "yes|no"
    startDate = "date"
    startTime = "time"
    url = "URL"
    username = "user name">

OR
<cfschedule
    action = "delete"
    task = "task name">

OR

<cfschedule
    action = "run"
    task = "task name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcookie`, `cfparam`, `cfregistry`, `cfsavecontent`, `cfset`

History

ColdFusion MX 6.1: Changed the way intervals are calculated. The day length now reflects changes between standard and daylight saving times. The month length is now the calendar month length, not four weeks. The scheduler handles leap years correctly.

Attributes

Attribute	Req/Opt	Default	Description
action	Required		• delete: deletes the specified task . • update: updates an existing task or creates a new task, if one with the name specified by the task attribute does not exist. • run: executes the specified task. • pause: pauses the specified task. • resume: continues executing the specified task.
task	Required		Name of the task.
endDate	Optional		Date when scheduled task ends.
endTime	Optional		Time when scheduled task ends (seconds).
file	Required if publish = "Yes"		Name of the file in which to store the published output of the scheduled task.
interval	Required if action = "update"		Interval at which task is scheduled: • number of seconds • once • daily • weekly • monthly
operation	Required if action = "update"		Operation that the scheduler performs. Must be HTTPRequest.
password	Optional		Password, if URL is protected.
path	Required if publish = "Yes"		Path to the directory in which to put the published file.
port	Optional	80	Port to use on the server that is specified by the url parameter. If resolveURL = "yes", retrieved document URLs that specify a port number are automatically resolved, to preserve links in the retrieved document. A port value in the url attribute overrides this value.
proxyPassword	Opt		Password to provide to the proxy server.
proxyPort	Optional	80	Port number to use on the proxy server.
proxyServer	Optional		Host name or IP address of a proxy server.
proxyUser	Opt		User name to provide to the proxy server.
publish	Optional	no	• yes: saves the result to a file. • no
requestTimeOut	Optional		Can be used to extend the default time-out period.
resolveURL	Optional	no	• yes: resolves links in the output page to absolute references. • no
startDate	Required if action = "update"		Date on which to first run the scheduled task.
startTime	Required if action = "update"		Time at which to run the scheduled of task starts.

Attribute	Req/Opt	Default	Description
url	Required if action = "update"		URL of the page to execute.
username	Optional		User name, if URL is protected.

Usage

This tag and the ColdFusion Administrator Scheduled task page schedule ColdFusion tasks. Tasks that you add or change using this tag are visible in the Administrator. You can disable this tag in the Administrator Sandbox/Resource security page. This tag's success or failure status is written to the schedule.log file in the *cf_root*/logs directory (*cf_webapp_root*/WEB-INF/cfusion/logs in the multiserver and J2EE configurations).

When you create a task, you specify the URL of the ColdFusion page to execute, the date, time and frequency of execution, and whether to publish the task output to a HTML file. If the output is published, you specify the output file path and file.

If you schedule a job to run monthly on any date in the range 28-31, the scheduler does the following:

- If you schedule a monthly job to run on the last day of a month, the scheduled job will run on the last day of each month. For example, if you schedule a monthly job to start on January 31, it will run on January 31, February 28 or 29, March 31, April 30, and so on.
- If you schedule a monthly job to run on the 29th or 30th of the month, the job will run on the specified day of each month for 30 or 31-day months, and the last day of February. For example, if you schedule a monthly job to start on January 30, the job will run on January 30, February 28 or 29, March 30, April 30, and so on.

If you schedule a job to run once, the starting time is in the past, and the task has not yet run, it runs immediately. If you schedule a recurring job with a start time in the past, ColdFusion schedules the job to run on the next closest interval in the future.

The Scheduler configuration file, *cf_root*\lib\neo-cron.xml contains all scheduled events, as individual entries.

Example

```
<h3>cfschedule Example</h3>
<!--- This read-only example schedules a task.
        To run the example, remove the comments around the code
        and change the startDate, startTime, url, file, and path attributes
        to appropriate values. --->
<!---
<cfschedule action = "update"
    task = "TaskName"
    operation = "HTTPRequest"
    url = "http://127.0.0.1/playpen/history.cfm"
    startDate = "8/7/03"
    startTime = "12:25 PM"
    interval = "3600"
    resolveURL = "Yes"
    publish = "Yes"
    file = "sample.html"
    path = "c:\inetpub\wwwroot\playpen"
    requestTimeOut = "600">
--->
```

cfscript

Description

Encloses a code block that contains `cfscript` statements.

Category

Application framework tags, Other tags

Syntax

```
<cfscript>
    cfscript code here
</cfscript>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfinvoke`, `cfmodule`, `CreateObject`; "Extending ColdFusion Pages with CFML Scripting" on page 92 in the *ColdFusion Developer's Guide*

History

ColdFusion MX:

- Changed how to invoke component methods: this tag can now invoke component methods, using the `CreateObject` function
- Changed use of reserved words: you cannot use ColdFusion reserved words within this tag
- Added the `try` and `catch` statements.

Usage

Performs processing in CFScript. This tag uses ColdFusion functions, expressions, and operators. You can read and write ColdFusion variables within this tag.

For a detailed description of the CFScript scripting language, including documentation of CFScript statements and the CFScript equivalents of CFML tags, see "Extending ColdFusion Pages with CFML Scripting" on page 92 in the *ColdFusion Developer's Guide* .

You can use this tag to enclose a series of assignment statements that would otherwise require `cfset` statements.

Important: *If you code a `cftry/cfcatch` block within this tag using an exception's Java class name, you must provide the fully qualified class name.*

You cannot use some ColdFusion reserved words in this tag. You cannot put a user-defined function whose name begins with any of these strings within this tag:

- cf
- cf_
- _cf
- coldfusion
- coldfusion_
- _coldfusion

You cannot use the elseif construct within a cfscript tag. You can use code such as the following:

```
else if ( condition )
{
...
}
```

Exception handling with the cfscript tag

To handle exceptions with this tag, use try and catch statements, which are equivalent to the cftry and cfcatch tags. For each try statement, you must have a catch statement. In the catch block, the variable exceptionVariable contains the exception type. This variable is the equivalent of the cfcatch tag built-in variable cfcatch.Type. For more information, see "Extending ColdFusion Pages with CFML Scripting" on page 92 in the *ColdFusion Developer's Guide*.

Invoking ColdFusion components with the cfscript tag

CFScript invokes component methods using the CreateObject function.

The following example shows how to invoke a component object with the cfscript tag, using ordered arguments:

```
<cfscript>
quote = CreateObject( "component", "nasdaq.quote" ) ;
<!--- Invocation using ordered arguments. --->
res = quote.getLastTradePrice( "macr" ) ;
</cfscript>
```

The following example shows how to use an attribute collection within the cfscript tag to pass parameters when invoking a component object. An attribute collection is a structure in which each key corresponds to a parameter name and each value is the parameter value passed for the corresponding key.

```
<cfscript>
    stArgs = structNew();
    stArgs.zipcode = "55987";
</cfscript>
...
<cfinvoke
    webservice = "http://www.xmethods.net/sd/2001/TemperatureService.wsdl"
    method = "getTemp"
    argumentCollection = "#stArgs#"
    returnVariable = "aTemp">
<cfoutput>The temperature at zip code 55987 is #aTemp#</cfoutput>
```

In this example, the structure is created in a cfscript block, but you can use any ColdFusion method to create the structure.

Consuming web services with the cfscript tag

The following example shows how to consume a web service with the cfscript tag. You use the CreateObject function to connect to the web service.

```
<cfscript>
    ws = CreateObject("webservice",
                    "http://www.xmethods.net/sd/2001/TemperatureService.wsdl");
    xlatstring = ws.getTemp("55987");
    writeoutput(xlatstring);
</cfscript>
```

For more information, see "Using Web Services" on page 902 in the *ColdFusion Developer's Guide*.

Example

```
<p>This simple example shows variable declaration and manipulation.
```

```
<cfif IsDefined("form.myValue")>
    <cfif IsNumeric(form.myValue)>
        <cfset x = form.myValue>
        <cfscript>
            y = x;
            z = 2 * y;
            StringVar = form.myString;
        </cfscript>
    <cfoutput><p>twice #x# is #z#.
        <p>Your string value was: <b><I>#StringVar#</i></b></cfoutput>
<cfelse>
```

cfsearch

Description

Searches one or more Verity collections.

A collection must be created and indexed before this tag can return search results.

A collection can be *created* in these ways:

- With the cfcollection tag
- In the ColdFusion MX Administrator
- Using a native Verity indexing tool, such as Vspider or MKVDK. For more information on Vspider and MKVDK, see "Introducing Verity and Verity Tools" on page 109 in *Configuring and Administering ColdFusion*.

If you use a native Verity tool to create a collection, it must be registered. A collection can be *registered with* ColdFusion in the following ways:

- With the cfcollection tag
- In the ColdFusion MX Administrator

A collection can be *indexed* in the following ways:

- In ColdFusion, with the cfindex tag
- In the ColdFusion Administrator, which calls the cfindex tag
- Using a native Verity indexing tool, such as Vspider or MKVDK

For more information, see "Building a Search Interface" on page 460 in the *ColdFusion Developer's Guide*.

Category

Extensibility tags

Syntax

```
<cfsearch
    collection = "collection name"
    name = "search name"
    category = "category[,category2,...]"
    categoryTree = "tree location"
    contextBytes = "number of bytes"
    contextHighlightBegin = "HTML string"
    contextHighlightEnd = "HTML string"
    contextPassages = "number of passages"
    criteria = "search expression"
    language = "language"
    maxRows = "number"
    previousCriteria = "criteria"
    startRow = "row number"
    status = ""
    suggestions = "suggestion option"
    type = "criteria">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfcollection, cfexecute, cfindex, cfobject, cfreport, cfwddx

History

ColdFusion MX 7:

- Added `category`, `categoryTree`, `status`, `suggestions`, `contextPassages`, `contextBytes`, `contextHighlightBegin`, `contextHighlightEnd`, and `previousCriteria` attributes.
- Added `author`, `category`, `categoryTree`, `context`, `rank`, `size`, `recordsSearched`, and `type` result columns.
- Added information on the status structure and its associated keys.
- Removed references to a separate K2 server and k2server.ini file.
- Removed references to unregistered collections.
- Removed references to external collections. ColdFusion MX now manages all collections through the Verity Search service.
- Changed `cflock` recommendation. It is no longer a best practice to surround the `cfsearch` tag with a `cflock` tag.

ColdFusion MX:

- Deprecated the `external` attribute. It might not work, and might cause an error, in later releases. (ColdFusion stores this information about each collection; it automatically detects whether a collection is internal or external.) This tag supports absolute (also known as fully qualified) collection pathnames and mapped collection names.
- Changed query result behavior: the `cfindex` tag can index the query results from a `cfsearch` operation.
- Changed Verity operations behavior: ColdFusion supports Verity operations on Acrobat PDF files.
- Changed multiple collection behavior: this tag can search multiple collections. In a multiple collection search, you cannot combine collections that are registered with K2Server and registered in another way.
- Changed acceptable collection naming: this tag accepts collection names that include spaces.
- Changed the following support: this tag supports Verity 2.6.1 and the LinguistX and ICU locales.
- Changed thrown exceptions: this tag can throw the SEARCHENGINE exception.

Attributes

Attribute	Req/Opt	Default	Description
`name`	Required		Name of the search query.
`collection`	Required		One or more collection names. You can specify more than one collection unless you are performing a category search (that is, specifying `category` or `categoryTree`).
`category`	Optional		A list of categories, separated by commas, to which the search is limited. If specified, and the collection does not have categories enabled, ColdFusion throws an exception.
`categoryTree`	Optional		The location in a hierarchical category tree at which to start the search. ColdFusion searches at and below this level. If specified, and the collection does not have categories enabled, ColdFusion throws an exception. Can be used in addition to the `category` attribute.
`criteria`	Optional		Search criteria. Follows the syntax rules of the `type` attribute. If you pass a mixed-case entry in this attribute, the search is case-sensitive. If you pass all uppercase or all lowercase, the search is case-insensitive. Follow Verity syntax and delimiter character rules; see "Using Verity Search Expressions" on page 489 in the *ColdFusion Developer's Guide*.
`contextBytes`	Optional	`300`	The maximum number of bytes Verity returns in the context summary.

Attribute	Req/Opt	Default	Description
contextHighlightBegin	Optional		The HTML to prepend to search terms in the context summary. Use this attribute in conjunction with contextHighlightEnd to highlight search terms in the context summary.
contextHighlightEnd	Optional		The HTML to append to search terms in the context summary. Use this attribute in conjunction with contextHighlightBegin to highlight search terms in the context summary.
contextPassages	Optional	0	The number of passages/sentences Verity returns in the context summary (that is, the context column of the results). The default i s 0, which disables context summary.
language	Optional	english	Deprecated. This attribute is now ignored and the language of the collection is used to perform the search.
maxRows	Optional	all	Maximum number of rows to return in query results.
previousCriteria	Optional		The name of a result set from an existing set of search results. Verity searches the result set for criteria without regard to the previous search score or rank. Use this attribute to implement searching within result sets.
startRow	Optional	1	First row number to get.
status	Optional		Specifies the name of the structure variable into which ColdFusion places search information, including alternative criteria suggestions (spelling corrections). For a list of keys in this structure, see "Status structure keys" on page 541.
suggestions	Optional	never	Specifies whether Verity returns spelling suggestions for possibly misspelled words. Use one of the following options: • Always: Verity always returns spelling suggestions. • Never: Verity never returns spelling suggestions. • positive integer: Verity returns spelling suggestions if the number of documents retrieved by the search is less than or equal to the number specified. There is a small performance penalty for retrieving suggestion data.
type	Optional	simple	Used to specify the parser that Verity uses to process the criteria. • simple: STEM and MANY operators are implicitly used. • explicit: operators must be invoked explicitly. Also known as Bool_Plus. • internet: for documents that are mostly WYSIWIG (what-you-see-is-what-you-get) documents. Also known as Internet_advanced. • internet_basic: minimizes search time. • natural: specifies the Query By Example (QBE) parser. Also known as FreeText. For more information, see "Using Verity Search Expressions" on page 489 in the *ColdFusion Developer's Guide*. Also see your Verity documentation.

Usage

The cfsearch tag returns a query object whose columns you can reference in a cfoutput tag. For example, the following code specifies a search for the exact terms "filming" or "filmed":

```
<cfsearch
    name = "mySearch"
    collection = "myCollection"
    criteria = '<WILDCARD>`film{ing,ed}`'
    type="explicit"
    startrow=1
    maxrows = "100">
<cfdump var = "#mySearch#>
```

In this example, the single-quotation mark (') and backtick (`) characters are used as delimiters; for more information, see "Using Verity Search Expressions" on page 489 in the *ColdFusion Developer's Guide*.

To optimize search performance, always specify the `maxrows` attribute, setting it to a value that matches your application's needs. A value less than 300 helps to ensure optimal performance.

Adobe does not recommend using the `cflock` tag with this tag; Verity provides the locking function. Using the `cflock` tag slows search performance.

The cfsearch tag result columns

Variable	Description
`context`	A context summary containing the search terms, highlighted in bold (by default). This is enabled if you set the `contextpassages` attribute to a number greater than zero.
`url`	Value of `URLpath` attribute in the `cfindex` tag used to populate a collection.
`key`	Value of the `key` attribute in the `cfindex` tag used to populate a collection.
`title`	Value of `title` attribute in `cfindex` tag used to populate the collection, including PDF and Office document titles. If a title is not extracted from the document, the tag uses the `cfindex title` attribute value for each row.
`score`	Relevancy score of document based on search criteria
`custom1, custom2, custom3, custom4`	Value of custom fields in `cfindex` tag used to populate a collection.
`size`	The number of bytes in the index document.
`rank`	The rank of this document in the search results.
`author`	Extracted from the HTML, Office, and PDF documents when available.
`type`	The MIME type of the document.
`category`	A list of the categories that were specified for this document when it was indexed.
`categoryTree`	A hierarchical category tree, or serial list of categories, that was specified for this document when it was indexed. Only a single tree is returned.
`summary`	Contents of automatic summary generated by `cfindex`.
`recordCount`	Number of records returned in record set
`currentRow`	Current row that `cfoutput` is processing.
`columnList`	List of column names within record set.
`recordsSearched`	Number of records searched. This is the same value for each row in the record set. Corresponds to the `searched` key in the status structure.

Status structure keys

Variable	Description
`found`	The number of documents that contain the search criteria.
`searched`	The number of documents searched. Corresponds to the `recordsSearched` column in the search results.
`time`	The number of milliseconds the search took, as reported by the Verity K2 search service.

Variable	Description
suggestedQuery	An alternative query, as suggested by Verity, that might produce better results. This often contains corrected spellings of search terms. Present only when the suggestions tag attribute criteria is met.
keywords	A structure containing each search term as a key to an array of up to five possible alternative terms, in order of preference. Present only when the suggestions attribute criteria is met.
keywordScore	A structure in the same format as for keywords, except it also includes Verity-reported weighted values from 0 to .99, in which higher scores indicate better-quality results.

To permit application users to search Verity collections for nonstandard strings, words, or characters (for example, "AB23.45.67" or "--->") that would otherwise cause an error, you can create a text file that lists these elements and defines their formats for Verity. Name the file style.lex and put copies of the file in these directories:

- Windows:
  - *cf_root*\verity\k2\common\style
  - *cf_root*\verity\Data\stylesets\ColdFusionK2
- UNIX:
  - *cf_root*//verity/k2/common/style
  - *cf_root*/verity/Data/stylesets/ColdFusionK2

In the multiserver and J2EE configurations, you install Verity in a separate directory.

Note: *To search for a character such as an angle bracket (< or >), you must use a criteria attribute value such as* `"&lt:"` *or* `"&lt:"`*. The bracket characters are reserved in Verity, and using a backslash to escape the character (criteria="\<") does not work in this context. For more information, see "Using Verity Search Expressions" on page 489 in the ColdFusion Developer's Guide.*

Example

```
<!--- #1 (TYPE=SIMPLE) ----------------------------->
<cfsearch
    name="name"
    collection="snippets,syntax,snippets"
    criteria="example"
    maxrows = "100">
<p>
<cfoutput>Search Result total =#name.RecordCount# </cfoutput><br>
<cfoutput>
    url=#name.url#<br>
    key=#name.key#<br>
    title=#name.title#<br>
    score=#name.score#<br>
    custom1=#name.custom1#<br>
    custom2=#name.custom2#<br>
    summary=#name.summary#<br>
    recordcount=#name.recordcount#<br>
    currentrow=#name.currentrow#<br>
    columnlist=#name.columnlist#<br>
    recordssearched=#name.recordssearched#<br>
</cfoutput>
<cfdump var = #name#>
<br>

<!--- #2 (TYPE=EXPLICIT) ----------------------------->
<cfsearch
```

```
    name = "snippets"
    collection = "snippets"
    criteria = '<wildcard>`film{ing,ed}`'
    type="explicit"
    startrow=1
    maxrows = "100">
<cfoutput
    query="snippets">
    url=#url#<br>
    key=#key#<br>
    title=#title#<br>
    score=#score#<br>
    custom1=#custom1#<br>
    custom2=#custom2#<br>
    summary=#summary#<br>
    recordcount=#recordcount#<br>
    currentrow=#currentrow#<br>
    columnlist=#columnlist#<br>
    recordssearched=#recordssearched#<br>
</cfoutput>
<cfdump var = #snippets#>
<br>

<!--- #3 (search by CF key) ----------------------------->
<cfsearch
    name = "book"
    collection = "custom_book"
    criteria = "cf_key=bookid2"
    maxrows = "100">
<cfoutput>
    url=#book.url#<br>
    key=#book.key#<br>
    title=#book.titleE#<br>
    score=#book.score#<br>
    custom1=#book.custom1#<br>
    custom2=#book.custom2#<br>
    summary=#book.summary#<br>
    recordcount=#book.recordcount#<br>
    currentrow=#book.currentrow#<br>
    columnlist=#book.columnlist#<br>
    recordssearched=#book.recordssearched#<br>
</cfoutput>
<cfdump var = #book#>
```

cfselect

Description

Constructs a drop-down list box form control. Used in a `cfform` tag. You can populate the list from a query, or by using the HTML `option` tag.

Category

Forms tags

Syntax

```
<cfselect
    name="name"
    bind="bind expression"
    bindAttribute="attribute name"
    bindOnLoad="yes|no"
    display="text"
    editable="yes|no"
    enabled="yes|no"
    group="query column name"
    height="number of pixels"
    id="HTML id"
    label="label"
    message="text"
    multiple="yes|no"
    onBindError="JavaScript function name"
    onChange="JavaScript or ActionScript"
    onClick="JavaScript function name"
    onError="JavaScript"
    onKeyDown="JavaScript or ActionScript"
    onKeyUp="JavaScript or ActionScript"
    onMouseDown="JavaScript or ActionScript"
    onMouseUp="JavaScript or ActionScript"
    query="query name"
    queryPosition="above|below"
    required="yes|no"
    selected="value or list"
    size="integer"
    sourceForTooltip="URL"
    style="style specification"
    tooltip="text"
    value="text"
    visible="yes|no"
    width="number of pixels">

zero or more HTML option tags


</cfselect>
```

Some attributes apply to only specific display formats. For details see the Attributes table.

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfapplet`, `cfcalendar`, `cfform`, `cfformgroup`, `cfformitem`, `cfgrid`, `cfinput`, `cfslider`, `cftextarea`, `cftree`; "Introduction to Retrieving and Formatting Data" on page 512 and "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*

History

ColdFusion 8:

- Added support for binding in HTML format forms, including the `bind`, `bindAttribute`, and `bindOnLoad`, and `onBindError` attributes.
- Added support for tool tips in HTML format forms, including the `sourceForTooltip` attribute.

ColdFusion MX 7:

- When populating a `cfselect` tag with a query when the database field has spaces instead of a value, the error processing of the `cfform` tag required field is not triggered as it was in ColdFusion MX 6.1.
- Added support for specifying multiple values to the `selected` attribute.
- Deprecated the `passthrough` attribute. The tag now supports all HTML `select` tag attributes directly.
- Added `on`-prefixed attributes.
- Added `enabled`, `group`, `height`, `label`, `queryPosition`, `tooltip`, `visible`, and `width` attributes.

Attributes

The following table lists attributes that ColdFusion uses directly. The tag also supports all HTML `select` tag attributes that are not on this list, and passes them directly to the browser.

Note: *Attributes that are marked as Flash only are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.*

Attribute	Req/Opt; Format	Default	Description
`name`	Required; All		Name of the select form element.
`bind`	Optional; HTML		A bind expression that dynamically sets an attribute of the control. For details, see Usage.
`bindAttribute`	Optional; HTML	Populate the options	Specifies the HTML tag attribute whose value is set by the `bind` attribute. You can only specify attributes in the browser's HTML DOM tree, not ColdFusion-specific attributes. Ignored if there is no `bind` attribute.
`bindOnLoad`	Optional; HTML	`no`	A Boolean value that specifies whether to execute the `bind` attribute expression when first loading the form. Ignored if there is no `bind` attribute.
`display`	Optional; All	Value of `value` attribute	Query column to use for the display label of each list element. Used with the `query` attribute.
`editable`	Optional; Flash	`no`	Boolean value specifying whether you can edit the contents of the control.
`enabled`	Optional; Flash	`yes`	Boolean value specifying whether the control is enabled. A disabled control appears in light gray. The inverse of the `disabled` attribute.
`group`	Optional; HTML and XML		Query column to use to group the items in the drop-down list into a two-level hierarchical list.
`height`	Optional; Flash		The height of the control, in pixels.
`id`	Optional; HTML	Value of `name` attribute	The HTML ID of the control.

Attribute	Req/Opt; Format	Default	Description
label	Optional; Flash and XML		Label to put next to the control on a Flash or XML-format form.
message	Optional; All		Message to display if required = "yes" and no selection is made.
multiple	Optional; All	no	• yes: allows selecting multiple elements in drop-down list. • no
onBindError	Optional; HTML	See Description	The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. (The status code is -1 if the error is not an HTTP error.) If you omit this attribute, and specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up appears.
onChange	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the control changes due to user action.
onClick	Optional; HTML and XML		JavaScript to run when the user clicks the control.
onError	Optional; HTML and XML		Custom JavaScript function to execute if validation fails.
onKeyDown	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user presses a keyboard key in the control.
onKeyUp	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a keyboard key in the control.
onMouseDown	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a mouse button in the control.
onMouseUp	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user presses a mouse button in the control.
query	Optional; All		Name of query to populate drop-down list.
queryPosition	Optional; All	above	If you populate the options list with a query and use HTML option child tags to specify additional entries, this attribute determines the location of the items from the query relative to the items from the option tags: • above: puts the query items above the options items. • below: puts the query items below the options items.
required	Optional; All	no	***Note:*** This attribute has no effect if you omit the size attribute or set it to 1, because the browser always submits the displayed item. You can work around this issue: format forms by having an initial option tag with value=" " (notice the space character between the quotation marks). • yes: a list element must be selected when the form is submitted. • no
selected	Optional; All		One or more option values to preselect in the selection list. To specify multiple values, use a comma-delimited list. This attribute applies only if selection list items are generated from a query. The cfform tag preserveData attribute can override this value.

Attribute	Req/Opt; Format	Default	Description
size	Optional; All	1	Number of entries to display at one time. The default value displays a drop-down list. Any other value displays a list box with size number of entries visible at one time.
sourceForTooltip	Optional; HTML		The URL of a page to display as a tool tip. The page can include CFML and HTML markup to control the tip contents and format, and the tip can include images. If you specify this attribute, an animated icon appears with the text "Loading..." while the tip is being loaded.
style	Optional; All		In HTML or XML format forms, ColdFusion passes the style attribute to the browser or XML. In Flash format, must be a style specification in CSS format, with the same syntax and contents as used in Flex for the corresponding Flash element.
tooltip	Optional; Flash, HTML		Text to display when the mouse pointer hovers over the control. The text can include HTML markup. Ignored if you specify a sourceForTooltip attribute.
value	Optional; All		Query column to use for the value of each list element. Used with the query attribute.
visible	Optional; Flash	yes	Boolean value that specifies whether to show the control. The display reserves empty space for an invisible control.
width	Optional; Flash		The width of the control, in pixels.

Usage

For this tag to work properly. the browser must have JavaScript enabled.

This tag requires an end tag and can include HTML option and optgroup child tags.

To ensure that a selected list box item persists across postbacks, use the cfform tag preserveData attribute with a list generated from a query. (This strategy works only with data that is populated from a query.)

If the cfform preserveData attribute is yes and the form posts back to the same page, and if the control is populated by a query, the posted selections for the cfselect control are used instead of the Selected attribute. For controls that are populated with regular HTML option tags, the developer must dynamically add the Selected attribute to the appropriate option tags.

The group option generates a query by using SQL GROUP BY syntax and places the value column entries from each group in an indented list under the group column's field value. This option generates an HTML optgroup tag for each entry in the group column.

Close each HTML option tag in the cfselect tag body with a </option> end tag. If you do not do so, and you specify queryPosition="below", the first item from the query might not appear in the list.

The bind attribute lets you set cfselect attributes dynamically. Often, it is used to dynamically create the options list based on values that the user enters in the form. For example, you can use the bind attribute to create a Cities option list based on the user's selection from a States cfselect control.

When you use a bind attribute to populate the selection list, the function or URL that returns the selection values must return one of the following:

- A two-dimensional array, where the first element in each array row is the option value and the second element in the row is the text to display in the option list.

• If the bind specifies a CFC function, a query, or, if the bind specifies a URL, a JSON representation of a query. The query must include columns whose names are the values of the `cfselect` tag `value` and `display` attributes. Although you can return additional columns, you cannot use them in your client-side code. When you call a CFC function, you do not have to convert the query to JSON format yourself; ColdFusion automatically does the conversion.

  To use this format, you must specify a `value` attribute. If you omit the `display` attribute, you must have only a single column in the query that contains the values; the values are also used as the displayed text.

For detailed information on binding, see "Binding data to form fields" on page 650 in the *ColdFusion Developer's Guide*.

For more information, see the `cfform` tag entry.

Example

Example 1: Without data binding

The following example lets you select one or more employee names from a list of all employees, grouped by departments, and displays the selected names and the employee's e-mail addresses. It includes an option to get data for all employees.

```
<!--- Get the employee names from the database. --->
<!--- Use SQL to create a Name field with first and last names. --->
<cfquery name = "GetAllEmployees" dataSource = "cfdocexamples"
        cachedwithin="#createTimeSpan(0,1,0,0)#">
    SELECT Emp_ID, EMail, Phone, Department, FirstName, LastName,
        FirstName ||' '
        ||lastName as Name
    FROM Employees
    GROUP BY Department, Emp_ID, EMail, Phone, FirstName, LastName, FirstName
</cfquery>

<h2>cfselect Example</h2>
<!-- The cfif statement is true if the form was submitted.
    Show the selected names. --->
<cfif IsDefined("form.employeeid")>
    <!--- The form was submitted. --->
    <h4>You Selected the following employees</h3>
    <cfif form.employeeid IS "">
        <!--- Select All option was selected. Show all employees. --->
        <cfoutput query="GetAllEmployees">
            #name#<br>
            Email: #email#<br><br>
        </cfoutput>
    <cfelse>
        <!---
            Use a query of queries to get the data for the selected users.
            Form.employeeid is a comma-delimited list of selected employee IDs.
        --->
        <cfquery name = "GetSelectedEmployees" dbtype="query">
            SELECT Emp_ID, EMail, Phone, Department, FirstName, LastName,
                FirstName
                ||' ' ||lastName as Name
            FROM GetAllEmployees
            WHERE Emp_ID in (#form.employeeid#)
        </cfquery>
        <!--- Display the names and e-mail addresses from the query. --->
        <cfoutput query="GetSelectedEmployees">
            #firstName# #lastName#<br>
```

```
            Email: #email#<br>
            <br>
        </cfoutput>
    </cfif>
</cfif>

<!--- The cfform tag posts back to the current page. --->
<h3>Select one or more employees</h3>
<cfform action = "#CGI.SCRIPT_NAME#">
    <!---
            Use cfselect to present the query's LastName column,
            grouped by department.
            Allow Multiple selections.
        --->
    <cfselect
        name = "employeeid"
        size = "15"
        multiple="yes"
        required = "Yes"
        message = "Select one or more employee names"
        query = "GetAllEmployees"
        group="Department"
        display ="name"
        value ="emp_id"
        queryPosition="Below">
        <!--- Add an option to select all employees. --->
        <option value = "">Select All</option>
    </cfselect><br><br>
    <input type="Submit">
</cfform>
```

Example 2: With data binding

The following example uses binding to fill in the options list of the Cities control only after the user selects a state. (In this example, only two states, California and New Jersey, have city entries.)

The CFML page is the simplest part of the example. It consists of the following lines:

```
<html>
<head>
</head>
<body>
<cfform name="mycfform">
    <!---
        The States selector.
        The bindonload attribute is required to fill the selector.
    --->
    <cfselect name="state" bind="cfc:bindFcns.getstates()" bindonload="true">
        <option name="0">--state--</option>
    </cfselect>
    <cfselect name="city" bind="cfc:bindFcns.getcities({state})">
        <option name="0">--city--</option>
    </cfselect>
</cfform>
</body>
</html>
```

The BinFcns CFC has three functions: `getstates`, to get the states; `getcities`, to get the cities; and an internal `getXmlData` function that parses an XML file to get the state and city information. The following examples shows the CFC:

```
<cfcomponent>
```

```
    <cffunction name="getXmlData" output="true">
        <cfset var xmlData = "">
        <cffile action="read" file="#expandpath('.')#\states.xml"
            variable="xmlData">
        <cfset xmlData = XmlParse(xmlData)>
        <cfreturn xmlData>
    </cffunction>

    <cffunction name="getstates" access="remote">
        <cfset state = arraynew(2)>
        <cfset xmlData = getXmlData()>
        <cfset numStates = 0>
        <cfset state[1][1] = "0">
        <cfset state[1][2] = "--state--">
        <cfset numStates = ArrayLen(xmlData.states.XmlChildren)>
        <cfloop from="1" to="#numStates#" index="j">
            <cfset state[j+1][1] =
                ltrim(xmlData.states.state[j].XmlAttributes.abr)>
            <cfset state[j+1][2] = ltrim(xmlData.states.state[j].name.xmlText)>
        </cfloop>
        <cfreturn state>
    </cffunction>

    <cffunction name="getcities" access="remote">
        <cfargument name="state" required="yes">
        <cfset var city = arraynew(2)>
        <cfset var xmlData = getXmlData()>
        <cfset var numStates = 0>
        <cfset var numCities = 0>
<cflog text="In getcities">
        <cfset city[1][1] = "0">
        <cfset city[1][2] = "--city--">
        <cftry>
            <cfset numStates = ArrayLen(xmlData.states.XmlChildren)>
            <cfloop from="1" to="#numStates#" index="j">
                <cfif xmlData.states.state[j].XmlAttributes.abr eq state>
                    <cfset numCities =
                        ArrayLen(xmlData.states.state[j].cities.XmlChildren)>
                    <cfloop from="1" to="#numCities#" index="k">
                        <cfset city[k+1][1] =
ltrim(xmlData.states.state[j].cities.city[k].XmlAttributes.name)>
                        <cfset city[k+1][2] =
ltrim(xmlData.states.state[j].cities.city[k].XmlAttributes.name)>
                    </cfloop>
                    <cfbreak>
                </cfif>
            </cfloop>
        <cfcatch type="any">
            <!--- Do nothing. --->
        </cfcatch>
        </cftry>
        <cfreturn city>
    </cffunction>

</cfcomponent>
```

The states.xml file has the following code. To keep the code short, only two states have cities, and only four states are listed.

```
<states>
    <state abr="NJ">
```

```
        <name>New Jersey</name>
        <cities>
            <city name="Edison" />
            <city name="Rahway" />
            <city name="Atlantic City" />
            <city name="Hoboken" />
            <city name="Jersey City" />
            <city name="Newark" />
            <city name="Trenton" />
            <city name="Union City" />
        </cities>
    </state>
    <state abr="CA">
        <name>California</name>
        <cities>
            <city name="Anaheim" />
            <city name="Beverly Hills" />
            <city name="Elk Grove" />
            <city name="Fairfield" />
            <city name="Fremont" />
            <city name="Indian Wells" />
            <city name="Long Beach" />
        </cities>
    </state>
    <state abr="ME">
        <name>Maine</name>
    </state>
    <state abr="MA">
        <name>Massachusetts</name>
    </state>
</states>
```

cfservlet

Description

This tag is deprecated. Executes a Java servlet on a JRun engine.

To access servlets that run on the same server as ColdFusion, use code such as the following, in which *path* specifies a servlet, JSP, or anything else:

```
GetPageContext().include(path)
GetPageContext().forward(path)
```

For more information, see the JSP PageContext API or the Servlet RequestDispatcher API.

History

ColdFusion MX: Deprecated this tag. It might not work, and it might cause an error, in later releases.

cfservletparam

Description

This tag is deprecated.

A child tag of the `cfservlet` tag. Passes data to a servlet. Each `cfservletparam` tag within the `cfservlet` block passes a separate item of data to the servlet.

To access servlets that run on the same server as ColdFUsion, use code such as the following, in which *path* specifies a servlet, JSP, or anything else:

```
GetPageContext().include(path)
GetPageContext().forward(path)
```

For more information, see the JSP PageContext API or the Servlet RequestDispatcher API.

History

ColdFusion MX: Deprecated this tag. It might not work, and it might cause an error, in later releases.

cfset

Description

Sets a value in ColdFusion. Used to create a variable, if it does not exist, and assign it a value. Also used to call functions.

Category

Variable manipulation tags

Syntax

```
<cfset
    var variable_name = expression>
```

See also

cfcookie, cfparam, cfregistry, cfsavecontent, cfschedule; "Elements of CFML" on page 10 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
variable_name	Required		A variable.
var	Optional		A keyword. Does not take a value. Identifies the variable as being local to a function. The variable only exists for the time of the current invocation of the function.

Usage

You use the cfset tag in several ways in your applications.

Calling functions

When you use the cfset tag to call a function, you do not have to assign the function return value to a variable if the function does not return a value or you do not have to use the value returned by the function. For example, the following line is a valid ColdFusion cfset tag for deleting the MyVariable variable from the Application scope:

```
<cfset StructDelete(Application, "MyVariable")>
```

Arrays

The following example assigns a new array to the variable months:

```
<cfset months = ArrayNew(1)>
```

This example creates a variable Array_Length that resolves to the length of the Scores array:

```
<cfset Array_Length = ArrayLen(Scores)>
```

This example assigns, to index position two in the array months, the value February:

```
<cfset months[2] = "February">
```

Dynamic variable names

In this example, the variable name is itself a variable:

```
<cfset myvariable = "current_value">
<cfset "#myvariable#" = 5>
```

Function local variables

The `var` keyword specifies that the variable being defined is only available inside the body of a function that you define by using the `cffunction` tag. The variable value that is set in one invocation of the function is not available in any other invocation of the function. The `var` keyword is the equivalent of the `var` statement in CFScript. The following rules apply to the `var` keyword:

- Any `cfset` tag that uses the `var` keyword must be inside the body of a `cffunction` tag. If you use the `var` keyword in a `cfset` tag outside a `cffunction` tag body, ColdFusion displays an error message.
- You must place all `cfset` tags that use the `var` keyword at the beginning of the `cffunction` tag body, before any other ColdFusion tags.

The following example shows how to use the new keyword:

```
<cffunction name="myFunct">
 <cfset var myVar = "This is a test">
 <cfreturn myVar & " Message.">
</cffunction>
<cfoutput>#myFunct()#</cfoutput>
```

In this example, the variable myVar exists only when the function myFunct executes, and it is not available elsewhere on the ColdFusion page.

COM objects

In this example, a COM object is created. A `cfset` tag defines a value for each method or property in the COM object interface. The last `cfset` creates a variable to store the return value from the COM object's `SendMail` method.

```
<cfobject action = "Create"
    name = "Mailer"
    class = "SMTPsvg.Mailer">

<cfset MAILER.FromName = form.fromname>
<cfset MAILER.RemoteHost = RemoteHost>
<cfset MAILER.FromAddress = form.fromemail>
<cfset MAILER.AddRecipient("form.fromname", "form.fromemail")>
<cfset MAILER.Subject = "Testing cfobject">
<cfset MAILER.BodyText = "form.msgbody">
<cfset Mailer.SMTPLog = "logfile">
<cfset success = MAILER.SendMail()>
<cfoutput> #success# </cfoutput>
```

Example

```
<!--- This example shows how to use cfset. --->
<cfquery name = "GetMessages" dataSource = "cfdocexamples">
    SELECT *
    FROM Messages
</cfquery>

<h3>cfset Example</h3>
<p>cfset sets and reassigns values to local or global variables within a page.

<cfset NumRecords = GetMessages.recordCount>
<p>For example, the variable NumRecords has been declared on this
    page to hold the number of records returned from query
    (<cfoutput>#NumRecords#</cfoutput>).

<p>In addition, cfset can be used to pass variables from other pages,
        such as this example, which takes the url parameter Test from this
        link: <a href = "cfset.cfm?test = <cfoutput>
        #URLEncodedFormat("hey, you, get off of my cloud")#</cfoutput>
```

```
        ">click here</A>) to display a message:
<p>
<cfif IsDefined ("url.test") is "True">
    <cfoutput><b><I>#url.test#</i></b></cfoutput>
<cfelse>
    <h3>The variable url.test has not been passed from another page.</h3>
</cfif>

<p>cfset can also be used to collect environmental variables, such as the
    time, the IP address of the user, or another function or expression.

<cfset the_date = #DateFormat(Now())# & " " & #TimeFormat(Now())#>
<cfset user_ip = CGI.REMOTE_ADDR>
<cfset complex_expr = (23 MOD 12) * 3>
<cfset str_example = Reverse(Left(GetMessages.body, 35))>

<cfoutput>
    <ul>
        <li>The date: #the_date#
        <li>User IP Address: #user_ip#
        <li>Complex Expression ((23 MOD 12) * 3): #complex_expr#
        <li>String Manipulation (the first 35 characters of
                the body of the first message in our query)
                <br><b>Reversed</b>: #str_example#
                <br><b>Normal</b>: #Reverse(str_example)#
    </ul>
</cfoutput>
```

cfsetting

Description

Controls aspects of page processing, such as the output of HTML code in pages.

Category

Page processing tags, Variable manipulation tags

Syntax

```
<cfsetting
    enableCFoutputOnly = "yes|no"
    requestTimeOut = "value in seconds"
    showDebugOutput = "yes|no" >
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcache`, `cfflush`, `cfheader`, `cfhtmlhead`, `cfinclude`, `cfprocessingdirective`, `cfsilent`; "Controlling debugging output with the cfsetting tag" on page 362 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 6.1: Changed behavior: if the tag has a body, ColdFusion executes its contents.

ColdFusion MX:

- Added the `requestTimeOut` attribute.
- The `catchExceptionsByPattern` attribute is obsolete. It does not work, and causes an error, in releases later than ColdFusion 5.
- Changed exception handling: the structured exception manager searches for the best-fit `cfcatch` handler. (In earlier releases, an exception was handled by the first `cfcatch` block that could handle an exception of its type.)

Attributes

Attribute	Req/Opt	Default	Description
`enableCFoutputOnly`	Required		• `yes`: blocks output of HTML that is outside `cfoutput` tags. • `no`: displays HTML that is outside `cfoutput` tags.
`requestTimeout`	Optional		• `integer`; number of seconds. Time limit, after which ColdFusion processes the page as an unresponsive thread. Overrides the time-out set in the ColdFusion Administrator.
`showDebugOutput`	Optional	yes	• `yes`: if debugging is enabled in the Administrator, displays debugging information. • `no`: suppresses debugging information that would otherwise display at the end of the generated page.

Usage

The `cfsetting requestTimeout` attribute replaces the use of `requestTimeOut` within a URL. To enforce a page time-out, detect the URL variable and use code such as the following to change the page time-out:

```
<cfsetting RequestTimeout = "#URL.RequestTimeout#">
```

You can use this tag to manage whitespace in ColdFusion output pages.

If you nest `cfsetting` tags: to make HTML output visible, you must match each `enableCFoutputOnly = "Yes"` statement with an `enableCFoutputOnly = "No"` statement. For example, after five `enableCFoutputOnly = "Yes"` statements, to enable HTML output, you must have five corresponding `enableCFoutputOnly = "No"` statements.

If HTML output is enabled (no matter how many `enableCFoutputOnly = "No"` statements have been processed) the first `enableCFoutputOnly = "Yes"` statement blocks output.

If the debugging service is enabled and `showDebugOutput =" Yes"`, the `IsDebugMode` function returns Yes; otherwise, No.

Note: *Releases after ColdFusion MX allow a `</cfsetting>` end tag; however, this end tag does not affect processing. The `cfsetting` attributes affect code inside and outside the `cfsetting` tag body. ColdFusion MX ignored code between `cfsetting` start and end tags.*

Example

```
<p>CFSETTING is used to control the output of HTML code in ColdFusion pages.
    This tag can be used to minimize the amount of generated whitespace.

<cfsetting enableCFoutputOnly = "Yes">
    This text is not shown
<cfsetting enableCFoutputOnly = "No">
    <p>This text is shown
<cfsetting enableCFoutputOnly = "Yes">
    <cfoutput>
        <p>Text within cfoutput is always shown
    </cfoutput>
<cfsetting enableCFoutputOnly = "No">
    <cfoutput>
        <p>Text within cfoutput is always shown
    </cfoutput>
```

cfsilent

Description

Suppresses output produced by CFML within a tag's scope.

Category

Data output tags, Page processing tags

Syntax

```
<cfsilent>
    ...
</cfsilent>
```

See also

cfcache, cfflush, cfheader, cfhtmlhead, cfinclude, cfsetting; "Writing and Calling User-Defined Functions" on page 134 in the *ColdFusion Developer's Guide*

Usage

This tag requires an end tag.

Example

```
<h3>cfsilent</h3>

<cfsilent>
<cfset a = 100>
<cfset b = 99>
<cfset c = b-a>
<cfoutput>Inside cfsilent block<br>
b-a = #c#</cfoutput><br>
</cfsilent>

<p>Even information within cfoutput tags does not display within a
cfsilent block.<br>
<cfoutput>
b-a = #c#
</cfoutput>
</p>
```

cfslider

Description

Puts a slider control, for selecting a numeric value from a range, in a ColdFusion form. The slider moves over the slider groove. As the user moves the slider, the current value displays. Used within a `cfform` tag. Not supported with Flash forms.

Category

Forms tags

Syntax

```
<cfslider
    name = "name"
    align = "top|left|bottom|baseline|texttop|absbottom|
        middle|absmiddle|right"
    bgColor = "color"
    bold = "yes|no"
    font = "font name"
    fontSize = "integer"
    height = "integer"
    hSpace = "integer"
    italic = "yes|no"
    label = "text"
    lookAndFeel = "motif|windows|metal"
    message = "text"
    notSupported = "text"
    onError = "text"
    onValidate = "script name"
    range = "minimum value, maximum value"
    scale = "integer"
    textColor = "color"
    value = "integer"
    vertical = "yes|no"
    vSpace = "integer"
    width = "integer">
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfapplet`, `cfcalendar`, `cfform`, `cfformgroup`, `cfformitem`, `cfgrid`, `cfinput`, `cfselect`, `cftextarea`, `cftree`; "Introduction to Retrieving and Formatting Data" on page 512 and "Building Dynamic Forms with cfform Tags" on page 531 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Deprecated the `img`, `imgStyle`, `grooveColor`, `refreshLabel`, `tickmarklabels`, `tickmarkmajor`, `tickmarkminor`, and `tickmarkimages` attributes. They might not work, and might cause an error, in later releases.

Attributes

Attribute	Req/Opt	Default	Description
name	Required		Name of cfslider control.
align	Optional		Alignment of slider: • top • left • bottom • baseline • texttop • absbottom • middle • absmiddle • right
bgColor	Optional		Background color of slider label. For a hexadecimal value, use the form: bgColor = "##xxxxxx", where x = 0-9 or A-F; use two number signs or none. • any color, in hexadecimal format • black • red • blue • magenta • cyan • orange • darkgray • pink • gray • white • lightgray • yellow
bold	Optional	no	• yes: label text in bold. • no: medium text.
font	Optional		Font name for label text.
fontSize	Optional		Font size for label text, in points.
height	Optional	40	Slider control height, in pixels.
hSpace	Optional		Horizontal spacing to left and right of slider, in pixels.
italic	Optional	no	• yes: label text in italics. • no: normal text.
label	Optional		Label to display with control; for example, "Volume" This displays: "Volume %value%" To reference the value, use "%value%". If %% is omitted, slider value displays directly after label.

Attribute	Req/Opt	Default	Description
lookAndFeel	Optional	Windows	• motif: renders slider using Motif style. • windows: renders slider using Windows style. • metal: renders slider using Java Swing style. If platform does not support choice, the tag defaults to the platform's default style.
message	Optional		Message text to appear if validation fails.
notSupported	Optional		Text to display if a page that contains a Java applet-based cfform control is opened by a browser that does not support Java or has Java support disabled. For example: " Browser must support Java to view ColdFusion Java Applets" Default message: Browser must support Java to view ColdFusion Java Applets!
onError	Optional		Custom JavaScript function to execute if validation fails. Specify only the function name.
onValidate	Optional		Custom JavaScript function to validate user input; in this case, a change to the default slider value. Specify only the function name.
range	Optional	"0,100"	Numeric slider range values. Separate values with a comma.
scale	Optional		Unsigned integer. Defines slider scale, within range. For example, if range = "0,1000" and scale = "100", the display values are: 0, 100, 200, 300, ... Signed and unsigned integers in ColdFusion are in the range -2,147,483,648 to 2,147,483,647.
textColor	Optional		Options: same as for bgcolor attribute.
value	Optional	Minimum in range	Starting slider setting. Must be within the range values.
vertical	Optional	no	• yes: renders slider in browser vertically. You must set width and height attributes; ColdFusion does not automatically swap width and height values. • no: renders slider horizontally.
vSpace	Optional		Vertical spacing above and below slider, in pixels.
width	Optional		Slider control width, in pixels.

Usage

This tag requires the client to download a Java applet. Using this tag may be slightly slower than using an HTML form element to display the same information. Also, if the client does not have an up-to-date Java plugin installed, the system might also have to download an updated Java plugin to display the tag.

For this tag to work properly, the browser must be JavaScript-enabled.

If the following conditions are true, a user's selection from query data that populates this tag's options continues to display after the user submits the form:

- The cfform preserveData attribute is set to "Yes".
- The cfform action attribute posts to the same page as the form itself (this is the default), or the action page has a form that contains controls with the same names as corresponding controls on the user entry form.

For more information, see the cfform tag entry.

Example

```
<!--- This example shows how to use cfslider within cfform. --->
<h3>cfslider Example</h3>
<p>cfslider, used within a cfform, can provide functionality
    to Java-enabled browsers.
<p>Try moving the slider back and forth to see the real-time value change.
    Then, submit the form to show how cfslider passes its value on to a new page.</p>
<cfif isdefined("form.mySlider") is true>
    <h3>You slid to a value of <cfoutput>#mySlider#</cfoutput></h3>
    Try again!
</cfif>
<cfform action = "cfslider.cfm">
    <cfslider name = "mySlider" value = "12"
        label = "Actual Slider Value "
        range = "1,100" align = "BASELINE"
        message = "Slide the bar to get a value between 1 and 100"
        height = "50" width = "150" font = "Verdana"
        bgColor = "Silver" bold = "No"
        italic = "Yes" refreshLabel = "Yes"> 100
<p><input type = "Submit" name = "" value = "Show the Result"></p>
</cfform>
```

cfsprydataset

Description

Creates a Spry XML or JSON data set from the results of a bind expression.

Category

Internet protocol tags

Syntax

```
<cfsprydataset
    bind = "bind expression"
    name = "data set name"
    onBindError = "JavaScript function name"
    options = "Spry options object"
    type = "xml|json"
    xpath = "XPath expression">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfajaximport`, "Using Spry with ColdFusion" on page 662 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`bind`	Required		A bind expression that returns an XML- or JSON- formatted string to populate the Spry data set. The bind expression specifies a CFC function or URL and includes bind parameters that represent the values of ColdFusion controls. For detailed information on bind expressions, see "Binding data to form fields" on page 650 in the *ColdFusion Developer's Guide*.
`name`	Required		The name of the Spry data set.
`onBindError`	Optional; HTML	See Description	The name of a JavaScript function to execute if the bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and specified a global error handler (by using the `ColdFusion.setGlobalErrorHandler` function), the handler displays the error message; otherwise a default error pop-up appears.
`options`	Optional		A JavaScript object containing constructor options for the data set. For example, to request the data using the HTTP POST method, specify the following attribute: `options="{method: 'POST'}"`. For detailed information on Spry options, see the Spry documentation.
`type`	Optional	`xml`	Specifies data set type, corresponding to the format of the data that is returned by the bind expression. The following values are valid: • `json` • `xml`
`xpath`	Required for xml type Not used for JSON		An XPath expression that extracts data from the XML returned by the bind expression. The data set contains only the data that matches the XPath expression.

Usage

Use this tag to use a bind expression to dynamically create the contents of a Spry XML or JSON data set based on the value of a ColdFusion control or another Spry data set. To create a Spry data set without using a bind expression, use the `Spry.Data.JSONDataSet()` and `Spry.Data.XMLDataSet()` JavaScript functions.

This tag cannot create a Spry HTML data set.

To use a filter to select the contents of a JSON data set from a JSON expression, specify a `path` or `subpath` option in the `options` attribute. For example, to create a Spry JSON data set by using only the items.item element from the JSON data, use a tag such as the following:

```
<cfsprydataset name="theItems" type="json"
    bind="CFC:dataMgr.getdetails(prodname={myform:mygrid.TITLE})"
    options="{path: 'items.item.'}">
```

Example

The following `cfsprydataset` tag updates the dsProduct Spry XML data set by calling the `GridDataManager.getProductDetails` CFC function each time the selected row in the bookgird control changes. It passes the TITLE field of the selected row to the CFC function as a `prodname` parameter. For a complete example that uses this tag, see "Using Spry with ColdFusion" on page 662 in the *ColdFusion Developer's Guide*.

```
<cfsprydataset
    name="dsProduct"
    type="xml"
    bind="CFC:GridDataManager.getProductDetails(prodname=
        {bookform:bookgrid.TITLE})"
    xpath="products/product"
    options="{method: 'POST'}"
    onBindError="errorHandler">
```

cfstoredproc

Description

Executes a stored procedure in a server database. It specifies database connection information and identifies the stored procedure.

Category

Database manipulation tags

Syntax

```
<cfstoredproc
    dataSource = "data source name"
    procedure = "procedure name"
    debug = "yes|no"
    blockFactor = "block size"
    password = "password"
    result = "result name"
    returnCode = "yes|no"
    username = "user name">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfinsert`, `cfqueryparam`, `cfprocparam`, `cfprocresult`, `cftransaction`, `cfquery`, `cfupdate`; "Optimizing database use" on page 243 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added the `result` attribute.

ColdFusion MX: Deprecated the `connectString`, `dbName`, `dbServer`, `dbtype`, `provider`, and `providerDSN` attributes. They do not work, and might cause an error, in releases later than ColdFusion 5. (Releases starting with ColdFusion MX use Type 4 JDBC drivers.)

Attributes

Attribute	Req/Opt	Default	Description
`dataSource`	Required		Name of data source that points to database that contains stored procedure.
`procedure`	Required		Name of stored procedure on database server.
`blockFactor`	Optional	`1`	Maximum number of rows to get at a time from server. Range is 1 to 100.
`debug`	Optional	`no`	• `yes`: lists debug information on each statement. • `no`
`password`	Optional		Overrides password in data source setup.
`result`	Optional		Specifies a name for the structure in which `cfstoredproc` returns the statusCode and ExecutionTime variables. If set, this value replaces `cfstoredproc` as the prefix to use when accessing those variables. For more information, see Usage.
`returnCode`	Optional	`no`	• `yes`: populates `cfstoredproc.statusCode` with status code returned by the stored procedure. • `no`
`username`	Optional		Overrides username in data source setup.

Usage

Use this tag to call a database stored procedure. Within this tag, you code cfprocresult and cfprocparam tags as follows:

- cfprocresult: If the stored procedure returns one or more result sets, code one cfprocresult tag per result set.
- cfprocparam: If the stored procedure uses input or output parameters, code one cfprocparam tag per parameter, ensuring that you include every parameter in the stored procedure definition.

If you set returnCode = "Yes", this tag sets the variable *prefix*.statusCode, which holds the status code for a stored procedure. Status code values vary by DBMS. For the meaning of code values, see your DBMS documentation.

This tag sets the variable *prefix*.ExecutionTime, which contains the execution time of the stored procedure, in milliseconds.

The value of *prefix* is either cfstoredproc or the value specified by the result attribute, if it is set. The result attribute provides a way for stored procedures that are called from multiple pages, possibly at the same time, to avoid overwriting the results of one call with another. If you set the result attribute to myResult, for example, you would access ExecutionTime as myResult.ExecutionTime. Otherwise, you would access it as cfstoredproc.ExecutionTime.

Before implementing this tag, ensure that you understand stored procedures and their usage.

The following examples use a Sybase stored procedure; for an example of an Oracle 8 or 9 stored procedure, see cfprocparam.

Example

```
<cfset ds = "sqltst">

<!---
If submitting a new book,
insert the record and display
confirmation --->
<cfif isDefined("form.title")>

<cfstoredproc procedure="Insert_Book" datasource="#ds#">

<cfprocparam
    cfsqltype="cf_sql_varchar"
    value="#form.title#">

<cfprocparam
    cfsqltype="cf_sql_numeric"
    value="#form.price#">

<cfprocparam
    cfsqltype="cf_sql_date"
    value="#form.price#">

<cfprocparam
    cfsqltype="cf_sql_numeric"
    type="out"
    variable="bookId">

</cfstoredproc>

<cfoutput>
<h3>'#form.title#' inserted into database.The ID is #bookId#.</h3>
```

```
</cfoutput>

</cfif>

<cfform action="#CGI.SCRIPT_NAME#" method="post">
<h3>Insert a new book</h3>

Title:
<cfinput type="text" size="20" required="yes" name="title"/>
<br/>

Price:
<cfinput type="text" size="20" required="yes" name="price" validate="float" />
<br/>

Publish Date:
<cfinput type="text" size="5" required="yes" name="publishDate" validate="date" />
<br/>

<input type="submit" value="Insert Book"/>

</cfform>
```

cfswitch

Description

Evaluates a passed expression and passes control to the cfcase tag that matches the expression result. You can, optionally, code a cfdefaultcase tag, which receives control if there is no matching cfcase tag value.

Category

Flow-control tags

Syntax

```
<cfswitch
    expression = "expression">
        one or more cfcase tags
        zero or one cfdefaultcase tags
</cfswitch>
```

See also

cfcase, cfdefaultcase, cfabort, cfloop, cfbreak, cfexecute, cfexit, cfif, cflocation, cfrethrow, cfthrow, cftry; "cfswitch, cfcase, and cfdefaultcase" on page 18 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Changed the way the ColdFusion parses <cfcase> values. Previously, <cfcase> tags with numeric value dates did not return expected results. For example, <cfcase value="00"> and <cfcase value="0A> were both evaluated to 0. The value "0A" was treated as a date and converted to 0 number of days from 12/30/1899. The value "00" was also evaluated to the value 0. This caused the exception "Context validation error for tag CFCASE. The CFSWITCH has a duplicate CFCASE for value "0.0"." The <cfswitch> tag now returns the expected result.

ColdFusion MX: Changed cfdefaultcase tag placement requirements: you can put the cfdefaultcase tag at any position within a cfswitch statement; it is not required to be the last item.

Attributes

Attribute	Req/Opt	Default	Description
expression	Required		ColdFusion expression that yields a scalar value. ColdFusion converts integers, real numbers, Booleans, and dates to numeric values. For example, true, 1, and 1.0 are all equal.

Usage

This tag requires an end tag. All code within this tag must be within a cfcase or cfdefaultcase tag. Otherwise, ColdFusion throws an error.

Use this tag followed by one or more cfcase tags. Optionally, include a cfdefaultcase tag. This tag selects the matching alternative from the cfcase and cfdefaultcase tags, jumps to the matching tag, and executes the code between the cfcase start and end tags.

The cfswitch tag provides better performance than a series of cfif/cfelseif tags, and the code is easier to read.

Example

```
<!---
    This example shows the use of cfswitch and cfcase to
    exercise a case statement in CFML.
--->
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
    SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
    FROM Employees
```

```
</cfquery>

<h3>cfswitch Example</h3>
<!--- By outputting the query and using cfswitch, we classify the
    output without using a cfloop construct. --->
<p>Each time the case is fulfilled, the specific information is printed;
if the case is not fulfilled, the default case is output </p>
<cfoutput query="GetEmployees">
<cfswitch expression="#Trim(Department)#">
    <cfcase value="Sales">
        #FirstName# #LastName# is in <b>sales</b><br><br>
    </cfcase>
    <cfcase value="Accounting">
        #FirstName# #LastName# is in <b>accounting</b><br><br>
    </cfcase> <cfcase value="Administration">
        #FirstName# #LastName# is in <b>administration</b><br><br>
    </cfcase>
    <cfdefaultcase>
        #FirstName# #LastName# is not in Sales, Accounting, or
            Administration.<br><br>
    </cfdefaultcase>
</cfswitch>
</cfoutput>
```

cftable

Description

Builds a table in a ColdFusion page. This tag renders data as preformatted text, or, with the `HTMLTable` attribute, in an HTML table. If you don't want to write HTML table tag code, or if your data can be presented as preformatted text, use this tag.

Preformatted text (defined in HTML with the `<pre>` and `</pre>` tags) displays text in a fixed-width font. It displays white space and line breaks exactly as they are written within the pre tags. For more information, see an HTML reference guide.

To define table column and row characteristics, use the `cfcol` tag within this tag.

Category

Data output tags

Syntax

```
<cftable
    query = "query name"
    border
    colHeaders
    colSpacing = "number of spaces"
    headerLines = "number of lines"
    HTMLTable
    maxRows = "maxrows table"
    startRow = "row number">
    ...
</cftable>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfcol`, `cfcontent`, `cflog`, `cfoutput`, `cfprocessingdirective`, `cftable`; "Retrieving data" on page 393 in the *ColdFusion Developer's Guide*

Attributes

Attribute	Req/Opt	Default	Description
`query`	Required		Name of `cfquery` from which to draw data.
`border`	Optional		Displays a border around the table.
			If you use this attribute (regardless of its value), ColdFusion displays a border around the table.
			Use this only if you use the `HTMLTable` attribute.
`colHeaders`	Optional		Displays column heads. If you use this attribute, you must also use the `cfcol` tag `header` attribute to define them.
			If you use this attribute (regardless of its value), ColdFusion displays column heads.
`colSpacing`	Optional	2	Number of spaces between columns.
`headerLines`	Optional	2	Number of lines to use for table header (the default leaves one line between header and first row of table).
`HTMLTable`	Optional		Renders data in an HTML 3.0 table.
			If you use this attribute (regardless of its value), ColdFusion renders data in an HTML table.

Attribute	Req/Opt	Default	Description
maxRows	Optional		Maximum number of rows to display in the table.
startRow	Optional	1	The query result row to put in the first table row.

Usage

This tag aligns table data, sets column widths, and defines column heads.

At least one cfcol tag is required within this tag. You must put cfcol and cftable tags adjacent in a page. The only tag that you can nest within this tag is the cfcol tag. You cannot nest cftable tags.

To display the cfcol header text, you must specify the cfcol header and the cftable colHeader attribute. If you specify either attribute without the other, the header does not display and no error is thrown.

Example

```
<!--- This example shows the use of cfcol and cftable to align information
    returned from a query. --->
<!--- This query selects employee information from cfdocexamples datasource. --->
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
    SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
    FROM Employees
</cfquery>

<html>
<body>
<h3>cftable Example</h3>

<!--- Note use of HTMLTable attribute to display cftable as an HTML table,
    rather than as PRE formatted information. --->
<cftable query = "GetEmployees"
    startRow = "1" colSpacing = "3" HTMLTable>
<!--- Each cfcol tag sets width of a column in table, and specifies header
    information and text/CFML with which to fill cell. --->
    <cfcol header = "<b>ID</b>"
        align = "Left"
        width = 2
        text= "#Emp_ID#">

    <cfcol header = "<b>Name/Email</b>"
        align = "Left"
        width = 15
        text= "<a href = 'mailto:#Email#'>#FirstName# #LastName#</A>">

    <cfcol header = "<b>Phone Number</b>"
        align = "Center"
        width = 15
        text= "#Phone#">
</cftable>
</body>
</html>
```

cftextarea

Description

Puts a multiline text entry box in a cfform tag and controls its display characteristics. Optionally, displays a rich text editor with configurable controls for formatting HTML text.

Category

Forms tags

Syntax

```
<cftextarea
    name="name"
    basepath="path"
    bind="bind expression"
    bindAttribute="attribute name"
    bindOnLoad="false|true"
    disabled="true|false" or no attribute value
    enabled="yes|no"
    fontFormats="comma separated list"
    fontNames="comma separated list"
    fontSizes="comma separated list"
    height="number of pixels"
    html="no|yes"
    label="text"
    maxlength="number"
    message="text"
    onBindError = "JavaScript function name"
    onChange="JavaScript or ActionScript"
    onClick="JavaScript or ActionScript"
    onError="script name"
    onKeyDown="JavaScript or ActionScript"
    onKeyUp="JavaScript or ActionScript"
    onMouseDown="JavaScript or ActionScript"
    onMouseUp="JavaScript or ActionScript"
    onValidate="script name"
    pattern="regexp"
    range="minimum value, maximum value"
    required="yes|no"
    richtext="no|yes"
    skin="default|silver|office2003|custom skin"
    sourceForToolTip="URL"
    style="style specification"
    stylesXML="path"
    templatesXML"path"
    toolbar="default|basic|custom toolbar"
    toolbarOnFocus"no|yes"
    tooltip="tip text"
    validate="data type"
    validateAt= one or more of "onBlur, onServer, onSubmit"
    value="text"
    visible="yes|no"
    width="number of pixels"
    wrap="off|hard|soft|virtual|physical">

        text

</cftextarea>
```

Some attributes apply to only specific display formats. For details see the Attributes table.

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfajaximport`, `cfapplet`, `cfcalendar`, `cfform`, `cfformgroup`, `cfformitem`, `cfgrid`, `cfinput`, `cfselect`, `cfslider`, `cftree`; "Introduction to Retrieving and Formatting Data" on page 512 and "Using Ajax form controls and features" on page 627 in the *ColdFusion Developer's Guide*

History

ColdFusion 8:

- Added support for the `bind` attribute in HTML format forms, including the `bindAttribute`, `bindOnLoad`, and `onBindError` attributes.
- Added support for tool tips in HTML format, including the `sourceForTooltip` attribute.
- Added support for a rich text editor in HTML format, including the `richtext`, `basepath`, `fontFormats`, `fontNames`, `fontSizes`, `skin`, `stylesXML`, `templatesXML`, `toolbar`, and `toolbarOnFocus` attributes and support for the `height` and `width` attributes.

ColdFusion MX 7: Added this tag.

Attributes

The following table lists attributes that ColdFusion uses directly. In HTML format, the tag also supports all HTML `textarea` tag attributes that are not on this list, and passes them directly to the browser.

Note: *Attributes that are not marked as All or XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.*

Attribute	Req/Opt; Format	Default	Description
`name`	Required; All		Name of the `cftextinput` control.
`basepath`	Optional; HTML	`/CFIDE/scripts/ajax/FCKEditor`	Path to the directory that contains the rich text editor. The editor configuration files are at the top level of this directory. Meaningful only if the `richText` attribute is `true`.
`bind`	Optional; Flash, HTML		A bind expression that dynamically sets an attribute of the control. For details, see Usage.
`bindAttribute`	Optional; HTML	`value`	Specifies the HTML tag attribute whose value is set by the `bind` attribute. You can only specify attributes in the browser's HTML DOM tree, not ColdFusion-specific attributes. Ignored if there is no `bind` attribute.
`bindOnLoad`	Optional; HTML	`no`	A Boolean value that specifies whether to execute the `bind` attribute expression when first loading the form. Ignored if there is no `bind` attribute.

Attribute	Req/Opt; Format	Default	Description
disabled	Optional; All	not disabled	Disables user input, making the control read-only. To disable input, specify disabled without an attribute, or disabled="Yes" (or any ColdFusion positive Boolean value, such as True). To enable input, omit the attribute or specify disabled="No" (or any ColdFusion negative Boolean value, such as False).
enabled	Optional; Flash	yes	Boolean value that specifies whether the control is enabled. A disabled control appears in light gray. The inverse of the disabled attribute.
fontFormats	Optional; HTML	All valid formats	A comma separated list of the font names to display in the rich text editor Formats selector. The formats specify the HTML tags to apply to typed or selected text. You can specify any subset of the following list: "p,div,pre,address,h1,h2,h3,h4,h5,h6".
fontNames	Optional; HTML	All valid font names	A comma separated list of the font names to display in the rich text editor Font selector. You can specify any subset of the following list: "Arial,Comic Sans MS,Courier New,Tahoma,Times New Roman,Verdana".
FontSizes	Optional; HTML	See Description	A comma separated list of the font sizes to display in the rich text editor Size selector. List entries must have the format of *numeric font size/descriptive text*. For example, to display the text "small font" to indicate a font size of 1, specify "1/small font". By default, the following values appear in the selector: 1/xx-small,2/x-small,3/small,4/medium,5/large,6/x-large,7/xx-large.
height	Optional; Flash, HTML		In Flash forms, height of the control, in pixels. In HTML forms, this attribute has an effect only if you specify richtext="true"; in this case, it is the height, in pixels, of the control, including the control bar and text box.
html	Optional; Flash	no	Boolean value that specifies whether the text area contains HTML.
label	Optional; Flash and XML		Label to put beside the control on a form.
maxLength	Optional; All		The maximum length of text that can be entered. ColdFusion uses this attribute only if you specify maxlength in the validate attribute.
message	Optional; All		Message text to display if validation fails.
onBindError	Optional; HTML	See Description	The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. (The status code is -1 if the error is not an HTTP error.) If you omit this attribute, and have specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up displays.
onChange	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the control changes due to user action.
onClick	Optional; HTML and XML		JavaScript to run when the user clicks the control.

Attribute	Req/Opt; Format	Default	Description
onError	Optional; HTML and XML		Custom JavaScript function to execute if validation fails.
onKeyDown	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) ActionScript to run when the user presses a keyboard key in the control.
onKeyUp	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a keyboard key in the control.
onMouseDown	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a mouse button in the control.
onMouseUp	Optional; All		JavaScript (HTML/XML) or ActionScript (Flash) to run when the user presses a mouse button in the control.
onValidate	Optional; HTML and XML		Custom JavaScript function to validate user input. The JavaScript DOM form object, input object, and input object value are passed to the function, which should return `True` if validation succeeds, `False` otherwise. If you specify this attribute, ColdFusion ignores the `validate` attribute.
pattern	Required if `validate = "regular_expression"` HTML and XML.		JavaScript regular expression pattern to validate input. Omit leading and trailing slashes. ColdFusion uses this attribute only if you specify *regex* in the `validate` attribute. For examples and syntax, see "Building Dynamic Forms with cfform Tags" on page 531 in the *ColdFusion Developer's Guide*.
range	Optional; All		Minimum and maximum allowed numeric values. ColdFusion uses this attribute only if you specify *range* in the `validate` attribute. If you specify a single number or a single number followed by a comma, it is treated as a minimum, with no maximum. If you specify a comma followed by a number, the maximum is set to the specified number, with no minimum.
required	Optional; All	`no`	• `yes`: the field must contain text. • `no`: the field can be empty.
richText	Optional; HTML	`no`	A Boolean value specifying whether this control is a rich text editor with tool bars to control the text formatting. For detailed information on using the rich text editor, see "Using the rich text editor" on page 641 in the *ColdFusion Developer's Guide*.
skin	Optional; HTML	`default`	Specifies the skin to be used for the rich text editor. By default, the valid values are `Default`, `silver`, and `office2003`. You can also create custom skins that you can then specify in this attribute. For more information, see "Using the rich text editor" on page 641 in the *ColdFusion Developer's Guide*.
sourceForTooltip	Optional; HTML		The URL of a page to display as a tool tip. The page can include CFML and HTML to control the contents and format, and the tip can include images. If you specify this attribute, an animated icon appears with the text "Loading..." while the tip is being loaded.

Attribute	Req/Opt; Format	Default	Description
style	Optional; All		In HTML or XML format forms, ColdFusion passes the style attribute to the browser or XML. In Flash format forms, must be a style specification in CSS format, with the same syntax and contents as used in Flex for the corresponding Flash element.
stylesXML	Optional; HTML	/CFIDE/scripts/ajax/FCKEditor/fckstyles.xml	The path of the file that defines the styles in the rich text editor Styles selector. Relative paths start at the directory that contains the fckeditor.html file, normally *cf_webRoot*/CFIDE/scripts/ajax/FCKeditor/editor. You can specify an absolute path starting at the web root, such as /myApps/RTEeditor.mystyles.xml. For information on configuring styles, see "Using the rich text editor" on page 641 in the *ColdFusion Developer's Guide*.
templatesXML	Optional; HTML	/CFIDE/scripts/ajax/FCKEditor/fcktemplates.xml	The pathof the file that defines the templates that are displayed when you click the rich text editor Templates icon. For pathing details, see stylesXML. For information on configuring templates, see "Using the rich text editor" on page 641 in the *ColdFusion Developer's Guide*.
toolbar	Optional; HTML	default	Specifies the rich text editor toolbar. By default, the valid values for this attribute are: Default, a complete set of controls, and Basic, a minimal configuration. You can add configurations; for details see "Using the rich text editor" on page 641 in the *ColdFusion Developer's Guide*. **Note**: This attribute's value is case sensitive.
toolbarOnFocus	Optional; HTML	no	A Boolean value that specifies whether the rich text editor toolbar expands and displays its controls only when the rich text editor has the focus.
tooltip	Optional; Flash, HTML		Text to display when the mouse pointer hovers over the control. Can include HTML formatting. Ignored if you specify a sourceForTooltip attribute.
validate	Optional; All		The type or types of validation to perform. Available validation types and algorithms depend on the format. For details, see the Usage section of the cfinput tag reference.
validateAt	Optional; HTML and XML	onSubmit	How to do the validation; one or more of the following: • onSubmit • onServer • onBlur For Flash format forms, onSubmit and onBlur are identical; for both, validation is done when the user submits the form. For multiple values, use a comma-delimited list. For details, see the Usage section of the cfinput tag reference.
value	Optional; All		Initial value to display in text control. You can specify an initial value as an attribute or in the tag body, but not in both places. If you specify the value as an attribute, you must put the closing cftextarea tag immediately after the opening cftextarea tag, with no spaces or line feeds between, or place a closing slash at the end of the opening cftextarea tag; for example <cftextarea name="description" value="Enter a description." />.

Attribute	Req/Opt; Format	Default	Description
visible	Optional; Flash	yes	Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank.
width	Optional; Flash, HTML		The width of the control, in pixels. In HTML forms, this attribute has an effect only if you specify richtext="true".
wrap	Optional All		• hard: wraps long lines, and sends the carriage return to the server. • off: does not wrap long lines. • physical: wraps long lines, and transmits the text at all wrap points. • soft: wraps long lines, but does not send the carriage return to the server. • virtual: wraps long lines, but does not send the carriage return to the server.

Usage

For this tag to work properly in HTML format. the browser must be JavaScript-enabled.

If you put text in the tag body, the control displays all text characters between the cftextarea opening and closing tags; therefore, if you use line feeds or white space to format your source text, they appear in the control.

If the cfform preserveData attribute is "yes", and the form posts back to the same page, the posted value (not the value of the value attribute) of the cftextinput control is used.

Validation

For a detailed description of the validation attribute and the types of validation supported by ColdFusion, see the Usage section of the cfinput tag reference. For more details on ColdFusion validation techniques, see "Validating Data" on page 554 in the *ColdFusion Developer's Guide*.

Flash form data binding

The bind attribute lets you populate form fields using the contents of other form fields. To specify text from another Flash form field in a cftextarea bind attribute, use the following format:

{*sourceTagName*.text}

For example, the following line uses the value of the text that the user enters in the from the userName field in the greeting in the comment text box. The user can change or replace this message with a typed entry.

```
<cfform format="flash" height="300">
    <cfformitem type="text">
        Enter your name here</cfformitem>
    <cftextarea name="userName" height="20" Width="500"/>
    <cftextarea name="comment" html height="100" Width="500"
        bind="Hello {userName.text}! Enter your comments here." />
</cfform>
```

HTML form data binding

The `bind` attribute lets you set any `cftextarea` attribute dynamically from the value of another form control or by calling a CFC or JavaScript function. By default it sets the control's `value` attribute, but you can specify a different attribute to set by using the `bindAttribute` attribute. For more information on binding, see "Binding data to form fields" on page 650 in the *ColdFusion Developer's Guide*.

Example

This example has two `cftextarea` controls. When you submit the form, ColdFusion copies the text from the first control into the second. The onBlur maxlength validation prevents you from entering more than 100 characters. The > character that closes the `cftextarea` opening tag, the text in the tag body, and the `cftextarea` closing tag are on a single line to ensure that only the desired text displays, but the line is split in this example for formatting purposes.

```
<h3>cftextarea Example</h3>
<cfparam name="text2" default="">
<cfif isdefined("form.textarea1") AND (form.textarea1 NEQ "")>
    <cfset text2=form.textarea1>
</cfif>

<cfform name="form1">
    <cftextarea name="textarea1" wrap="virtual" rows="5" cols="25"
    validate="maxlength" validateAt="onBlur" maxlength="100"
    >Replace this text. Maximum length is 100 Characters, and this text is
    currently 99 characters long.</cftextarea>
    <cftextarea name="textarea2" wrap="virtual" rows="5" cols="50" value="#text2#"
/><br><br>
    <input type="submit" value="submit field"><br>
</cfform>
```

cftextinput

Description

Puts a single-line text entry box in a `cfform` tag and controls its display characteristics.

This tag is deprecated, and is not supported in XML format forms. In its place, you should use a `cfinput` or `cftextarea` tag and use a cascading style sheet (CSS) to specify the text style characteristics.

History

ColdFusion MX 7: This tag is deprecated. In later releases it might not work, and might cause an error.

ColdFusion MX 6.1: Changed the `validate = "creditcard"` option requirements: the text entry must have 13-16 digits.

cfthread

Description

The `cfthread` tag enables you to create *threads*, independent streams of code execution, in your ColdFusion application. You use this tag to run or end a thread, temporarily stop thread execution, or join together multiple threads.

Category

Application framework tags

Syntax

```
join
    <cfthread
    required
    name="thread name[,thread name]..."
    optional
    action="join"
    timeout="milliseconds"/>

run
    <cfthread
    required
    name="thread name"
    optional
    action="run"
    priority="NORMAL|HIGH|LOW"
    zero or more application-specific attributes>

        Thread code

    </cfthread>

sleep
    <cfthread
    required
    action="sleep"
    duration="milliseconds"/>

terminate
    <cfthread
    required
    action="terminate"
    name="thread name"/>
```

For all actions except `run`, the `cfthread` tag must have an empty body and be followed immediately by a `</cfthread>` end tag, or must have no end tag and have a slash before the tag closure, as in `<cfthread action="sleep" duration="1000"/>`.

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`Sleep`, "Using ColdFusion Threads" on page 301 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag

Attributes

Attribute	Req/Opt	Default	Applies to	Description
action	Optional	run	All	The action to take, one of the following values: • join: Makes the current thread wait until the thread or threads specified in the name attribute complete processing, or until the period specified in the timeout attribute passes, before continuing processing. If you don't specify a timeout and the thread you are joining to doesn't finish, the current thread also cannot finish processing. • run: Creates a thread and starts it processing. Code in the cfthread tag body runs simultaneously and independently of page-level code and code in other cfthread tags. • sleep: Suspends the current thread's processing for the time specified by the duration attribute. Equivalent to the Sleep function. • terminate: Stops processing of the thread specified in the name attribute. If you terminate a thread, the thread scope includes an ERROR metadata structure with information about the termination.
duration	Required		sleep	The number of milliseconds for which to suspend thread processing.
name	Optional, Required , if action = "join" or "terminate"		join run terminate	The name of the thread to which the action applies: • terminate: The name of the thread to stop. • join: The name of the thread or threads to join to the current thread. To specify multiple threads, use a comma-delimited list. • run: The name to use to identify the thread being created.
priority	Optional	NORMAL	run	The priority level at which to run the thread. The following values are valid: • HIGH • LOW • NORMAL Higher priority threads get more processing time than lower priority threads. Page-level code, the code that is outside of cfthread tags, always has NORMAL priority.
timeout	Optional	0	join	The number of milliseconds that the current thread waits for the thread or threads being joined to finish. If any thread does not finish by the specified time, the current thread proceeds. If the attribute value is 0, the following action occurs: • The current thread continues waiting until all joining threads finish. • If the current thread is the page thread, the page continues waiting until the threads are joined, even if you specify a page time-out.

Usage

Page-level code (code outside any cfthread tags) executes in its own thread, referred to as the *page thread*. Only the page thread can create other threads. A thread that you create cannot create a child thread.

Note: *If a thread never completes processing (is hung), it continues to occupy system resources. You can use the ColdFusion Sever Monitor to check for and terminate hung threads.*

ColdFusion makes a complete (deep) copy of all the attribute variables before passing them to the thread, so the values of the variables inside the thread are independent of the values of any corresponding variables in other threads, including the page thread. Thus, the values passed to threads are thread safe because the attribute values cannot be changed by any other thread.

Thread scopes

Each thread has three special scopes:

- The thread-local scope is an implicit scope that contains variables that are available only to the thread, and exist only for the life of the thread.
- The Thread scope is available to the page and to all other threads started from the page. Its data remains available until the page and all threads started from the page finish, even if the page finishes before the threads complete processing.
- The Attributes scope contains attributes that are passed to the scope, and is available only within the thread and only for the life of the thread.

For detailed information about using ColdFusion scopes in threads, see "Using ColdFusion Threads" on page 301 in the *ColdFusion Developer's Guide*.

All threads in a page share a single Variables scope, so you can use it for data that is common across all threads. You must be careful to lock access to the variables, if necessary, to prevent deadlocks or race conditions between threads.

Note: *When ColdFusion uses a connector to access the web server, after the page gets completed, the CGI and Request scopes are not accessible to threads that you create by using the* `cfthread` *tag. This limitation does not apply if you use the integrated web server or if you run ColdFusion as a J2EE application.*

Metadata variables

The thread scope contains the following variables that provide information about the thread (metadata):

Variable	Description
ElapsedTime	The amount of processor time that was spent handling the thread.
Error	The structure that is generated if an error occurs during thread execution. The structure contains the keys that you can access in a `cfcatch` tag. If an error occurs in a thread, page-level processing is not affected, and ColdFusion does not generate an error message. The thread with the error terminates and the page-level code or other threads can get the error information from the Error field and handle the error appropriately. For detailed information, see "Handling ColdFusion thread errors" on page 309 in the *ColdFusion Developer's Guide*.
Name	The thread name.
Output	Output that is generated by the thread. A thread cannot display output; page-level code must use this variable to display thread results. For detailed information, see "Handling thread output" on page 309 in the *ColdFusion Developer's Guide*.

Variable	Description
Priority	The thread processing priority, as specified in the `cfthread priority` attribute. The following values are valid: • `HIGH` • `LOW` • *`NORMAL`*
Starttime	The time at which the thread began processing, in ColdFusion date-time format.
Status	The current status of the thread; one of the following values: • `NOT_STARTED`: The thread has been queued but is not processing yet. • `RUNNNG`: The thread is running normally. • `TERMINATED`: The thread stopped running due to a `cfthread` tag with a `terminate` action, an error, or an administrator action. • `COMPLETED`: The thread ended normally. • `WAITING`: The thread has executed a `cfthread` tag with `action="join"`, but one or more threads being joined has not completed.

Example

The following example uses three threads to get the results of three RSS feeds. The user must submit the form with all three feeds specified. The application joins the threads with a time-out of 6 seconds, and displays the feed titles and the individual item titles as links.

```
<!--- Run this code if the feed URL form has been submitted. --->
<cfif isDefined("Form.submit")>
    <cfloop index="i" from="1" to="3">
        <!--- Use array notation and string concatenation to create a variable
                 for this feed. --->
        <cfset theFeed = Form["Feed"&i]>
        <cfif theFeed NEQ "">
            <!--- Use a separate thread to get each of the feeds. --->
            <cfthread action="run" name="t#i#" feed="#theFeed#">
                <cffeed source="#feed#"
                    properties="thread.myProps"
                    query="thread.myQuery">
            </cfthread>
        <cfelse>
            <!--- If the user didn't fill all fields, show an error message. --->
            <h3>This example requires three feeds.<br />
            Click the Back button and try again.</h3>
            <cfabort>
        </cfif>
    </cfloop>

    <!--- Join the three threads. Use a 6 second time-out. --->
    <cfthread action="join" name="t1,t2,t3" timeout="6000" />

    <!--- Use a loop to display the results from the feeds. --->
    <cfloop index="i" from="1" to="3">
        <!--- Use the cfthread scope and associative array notation to get the
                Thread scope dynamically. --->
        <cfset feedResult=cfthread["t#i#"]>
        <!--- Display feed information only if you got items,
                 for example the feed must complete before the join. --->
        <cfif isDefined("feedResult.myQuery")>
```

```
            <cfoutput><h2>#feedResult.myProps.title#</h2></cfoutput>
            <cfoutput query="feedResult.myQuery">
                <p><a href="#RSSLINK#">#TITLE#</a></p>
            </cfoutput>
        </cfif>
    </cfloop>

</cfif>

<!--- The form for entering the feeds to aggregate. --->
<cfform>
    <h3>Enter three RSS Feeds</h3>
    <cfinput type="text" size="100" name="Feed1" validate="url"
        value="http://rss.adobe.com/events.rss?locale=en"><br />
    <cfinput type="text" size="100" name="Feed2" validate="url"
        value="http://weblogs.macromedia.com/dev_center/index.rdf"><br />
    <cfinput type="text" size="100" name="Feed3" validate="url"
        value="http://rss.adobe.com/studio.rss?locale=en"><br />
    <cfinput type="submit" name="submit">
</cfform>
```

cfthrow

Description

Throws a developer-specified exception, which can be caught with a `cfcatch` tag that has any of the following `type` attribute options:

- `type = "custom_type"`
- `type = "Application"`
- `type = "Any"`

Category

Exception handling tags, Flow-control tags

Syntax

```
<cfthrow
    detail = "detail description"
    errorCode = "error code"
    extendedInfo = "additional information"
    message = "message"
    object = "java except object"
    type = "exception type">

OR

<cfthrow
    object = #object_name#>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cferror`, `cfrethrow`, `cftry`, `onError`; "Handling Errors" on page 247 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Changed thrown exceptions: this tag can throw ColdFusion component method exceptions.

Attributes

Attribute	Req/Opt	Default	Description
`detail`	Optional		Description of the event. ColdFusion appends error position to description; server uses this parameter if an error is not caught by your code.
`errorCode`	Optional		A custom error code that you supply.
`extendedInfo`	Optional		A custom error code that you supply.
`message`	Optional		Message that describes exception event.

Attribute	Req/Opt	Default	Description
object	Optional		Requires the value of the cfobject tag name attribute. Throws a Java exception from a CFML tag. This attribute is mutually exclusive with all other attributes of this tag.
type	Optional	Application	• A custom type • Application Do not enter another predefined type; types are not generated by ColdFusion applications. If you specify Application, you need not specify a type for cfcatch.

Usage

Use this tag within a cftry block, to throw an error. The cfcatch block can access accompanying information, as follows:

- Message, with cfcatch.message
- Detail, with cfcatch.detail
- Error code, with cfcatch.errorcode

To get more information, use cfcatch.tagContext. This array shows where control switches from one page to another in the tag stack (for example, cfinclude, cfmodule).

To display the information displayed by tagContext: in the ColdFusion Administrator Debugging page, select Enable CFML Stack Trace.

To use this tag with the object parameter, you must first use a cfobject tag that specifies a valid Java exception class. For example, the following cfobject tag defines an object, obj, of the exception class myException (which you must create in Java):

```
<cfobject
    type="java"
    action="create"
    class="myException"
    name="obj">
```

If your exception class has constructors that take parameters, such as a message, you can use the special init method to invoke the constructor, as in the following line. If you do not need to specify any constructor attributes, you can omit this step.

```
<cfset obj.init("You must save your work before preceding")>
```

You can then use the, the cfthrow statement to throw the exception as follows:

```
<cfthrow object=#obj#>
```

For more information on using Java objects in ColdFusion, see "Integrating J2EE and Java Elements in CFML Applications" on page 929 in the *ColdFusion Developer's Guide*.

Example

```
<h3>cfthrow Example</h3>
<!--- Open a cftry block. --->
<cftry>
<!--- Define a condition upon which to throw the error. --->
<cfif NOT IsDefined("URL.myID")>
    <!--- throw the error --->
    <cfthrow message = "ID is not defined">
</cfif>
```

```
<!--- Perform the error catch. --->
<cfcatch type = "application">
<!--- Display your message. --->
    <h3>You've Thrown an <b>Error</b></h3>
<cfoutput>
    <!--- And the diagnostic feedback from the application server. --->
<p>#cfcatch.message#</p>
    <p>The contents of the tag stack are:</p>
    <cfloop
        index = i
        from = 1 to = #ArrayLen(cfcatch.tagContext)#>
            <cfset sCurrent = #cfcatch.tagContext[i]#>
                <br>#i# #sCurrent["ID"]#
                (#sCurrent["LINE"]#,#sCurrent["COLUMN"]#)
                #sCurrent["TEMPLATE"]#
    </cfloop>
</cfoutput>
</cfcatch>
</cftry>
```

The following example shows how to throw an exception from a component method:

```
<cfcomponent>
    <cffunction name="getEmp">
    <cfargument name="lastName" required="yes">
         <cfquery name="empQuery" datasource="cfdocexamples" >
            SELECT LASTNAME, FIRSTNAME, EMAIL
            FROM tblEmployees
            WHERE LASTNAME LIKE '#arguments.lastName#'
         </cfquery>
         <cfif empQuery.recordcount LT 1>
            <cfthrow type="noQueryResult"
                message="No results were found. Please try again.">
            <cfelse>
                <cfreturn empQuery>
        </cfif>
    </cffunction>
</cfcomponent>
```

For an explanation of the example and more information, see “Building and Using ColdFusion Components” on page 158 in the *ColdFusion Developer’s Guide*.

cftimer

Description

Displays execution time for a specified section of CFML code. ColdFusion MX displays the timing information along with any output produced by the timed code.

Note: *To permit this tag to execute, you must enable the Enable Debugging and the Timer Information options on the Debugging Settings page in the ColdFusion Administrator.*

Category

Debugging tags

Syntax

```
<cftimer
    label= "text"
    type = "inline|outline|comment|debug" >

    CFML statement(s)

</cftimer>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfdump`, `cftrace`; "Debugging and Troubleshooting Applications" on page 352 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`label`	Optional	`" "`	Label to display with timing information.
`type`	Optional	`debug`	• `inline`: displays timing information inline, following the resulting HTML. • `outline`: displays timing information and also displays a line around the output produced by the timed code. The browser must support the FIELDSET tag to display the outline. • `comment`: displays timing information in an HTML comment in the format <!-- *label*: *elapsed-time* ms -->. The default label is `cftimer`. • `debug`: displays timing information in the debug output under the heading CFTimer Times.

Usage

Use this tag to determine how long it takes for a block of code to execute. You can nest `cftimer` tags.

This tag is useful for debugging CFML code during application development. In production, you can leave `cftimer` tags in your code as long as you have disabled the debugging option in the ColdFusion Administrator.

Example

```
...
<!--- type="inline"> --->
```

```
 <cftimer label="Query and Loop Time Inline" type="inline">
        <cfquery name="empquery" datasource="cfdocexamples">
            SELECT *
            FROM Employees
        </cfquery>

        <cfloop query="empquery">
        <cfoutput>#lastname#, #firstname#</cfoutput><br>
        </cfloop>
    </cftimer>
<hr><br>

<!--- type="outline" --->
 <cftimer label="Query and CFOUTPUT Time with Outline" type="outline">
        <cfquery name="coursequery" datasource="cfdocexamples">
            SELECT *
            FROM CourseList
        </cfquery>
    <table border="1" width="100%">
        <cfoutput query="coursequery">
        <tr>
        <td>#Course_ID#</td>
        <td>#CorName#</td>
        <td>#CorLevel#</td>
        </tr>
        </cfoutput>
        </table>
    </cftimer>
<hr><br>

<!--- type="comment" --->
 <cftimer label="Query and CFOUTPUT Time in Comment" type="comment">
        <cfquery name="parkquery" datasource="cfdocexamples">
                SELECT *
                FROM Parks
        </cfquery>
<p>Select View &gt; Source to see timing information</p>
    <table border="1" width="100%">
        <cfoutput query="parkquery">
        <tr>
        <td>#Parkname#</td>
        </tr>
        </cfoutput>
        </table>
    </cftimer>
<hr><br>

<!--- type="debug" --->
 <cftimer label="Query and CFOUTPUT Time in Debug Output" type="debug">
        <cfquery name="deptquery" datasource="cfdocexamples">
                SELECT *
                FROM Departments
        </cfquery>
<p>Scroll down to CFTimer Times heading to see timing information</p>
    <table border="1" width="100%">
        <cfoutput query="deptquery">
        <tr>
        <td>#Dept_ID#</td>
        <td>#Dept_Name#</td>
        </tr>
        </cfoutput>
```

```
        </table>
    </cftimer>
...
```

cftooltip

Description

Specifies tool tip text that displays when the user hovers the mouse pointer over the elements in the tag body. This tag does not require a form and is not used inside Flash forms.

Category

Display management tags

Syntax

```
<cftooltip
    autoDismissDelay="5000"
    hideDelay="250"
    preventOverlap="true|false"
    showDelay="200"
    sourceForTooltip="URL"
    tooltip="text">

    Display tags

</cftooltip>
```

This tag must have an end tag.

Note: *You can specify this tag's attribute in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute name as structure key.*

See also

`cfajaximport`, "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag

Attributes

Attribute	Req/Opt	Default	Description
`autoDismissDelay`	Optional	5000	The number of milliseconds between the time when the user moves the mouse pointer over the component (and leaves it there) and when the tool tip disappears.
`hideDelay`	Optional	250	The number of milliseconds to delay between the time when the user moves the mouse pointer away from the component and when the tool tip disappears.
`preventOverlap`	Optional	`true`	A Boolean value specifying whether to prevent the tool tip from overlapping the component that it describes.
`showDelay`	Optional	200	The number of milliseconds to delay between the time when the user moves the mouse over the component and when the tool tip appears.
`sourceForTooltip`	Optional		The URL of a page with the tool tip contents. The page can include HTML markup to control the format, and the tip can include images. If you specify this attribute, an animated icon appears with the text "Loading..." while the tip is being loaded.
`tooltip`	Optional		Tip text to display. The text can include HTML formatting. Ignored if you specify a `sourceForTooltip` attribute.

Usage

Specify a `tooltip` or a `sourceForTooltip` attribute; otherwise, this tag has no effect.

If you specify the path to a CFML page in the `sourceForTooltip` attribute, ColdFusion processes the page and uses its output in the tip text. You can therefore use CFML programming, in addition to HTML formatting, to control the contents and appearance of the tip text.

You should use the `cftooltip` tag for text and simple components, such as images, not for complex Ajax components such as windows, pods, or layout areas. If you use the `cftooltip` tag with complex components, you might get unexpected behavior; for example, the tool tip might overlap window contents, even if you specify the `preventoverlap` attribute.

You can nest tool tips, although this may result in multiple tool tips obscuring one another. **Example**

The following simple example can dynamically display different tool-tip text based on the value of the `theItem` variable on the main CFML page.

The main CFML page:

```
<!--- These variables could be set dynamically --->
<cfset theItem="left-handed & other specialty wrenches">
<cfset theImage="lhbwrench.jpg">

<!--- The theItem string has an ampersand, so you must URL-encode it. --->
<cftooltip sourceForTooltip="tiptext.cfm?itemid=#URLEncodedFormat(theItem)#">
    <cfoutput>
        <b>Try this one!</b>
        <img src="#theImage#" />
    </cfoutput>
</cftooltip>
```

The tiptext.cfm page could have a single CFML tag:

```
<cfoutput><b> Click to find more about #URL.itemid# </b></cfoutput>
```

cftrace

Description

Displays and logs debugging data about the state of an application at the time the `cftrace` tag executes. Tracks run-time logic flow, variable values, and execution time. Displays output at the end of the request or in the debugging section at the end of the request; or, in Dreamweaver MX and later, in the Server Debug tab of the Results window.

ColdFusion logs `cftrace` output to the file logs\cftrace.log, in the ColdFusion installation directory.

Note: *To permit this tag to execute, you must enable debugging in the ColdFusion Administrator. Optionally, to report trace summaries, enable the Trace section*

Category

Debugging tags, Variable manipulation tags

Syntax

```
<cftrace
    abort = "yes|no"
    category = "string"
    inline = "yes|no"
    text = "string"
    type = "format"
    var = "variable name">
</cftrace>
```

Note: *You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.*

See also

`cfdump`, `cferror`, `cfrethrow`, `cftimer`, `cftry`; "Debugging and Troubleshooting Applications" on page 352 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`abort`	Optional	`no`	• `yes`: calls a `cfabort` tag when the tag is executed. • `no`
`category`	Optional		User-defined string that identifies trace groups.
`inline`	Optional	`no`	• `yes`: displays trace code inline on the page in the location of the `cftrace` tag, in addition to the debugging information output. • `no`
`text`	Optional		User-defined string, which can include simple variables, but not complex variables such as arrays. Outputs to the `cflog text` attribute.

Attribute	Req/Opt	Default	Description
type	Optional	Information	Corresponds to the cflog type attribute; displays an appropriate icon: • Information • Warning • Error • Fatal Information
var	Optional		The name of a simple or complex variable to display. Useful for displaying a temporary value, or a value that does not display on any CFM page.

Usage

You cannot put application code within this tag. (This avoids problems that can occur if you disable debugging.)

This tag is useful for debugging CFML code during application development.

You can display cftrace tag output in the following ways:

- As a section in the debugging output
- Inline in an application page, and as a section in debugging output. If you specify inline tracing, ColdFusion flushes all output up to the cftrace tag, and displays the trace output when it encounters the tag.

This is an example of a log file entry:

```
"Information","web-4","04/08/02","23:21:30", ,"[30 ms (1st trace)]
[C:\CFusion\wwwroot\generic.cfm @ line: 9] -
    [thisPage = /generic.cfm]"
"Information","web-0","04/08/02","23:58:58", ,"[5187 ms (10)]
[C:\CFusion\wwwroot\generic.cfm @ line: 14] - [category]
    [thisPage = /generic.cfm] [ABORTED] thisPage "
```

For a complex variable, ColdFusion lists the variable name and the number of elements in the object; it does not log the contents of the variable.

Example

The following example traces a FORM variable that is evaluated by a cfif block:

```
<cftrace var="FORM.variable"
    text="doing equivalency check for FORM.variable"
    category="form_vars"
    inline="true">
<cfif isDefined("FORM.variable") AND #FORM.variable# EQ 1>
    <h1>Congratulations, you're a winner!</h1>
<cfelse>
    <h1>Sorry, you lost!</h1>
</cfif>
```

cftransaction

Description

For enterprise database management systems that support transaction processing, instructs the database management system to treat multiple database operations as a single transaction. Provides database commit and rollback processing. See the documentation for your database management system to determine whether it supports SQL transaction processing.

Category

Database manipulation tags

Syntax

```
<cftransaction
    action = "begin|commit|rollback|setsavepoint"
    isolation = "read_uncommitted|read_committed|repeatable_read"
    savepoint = "savepoint name">
</cftransaction>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfinsert`, `cfprocparam`, `cfprocresult`, `cfquery`, `cfqueryparam`, `cfstoredproc`, `cfupdate`; "Commits, rollbacks, and transactions" on page 382 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `setsavepoint` value to the `action` attribute. Added the `savepoint` attribute.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Optional	`begin`	• `begin`: The start of the block of code to execute. • `commit`: Commits a pending transaction. • `rollback`: Rolls back a pending transaction. • `setsavepoint`: Saves a specific state within a transaction
`isolation`	Optional		Isolation level, which indicates which type of read can occur during the execution of concurrent SQL transactions. The possible read actions include *dirty read*, in which a second SQL transaction reads a row before the first SQL transaction executes a COMMIT; *non-repeatable read*, in which a SQL transaction reads a row and then a second SQL transaction modifies or deletes the row and executes a COMMIT; and *phantom*, in which a SQL transaction reads rows that meet search criteria, a second SQL transaction then generates at least one row that meets the first transaction's search criteria, and then the first transaction repeats the search, resulting in a different result set. • `read_uncommitted`: Allows dirty read, non-repeatable read, and phantom • `read_committed`: Allows non-repeatable read and phantom. Does not allow dirty read. • `repeatable_read`: Allows phantom. Does not allow dirty read or non-repeatable read. • `serializable`: Does not allow dirty read, non-repeatable read, or phantom.
`savepoint`	Optional		The name of the savepoint in the transaction. Setting savepoints lets you roll back portions of a transaction. For example, if your transaction includes an insert, an update, and a delete, and you set a savepoint after the update, you can roll back the transaction to exclude the delete.

Usage

If you do not specify a value for the `action` attribute, automatic transaction processing proceeds as follows:

- If the `cfquery` operations within the transaction block complete without an error, the transaction is committed.
- If a `cfquery` tag generates an error within a `cftransaction` block, all `cfquery` operations in the transaction roll back.

If you do not specify a value for the `isolation` attribute, ColdFusion uses the default isolation level for the associated database.

By using CFML error handling and the `action` attribute, however, you can explicitly control whether a transaction is committed or rolled back, based on the success or failure of the database query. In a transaction block, you can do the following:

- Commit a database transaction by nesting the `<cftransaction action = "commit"/>` tag in the block.
- Roll back a transaction by nesting the `<cftransaction action = "rollback"/>` tag in the block.

(In these examples, the slash is an alternate syntax that is the equivalent of an end tag.)

In a transaction block, you can write queries to more than one database, but you must commit or roll back a transaction to one database before writing a query to another.

To control how the database engine performs locking during the transaction, use the `isolation` attribute.

The `cftransaction` tag does not work as expected if you use the `cfthread` tag in it to make query calls.

Example

```
<p>The cftransaction tag can be used to group multiple queries that use
the cfquery tag into one business event. Changes to data that is requested
by the queries are not committed to the datasource until all actions within
the transaction block have executed successfully.
<p>This a view-only example.
<!---
<cftransaction>
    <cfquery name='makeNewCourse' datasource='Snippets'>
    INSERT INTO Courses
        (Number, Descript)
    VALUES
        ('#myNumber#', '#myDescription#')
    </cfquery>

    <cfquery name='insertNewCourseToList' datasource='Snippets'>
    INSERT INTO CourseList
        (CorNumber, CorDesc, Dept_ID,
        CorName, CorLevel, LastUpdate)
    VALUES
        ('#myNumber#', '#myDescription#', '#myDepartment#',
        '#myDescription#', '#myCorLevel#', #Now()#)
    </cfquery>
</cftransaction>
--->
```

You can set savepoints at the completion of insert, update, and delete actions of a transaction. You then use error handling logic to determine whether it is necessary to roll back to a previous savepoint.

Example

```
<!--- This example performs batch processing of withdrawals --->
<!--- from a bank account. The withdrawal amounts are stored --->
```

```
<!--- in an array. --->
<!--- There is a CFC named bank.cfc whose contains appear --->
<!--- after the example. --->

<cftransaction>
    <!--- Get the account balance. --->
    <cfinvoke component="bank" method="getBalance"
        returnvariable="getacctbalance" accountnum=1>

<cfloop index="withdrawnum" from="1" to="#ArrayLen(withdrawals)#">
    <!--- Set a savepoint before making the withdrawal. --->
    <cfset noxfer = "point" & #withdrawnum#>
    <cftransaction action = "setsavepoint" savepoint = "#noxfer#"/>

    <!--- Make the withdrawal. --->
    <cfinvoke component="bank" method="makewithdrawal"
        returnvariable="getacctbalance" accountnum=1
        withdrawamount="#withdrawals[withdrawnum]#">

    <!--- Get the account balance. --->
    <cfinvoke component="bank" method="getBalance"
        returnvariable="getacctbalance" accountnum=1>

    <!--- If the balance is negative, roll back the transaction. --->
    <cfif getacctbalance.balance lt 0>
        <cftransaction action="rollback" savepoint="#noxfer#" />
    </cfif>
    </cfloop>
</cftransaction>

<!--- The bank.cfc contains the following:

cfcomponent>
    <cffunction name="getBalance" access="public" returntype="query">
        <cfargument name="accountnum" type="numeric" required="yes">
            <cfquery name="getacctbalance" datasource="testsqlserver">
                SELECT * FROM dbo.mybank
                WHERE accountid = #accountnum#
            </cfquery>
        <cfreturn getacctbalance>
    </cffunction>

    <cffunction name="makewithdrawal" access="public" returntype="query">
        <cfargument name="accountnum" type="numeric" required="yes">
        <cfargument name="withdrawamount" type="numeric" required="yes">
            <cfquery name="withdrawfromacct" datasource="testsqlserver">
                UPDATE dbo.mybank SET balance = balance - #withdrawamount#
                WHERE accountid = 1
            </cfquery>
            <cfinvoke method="getBalance" returnvariable="getacctbalance"
        accountnum=1>
        <cfreturn getacctbalance>
    </cffunction>
</cfcomponent>
--->
```

cftree

Description

Inserts a tree control in a form. Validates user selections. Used within a cfform tag block. Use a ColdFusion query to supply data to the tree.

Category

Forms tags

Syntax

```
<cftree
    name="name"
    align="top|left|bottom|baseline|texttop|absbottom|
        middle|absmiddle|right"
    appendKey="yes|no"
    bold="yes|no"
    border="yes|no"
    cache="yes|no"
    completePath="yes|no"
    delimiter="delimiter"
    enabled="yes|no"
    font="font"
    fontSize="size"
    format="applet|flash|html|object|xml"
    height="integer"
    highlightHref="yes|no"
    hScroll="yes|no"
    hSpace="integer"
    italic="yes|no"
    lookAndFeel="motif|windows|metal"
    message="text"
    notSupported="text">
    onBlur="ActionScript to invoke"
    onChange="ActionScript to invoke"
    onError="text"
    onFocus="Actionscript to invoke"
    onValidate="script name"
    required="yes|no"
    style= "style specification"
    tooltip="text"
    visible="yes|no"
    vScroll="yes|no"
    vSpace="integer"
    width="integer">
</cftree>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfajaximport, cfapplet, cfcalendar, cfform, cfformgroup, cfformitem, cfgrid, cfinput, cfselect, cfslider, cftextarea, cftreeitem; "Working with action pages" on page 515,"Building tree controls with the cftree tag" on page 533,and "Using HTML format trees" on page 636 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added support for Ajax based HTML trees, including the `cache` attribute and the `html` value for `format` attribute.

ColdFusion MX7.01: Added support for onBlur and onFocus events.

ColdFusion MX 7:

- Added the `format` attribute and support for generating Flash and XML and object output.
- Added `enabled`, `onChange`, `style`, `tooltip`, and `visible` attributes (Flash format only).

ColdFusion MX: Changed behavior: ColdFusion renders a tree control regardless of whether there are any `treeitems` within it.

Attributes

Note: *In XML format, ColdFusion passes all attributes to the XML. The supplied XSLT skins do not handle or display XML format trees, but do display applet and Flash format trees.*

Attribute	Req/Opt Format	Default	Description
`name`	Required; All		Name for a tree control.
`align`	Optional; Applet, object		• `top` • `left` • `bottom` • `baseline` • `texttop` • `absbottom` • `middle` • `absmiddle` • `right`
`appendKey`	Optional; All	`yes`	• `yes`: if you use `cftreeitem href` attributes, ColdFusion appends a CFTREEITEMKEY query string variable with the value of the selected tree item to the `cfform` action URL. • `no`: does not append the tree item value to the URL.
`bold`	Optional; Applet, Flash, HTML	`no`	• `yes`: displays tree control text in bold. • `no`
`border`	Optional; Applet, object	`yes`	• `yes`: displays a border around the tree control. • `no`
`cache`	Optional; HTML	`yes`	Applies only if the tree's child `treeitem` tag uses a bind expression. A Boolean value that specifies whether to get new data each time the user expands tree nodes, as follows: • `yes`: fetches a node's child items only once, when the node is first expanded • `no`: fetches child items each time the node is expanded.

Attribute	Req/Opt Format	Default	Description
completePath	Optional; Applet, HTML, object	no	• yes: starts the Form.*treename*.path variable with the root of the tree path when cftree is submitted. • no: omits the root level from the Form.*treename*.path variable; the value starts with the first child node in the tree. For the preserveData attribute of cfform to work with the tree, you must set this attribute to yes. For tree items populated by a query, if you use the cftreeitem queryasroot attribute to specify a root name, that value is returned. If you do not specify a root name, ColdFusion returns the query name.
delimiter	Optional; All	\\	Character to separate elements in the Forms.*treename*.path variable of the action page.
enabled	Optional; Flash	yes	Flash format only: Boolean value that specifies whether the control is enabled. A disabled control appears in light gray.
font	Optional; Applet, HTML		Font name for text in the tree control.
fontSize	Optional; Applet, Flash, HTML		Font size for text in the tree control, in pixels.
format	Optional; All	applet	• applet: displays the tree using a Java applet in the browser. • flash: displays the tree using a Flash control • html: displays the tree uses Ajax-based HTML • object: returns the tree as a ColdFusion structure with the name specified by the name attribute, For details of the structure contents, see object format. • xml: generates an XML representation of the tree. In XML format forms, includes the generated XML in the form. In HTML format forms, puts the XML in a string variable with the name specified by the name attribute.
height	Optional; Applet, Flash	320 (applet only)	Tree control height, in pixels. If you omit this attribute in Flash format, Flash automatically sizes the tree.
highlightHref	Optional; Applet, Object	yes	• yes: highlights as a link the displayed value for any cftreeitem tag that specifies an href attribute. • no: disables highlighting.
hScroll	Optional; Applet, object	yes	• yes: permits horizontal scrolling. • no
hSpace	Optional; Applet		Horizontal spacing to left and right of tree control, in pixels.
italic	Optional; Applet, Flash, HTML	no	• yes: displays tree control text in italics. • no

Attribute	Req/Opt Format	Default	Description
lookAndFeel	Optional; Applet, object	windows	• motif: renders the tree in Motif style. • windows: renders the tree in Windows style. • metal: renders the tree in Java Swing style. If the platform does not support a style option, the tag uses the default style for the platform.
message	Optional; Applet, HTML		Message to display if validation fails.
notSupported	Optional; Applet	See Description	Text to display if a page that contains a Java applet-based cfform control is opened by a browser that does not support Java or has Java support disabled, for example; " Browser must support Java to view ColdFusion Java Applets" Default message: Browser must support Java to view ColdFusion Java Applets!
onBlur	Optional		ActionScript to run when the tree loses focus.
onChange	Optional; Flash		ActionScript to run when the control changes due to user action. If you specify an onChange event handler, the Form scope of the ColdFusion action page does not automatically get information about selected items. The ActionScript onChange event handler must handle all changes and selections.
onError	Optional; Applet, HTML		A JavaScript function to run if validation fails.
onFocus	Optional; Flash		ActionScript to run when the tree gets focus. The JavaScript DOM form object, value of the name attribute, value that failed validation, and any error text specified by the message attribute are passed to the method.
onValidate	Optional; Applet, HTML		JavaScript function to validate user input. The JavaScript DOM form object, input object, and input object value are passed to the specified routine, which should return true if validation succeeds; false, otherwise.
required	Optional; Applet, Flash, HTML	no	• yes: users must select an item in the tree control. • no
style	Optional; Flash, HTML		Must be a style specification in CSS format. In HTML format, this attribute corresponds to the value of an HTML style attribute. In Flash format, use the same syntax and contents as used in Flex for the corresponding Flash element.
tooltip	Optional; Flash		Flash format only: Text to display when the mouse pointer hovers over the control.
visible	Optional; Flash	yes	Flash format only: Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank.
vScroll	Optional; Applet, object	yes	• yes: permits vertical scrolling. • no

Attribute	Req/Opt	Default	Description
	Format		
vSpace	Optional; Applet		Vertical margin above and below tree control, in pixels.
width	Optional; Applet, Flash	200 (applet only)	Tree control width, in pixels. If you omit this attribute in Flash format, Flash automatically sizes the tree.

Note: *All attributes are passed to the XML generated in XML format, but no ColdFusion skin interprets* `cftree` *XML.*

Usage

This tag must be in a `cfform` tag block.

The applet format tree requires the client to download a Java applet. Also, if the client does not have an up-to-date Java plug-in installed, the system might also have to download an updated Java plug-in to display an applet format tree. The Flash format tree uses a Flash control, and can be embedded in an HTML format `cfform` tag. For this tag to work properly in Flash, HTML, or applet format, the browser must also be JavaScript-enabled.

Note: *If you specify Flash format for this tag in an HTML format form, and you do not specify* `height` *and* `width` *attributes, Flash takes up more than the remaining visible area on the screen. If any other output follows the tree, including any form controls, users must scroll to see it. Therefore, if you follow a Flash tree in an HTML form with additional output, specify* `height` *and* `width` *values.*

If the following conditions are true, a user's selection from query data that populates this tag's options continues to display after the user submits the form:

- The `cfform preserveData` attribute is set to `"yes"`
- The `cfform action` attribute posts to the same page as the form itself (this is the default), or the action page has a form that contains controls with the same names as corresponding controls on the user entry form

For more information, see the `cfform` tag entry.

Form variables

When you select a tree item and submit the form that contains the tree, ColdFusion creates a structure with two variables in the action page Form scope. The structure name is the tree name. The following table lists the fields:

Field	Value
path	The path through the tree to the selected node, in the form [*root*\]*node_1**node_2*\.... In applet format, the path includes the root node only if the `completePath` attribute is `true`. In Flash format, the path always includes the root node.
node	The value of the selected tree node.

object format

If you specify `object` in the `format` attribute, ColdFusion returns the tree as a ColdFusion structure, and does not send the tree to the browser. You can, for example, loop over the structure to populate a menu, generate "bread-crumb" links for page navigation, or create a DHTML tree.

Note: *If you specify an object format tree in an XML format form, ColdFusion does not generate the tree.*

The structure variable name is specified by the `cftree name` attribute. The top level of the structure has two types of entries:

- Attribute settings

- A children array

Attribute settings

The structure has top-level entries with the values of the following `cftree` attributes:

`align`	`completePath`	`highlightHref`	`lookAndFeel`
`appendKey`	`delimiter`	`hScroll`	`name`
`bold`	`fontWeight`	`italic`	`vscroll`
`border`			

Children array

The top-level children entry is an array of items entries. Each item has the following entries:

Field	Value
children	This item's child items; an array of item structures.
display	Tree item label, as determined by the `cftreeitem display` attribute.
expand	Whether to expand the item to display any children; value of `cftreeitem expand` attribute.
href	The URL to link to when the user selects the item; value of the `cftreeitem href` attribute.
img	The tree image icon Image to display as an icon for the tree item; value of `cftreeitem img` attribute. You can use the `img` attribute to display custom icons only in the Applet version; not in the Flash version.
imgOpen	Image to display when the tree item is open (expanded); value of `cftreeitem imgopen` attribute.
parent	Value of this item's parent item in the tree.
path	The node path from the tree root to the current element.
queryAsRoot	Whether the query is the root of the item; value of `cftreeitem queryAsRoot` attribute.
target	The link target, such as _blank; value of the item's `cftreeitem target` attribute.
value	The item's value, as determined by the `cftreeitem value` attribute.

Example

The following example creates a tree that shows available courses from the CourseList table of the cfdocexamples database, and puts each department's courses in a folder. This example is displayed in Flash and uses the Departments list to get department names.

```
<cfquery name="getCourses" datasource="cfdocexamples">
    SELECT d.dept_name, c.course_id, c.CorName, c.CorLevel, c.corName ||' ( ' ||c.corLevel
||' )' AS corLabel
    FROM CourseList c, Departments d
    WHERE d.Dept_ID = c.Dept_ID
    ORDER BY d.dept_Name, c.corName, c.corLevel
</cfquery>

<cfform name="studentForm" format="flash" width="400" height="450">
    <cftree name="courseTree" width="350" height="400">
        <cftreeitem
            query="getCourses"
            value="dept_name,Course_id"
            display="dept_name,CorLabel" queryasroot="NO" expand="yes,no">
    </cftree>
</cfform>
```

The following example creates a tree that shows the basic information about all employees in an organization, organized by department. The departments are expanded to show all employees. You click the + signs to display additional information. If you click the employee name, ColdFusion links back to the same page and displays the Path and node values for the selection.

```
<!--- Query the datasource to get employee information. --->
<!--- Group the output by Department.
    (All fields are required in Group By clause.) --->
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
    SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
    FROM Employees
    GROUP BY Department, Emp_ID, FirstName, LastName, EMail, Phone
</cfquery>
<html>
<body>
<h3>cftree Example</h3>

<!--- The following runs if the user clicked on a link in the tree.
    A complete application would use the ID for additional processing. --->

<cfif isdefined("Form.fieldnames")>
<b>Selected item information</b><br>
<cfoutput>
<b>Path: </b>#form.Employees.Path#<br>
<b>node: </b>#form.Employees.node#<br>
<br>
</cfoutput>
</cfif>

<!--- Display the tree. The cftree tag must be in a cfform. --->
<cfform action="#cgi.script_name#" preservedata="Yes" format="Flash">
    <cftree name = "Employees" height = "400" width = "400"
        font = "Arial Narrow" italic="yes" highlighthref="No" HScroll="no" VScroll="no"
        completepath="no" lookandfeel="windows" border="No" required="yes">
        <!--- cfoutput tag with a group attribute loops over the departments. --->
        <cfoutput group="Department" query = "GetEmployees">
            <cftreeitem value="#Department#" parent="Employees" expand="yes">
            <!--- This cfoutput tag loops over the records for the department.
                    The cfoutput tag does not need any attributes. --->
            <cfoutput>
                <!--- Create an item for each employee in the department.
                    Do not expand children. Each employee name links to this page,
                    and sends the employee ID in the query string.--->
                <cftreeitem value = "#LastName#, #FirstName#"
                    parent = "#Department#" expand="false" img="cd"
                    href="#cgi.script_name#?user_id=#emp_id#">
                <!--- Each Employee entry has Id, and contact info children. --->
                <cftreeitem value = "#Emp_ID#" display = "Employee ID: #Emp_ID#"
                    parent = "#LastName#, #FirstName#" img="remote">
                <!--- Each node must be unique value, so use Emp_ID om value. --->
                <cftreeitem value = "#Emp_ID#_ContactInfo" img="computer"
                    display = "Contact Information"
                    parent = "#LastName#, #FirstName#" expand = "false">
                <!--- ContacInfo has two children --->
                <cftreeitem value = "#Phone#" parent = "#Emp_ID#_ContactInfo">
                <cftreeitem value = "#Email#" parent = "#Emp_ID#_ContactInfo">
            </cfoutput>
        </cfoutput>
    </cftree>
    <cfinput type="Submit" name="submitit" value="Submit" width="100">
```

```
</cfform>
```

cftreeitem

Description

Populates a form tree control, created with the `cftree` tag, with one or more elements.

Category

Forms tags

Syntax

```
<cftreeitem
    value = "text"
    bind = "bind expression"
    display = "text"
    expand = "yes|no"
    href = "URL"
    img = "filename"
    imgopen = "filename"
    parent = "parent name"
    query = "queryname"
    queryAsRoot = "yes|no"
    target = "URL target">

OR

<cftreeitem
    bind = "bind expression">
    onBindError = "JavaScript function name"
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

History

ColdFusion 8: Added the `bind` and `onBindError` attributes.

See also

`cfapplet`, `cfform`, `cfformgroup`, `cfformitem`, `cfgrid`, `cfinput`, `cfselect`, `cfslider`, `cftextarea`, `cftree`; "Building tree controls with the cftree tag" on page 533 and "Using HTML format trees" on page 636 in the *ColdFusion Developer's Guide*

Attributes

Note: *In XML format, ColdFusion passes all attributes to the XML. The supplied XSLT skins do not handle or display XML format trees, but do display applet and Flash format trees.*

Attribute	Req/Opt; Format	Default	Description
value	Required for applet, Flash, XML. value or bind is required for HTML.		Value passed when the form containing the tree is submitted. When populating a tree with data from a cfquery, you can specify multiple columns to use in a delimited list; for example, value = "dept_id,emp_id". In this case, each column generates an item that is a child of the column that precedes it in the list.
bind	value or bind is required for HTML		A bind expression specifying a CFC function, JavaScript function, or URL that dynamically gets all tree nodes. You can use this attribute only at the top level of the tree, and in this case, the tree can have only cftreeitem tag. If you use the bind attribute, the only other allowed attribute is onBindError. For details creating trees that using binding, see "Using HTML format trees" on page 636 in the *ColdFusion Developer's Guide*
display	Optional; All	value	Tree item label. When populating a tree with data from a query, specify names in a delimited list, for example: `display = "dept_name,emp_name"`
expand	Optional; All	yes	• yes: expands tree to show tree item children. • no: keeps tree item collapsed.
href	Optional; All		URL to link to if the user clicks the tree item. If you use a query attribute, the href attribute can specify a query column that contains URLs. If href is not a query column, the attribute text must be a URL or list of URLs. When populating a tree with data from a query, specify the URLs in a comma-delimited list, for example: `href = "http://dept_svr,http://emp_svr"`

img	Optional; Applet, HTML, object	folder	Image name, filename, or file URL for tree item icon.
			The following values are provided:
			• cd
			• computer
			• document
			• element
			• folder
			• floppy
			• fixed
			• remote
			You can also specify a custom image. To do so, include path and file extension; for example:
			`img = "../images/page1.gif"`
			You can also specify a path relative to the web root.
			Custom images are not supported for Flash format.
			To specify more than one image in a tree, or an image at the second or subsequent level, use commas to separate names, corresponding to level; for example:
			`img = "folder,document"`
			`img = ",document" (example of second level)`

Attribute	Req/Opt; Format	Default	Description
imgopen	Optional; Applet, HTML, object		Icon displayed with open tree item, as described for the img attribute.
onBindError	Optional; HTML	see Description	The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and you specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up appears.
parent	Optional; All		Value of the tree item parent. Determines the item's placement in the tree hierarchy. If omitted, the item is placed at the tree root level, or if the queryAsRoot attribute is True, directly under the query.
query	Optional; All		Query name to use to populate the treeitem. ColdFusion generates an item for each field value in the query column list specified by the value attribute. The fields in each row are hierarchically linked to the first column.
queryAsRoot	Optional; All	yes	Applies only if you specify a query attribute. Defines the query as the root level for all items generated by this tag. This attribute lets you avoid creating a parent cftreeitem. • yes: generates a parent (root) item for all other items generated by the tag, with the query name as its value; if you specify a parent attribute, the root item is a child of the specified parent. • no: uses the item specified by the parent attribute as the immediate parent of all items generated by this tag. If there is no parent attribute, use the query as the parent. • A string: creates a root item and uses the specified string as the item name; if you specify a parent attribute, the root item is a child of the specified parent.
target	Optional; All		Target attribute of href URL. When populating a tree with data from a query, specify targets in a comma-delimited list, for example: target = "FRAME_BODY,_blank"

Usage

For this tag to work properly. the browser must be JavaScript-enabled. This tag must be a child of a cftree tag.

The cftreeitem tag has three basic formats:

- If you do not use a query or bind attribute to populate this tag, it creates a single tree item.
- If you use a query, it creates multiple items; each row of the query creates a hierarchically nested set of items with one item per column.
- If you use the bind attribute, the client side tree dynamically gets the data for the tree item's immediate children, and creates the child items, when a tree item expands. For detailed information on using the bind attribute to populate an HTML format tree, see "Using HTML format trees" on page 636 in the *ColdFusion Developer's Guide*.

Example

The following example creates a simple tree by using a single cftreeitem tag and a query:

```
<cfform action = "#cgi.script_name#">
    <cftree name = "Employees" height = "400" width = "200">
        <cftreeitem value="LastName, FirstName, Emp_ID" query="getEmployees"
         queryAsRoot="False">
    </cftree>
</cfform>
```

The following example creates a tree that shows the basic information about all employees in an organization, organized by department. The departments are expanded to show all employees. You click the + signs to display additional information. If you click the employee name, ColdFusion links back to the same page and displays the selected employee's ID.

```
<!--- Query the datasource to get employee information.--->
<!--- Group the output by Department.
    (All fields are required in Group By clause.) --->
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
    SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
    FROM Employees
    GROUP BY Department, Emp_ID, FirstName, LastName, EMail, Phone
</cfquery>
<html>
<body>
<h3>cftreeitem Example</h3>

<!--- The following runs if the user clicked on a link in the tree.
    A complete application would use the ID for additional processing. --->
<cfif isdefined("URL.user_ID")>
    <cfoutput>
        <!--- URL.cftreeitemkey is the selected tree item's value attribute. --->
        You Requested information on #URL.cftreeitemKey#; User ID #URL.user_ID#
    </cfoutput>
    <br><br>
</cfif>
<!--- Display the tree. The cftree tag must be in a cfform. --->
<cfform>
    <cftree name = "Employees" height = "400" width = "200"
        font = "Arial Narrow" highlighthref="No" hscroll="No">
        <!--- cfoutput tag with a group attribute loops over the departments. --->
        <cfoutput group="Department" query = "GetEmployees">
            <cftreeitem value="#Department#" parent="Employees" expand="yes">
            <!--- This cfoutput tag loops over the records for the department.
                    The cfoutput tag does not need any attributes. --->
            <cfoutput>
                <!--- Create an item for each employee in the department.
                    Do not expand children. Each employee name links to this page,
                    and sends the employee ID in the query string.--->
                <cftreeitem value = "#LastName#, #FirstName#"
                    display = "#LastName#, #FirstName#"
                    parent = "#Department#" expand="no"
                    href="#cgi.script_name#?user_id=#emp_id#">
                <!--- Each Employee entry has ID and ContactInfo children. --->
                <cftreeitem value = "#Emp_ID#" display = "Employee ID: #Emp_ID#"
                    parent = "#LastName#, #FirstName#">
                <!--- Each node must be unique value, so use Emp_ID om value. --->
                <cftreeitem value = "#Emp_ID#_ContactInfo"
                    display = "Contact Information"
                    parent = "#LastName#, #FirstName#" expand = "No">
                <!--- ContactInfo has two children. --->
                <cftreeitem value = "#Phone#" parent = "#Emp_ID#_ContactInfo">
                <cftreeitem value = "#Email#" parent = "#Emp_ID#_ContactInfo">
            </cfoutput>
        </cfoutput>
    </cftree>
</cfform>
```

cftry

Description

Used with one or more cfcatch tags. Together, they catch and process exceptions in ColdFusion pages. *Exceptions* are events that disrupt the normal flow of instructions in a ColdFusion page, such as failed database operations, missing include files, and developer-specified events.

Category

Exception handling tags

Syntax

```
<cftry>
    Code that might throw an exception
    One or more cfcatch blocks
</cftry>
```

See also

cfcatch, cferror, cfrethrow, cfthrow, onError; "Handling Errors" on page 247 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Changed cfscript to include try and catch statements that are equivalent to the cftry and cfcatch tags.

Usage

Within a cftry block, put the code that might throw an exception, followed by one ore more cfcatch tags that catch and process exceptions. This tag requires an end tag.

Example

```
<!--- cftry example, using TagContext to display the tag stack. --->
<h3>cftry Example</h3>
<!--- Open a cftry block. --->
<cftry>
    <!--- Note misspelled tablename "employees" as "employeeas". --->
    <cfquery name = "TestQuery" dataSource = "cfdocexamples">
        SELECT *
        FROM EMPLOYEES
    </cfquery>

    <!--- <p>... other processing goes here --->
    <!--- specify the type of error for which we search --->
    <cfcatch type = "Database">
    <!--- the message to display --->
        <h3>You've Thrown a Database <b>Error</b></h3>
        <cfoutput>
            <!--- and the diagnostic message from the ColdFusion server --->
            <p>#cfcatch.message#</p>
            <p>Caught an exception, type = #CFCATCH.TYPE# </p>
            <p>The contents of the tag stack are:</p>
            <cfloop index = i from = 1
                    to = #ArrayLen(CFCATCH.TAGCONTEXT)#>
                <cfset sCurrent = #CFCATCH.TAGCONTEXT[i]#>
                <br>#i# #sCurrent["ID"]#
                    (#sCurrent["LINE"]#,#sCurrent["COLUMN"]#)
                    #sCurrent["TEMPLATE"]#
```

```
            </cfloop>
        </cfoutput>
    </cfcatch>
</cftry>
```

cfupdate

Description

Updates records in a data source from data in a ColdFusion form or form Scope.

Category

Database manipulation tags

Syntax

```
<cfupdate
    dataSource = "ds_name"
    tableName = "table_name"
    formFields = "field_names"
    password = "password"
    tableOwner = "name"
    tableQualifier = "qualifier"
    username = "username">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfinsert`, `cfprocparam`, `cfprocresult`, `cfquery`, `cfqueryparam`, `cfstoredproc`, `cftransaction`; "Creating an update action page with cfupdate" on page 409 in the *ColdFusion Developer's Guide*.

History

ColdFusion MX: Deprecated the `connectString`, `dbName`, `dbServer`, `dbtype`, `provider`, and `providerDSN` attributes. They do not work, and might cause an error, in releases later than ColdFusion 5.

Attributes

Attribute	Req/Opt	Default	Description
`dataSource`	Required		Name of the data source that contains the table.
`tableName`	Required		Name of table to update. • For Oracle drivers, must be uppercase. • For Sybase driver, case-sensitive; must be in same case as used when the table was created.
`formFields`	Optional	(all on form, except `keys`)	Comma-delimited list of form fields to update. If a form field is not matched by a column name in the database, ColdFusion throws an error. The formFields list must include the database table primary key field, which must be present in the form. It can be hidden.
`password`	Optional		Overrides the `password` value specified in ODBC setup.
`tableOwner`	Optional		For data sources that support table ownership (for example, SQL Server, Oracle, Sybase SQL Anywhere), the table owner.

Attribute	Req/Opt	Default	Description
tableQualifier	Optional		For data sources that support table qualifiers. The purpose of table qualifiers is as follows: • SQL Server and Oracle: name of the database that contains the table • Intersolv dBASE driver: directory of DBF files
username	Optional		Overrides username value specified in ODBC setup.

Example

```
<!--- This example lets you update a person's telephone number in the employee table. --->
<cfif isDefined("form.phone")>
    <cfupdate datasource="cfdocexamples" tablename="EMPLOYEES">
</cfif>

<cfquery name="empTable" datasource="cfdocexamples">
    SELECT * FROM EMPLOYEES
</cfquery>

<!--- This code shows the contents of the employee table and allows you to choose a row for
updating. --->
<table border="1">
<cfoutput query="empTable">
    <tr>
        <td>#firstName#</td>
        <td>#lastName#</td>
        <td>#phone#</td>
        <td><a href="cfupdate.cfm?id=#emp_id#">Edit</a></td>
    </tr>
</cfoutput>
</table>

<cfif isDefined("url.id")>
    <cfquery name="phoneQuery" datasource="cfdocexamples">
        SELECT * FROM employees WHERE emp_id=#url.id#
    </cfquery>
<!--- This code displays the row to edit for update. --->
    <cfoutput query="phoneQuery">
        <form action="cfupdate.cfm" method="post">
        #phoneQuery.firstName# #phoneQuery.lastName#
        <input name="phone" type="text" value="#phone#" size="12">
        <input type="submit" value="Update">
        <input name="emp_id" type="hidden" value="#emp_id#">
        <!--- The emp_id is passed as a hidden field to be used as a primary
            key in the CFUPDATE. --->
        </form>
    </cfoutput>
</cfif>
```

cfwddx

Description

Serializes and deserializes CFML data structures to the XML-based WDDX format. The WDDX is an XML vocabulary for describing complex data structures in a standard, generic way. Implementing it lets you use the HTTP protocol to such information among application server platforms, application servers, and browsers.

This tag generates JavaScript statements to instantiate JavaScript objects equivalent to the contents of a WDDX packet or CFML data structure. Interoperates with Unicode.

Category

Extensibility tags

Syntax

```
<cfwddx
    action = "cfml2wddx|wddx2cfml|cfml2js|wddx2js"
    input = "inputdata"
    output = "result variable name"
    topLevelVariable = "top-level variable name for JavaScript"
    useTimeZoneInfo = "yes|no"
    validate = "yes|no" >
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfcollection`, `cfdump`, `cfexecute`, `cfindex`, `cfobject`, `cfreport`, `cfsearch`, `ToScript`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History

ColdFusion MX

- Changed column name case behavior: ColdFusion preserves the case of column names in JavaScript. (Earlier releases converted query column names to lowercase.)
- Changed encoding format support: this tag supports several encoding formats. The default encoding format is UTF-8. The tag interoperates with Unicode.

Attributes

Attribute	Req/Opt	Default	Description
`action`	Required		• `cfml2wddx`: serializes CFML to WDDX. • `wddx2cfml`: deserializes WDDX to CFML. • `cfml2js`: serializes CFML to JavaScript. • `wddx2js`: deserializes WDDX to JavaScript.
`input`	Required		A value to process.
`output`	Required if `action = "wddx2cfml"`		Name of variable for output. If `action = "WDDX2JS"` or `"CFML2JS"`, and this attribute is omitted, result is output in HTML stream.

Attribute	Req/Opt	Default	Description
topLevelVariable	Required if action = "wddx2js" or "cfml2js"		Name of top-level JavaScript object created by deserialization. The object is an instance of the WddxRecordset object.
useTimeZoneInfo	Optional	yes	Whether to output time-zone information when serializing CFML to WDDX. • yes: the hour-minute offset, represented in ISO8601 format, is output. • No: the local time is output.
validate	Optional	no	Applies if action = "wddx2cfml" or "wddx2js". • yes: validates WDDX input with an XML parser using WDDX DTD. If parser processes input without error, packet is deserialized. Otherwise, an error is thrown. • no: does not perform input validation.

Usage

ColdFusion preserves the case of column names cases in JavaScript.

The wddx2js and cfml2js actions create a WddxRecordset javascript object when they encounter a RecordSet java object. The serialized JavaScript code requires a wddx.js file.

This tag performs the following conversions:

From	To
CFML	WDDX
CFML	JavaScript
WDDX	CFML
WDDX	JavaScript

For more information, and a list of the ColdFusion array and structure functions that you can use to manage XML document objects and functions, see "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*.

Note: *The cfwddx tag throws an exception if you attempt to serialize a CFC or user-defined function (UDF).*

Example

```
<!--- This example shows basic use of the cfwddx tag. --->
<html>
<body>
<!--- Create a simple query. --->
<cfquery name = "q" dataSource = "cfdocexamples">
    SELECT Message_Id, Thread_id, Username FROM messages
</cfquery>

The recordset data is:...<p>
<cfoutput query = q>
    #Message_ID# #Thread_ID# #Username#<br>
</cfoutput><p>

<!--- Serialize data to WDDX format. --->
Serializing CFML data...<p>
<cfwddx action = "cfml2wddx" input = #q# output = "wddxText">

<!--- Display WDDX XML packet. --->
```

```
Resulting WDDX packet is:
<xmp><cfoutput>#wddxText#</cfoutput></xmp>

<!--- Deserialize to a variable named wddxResult. --->
Deserializing WDDX packet...<p>
<cfwddx action = "wddx2cfml" input = #wddxText# output = "qnew">

The recordset data is:...<p>
<cfoutput query = qnew>
    #Message_ID# #Thread_ID# #Username#<br>
</cfoutput><p>
```

cfwindow

Description

Creates a pop-up window in the browser. Does not create a separate browser pop-up instance.

Category

Display management tags

Syntax

```
<cfwindow
    bodyStyle = "CSS style specification"
    center="true|false"
    closable="true|false"
    draggable="true|false"
    headerStyle = "CSS style specification"
    height="number of pixels"
    initShow="false|true"
    minHeight="number of pixels"
    minWidth="number of pixels"
    modal="true|false"
    name="string"
    onBindError = "JavaScript function name"
    refreshOnShow = "false|true"
    resizable="true|false"
    source="path"
    title="string"
    width="number of pixels"
    x="numeric pixel coordinate"
    y="numeric pixel coordinate">

        window contents

</cfwindow>
```

If you use the `source` attribute, ColdFusion ignores any tag body contents. If you do not have a tag body and omit the `</window>` end tag, you must close the `cfwindow` tag with the /> character combination.

Note: *You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.*

See also

`cfajaximport`, `cfdiv`, `cflayout`, `cfpod`, `ColdFusion.Window.create`, "Using pop-up windows" on page 620 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
bodyStyle	Optional		A CSS style specification for the window body. As a general rule, use this attribute to set color and font styles. Using this attribute to set the height and width, for example, can result in distorted output.
center	Optional	false	A Boolean value that specifies whether to center the window over the browser window. • If true, ColdFusion ignores the x and y attribute values. • If false, and you do not specify x and y attributes, ColdFusion centers the window.
closable	Optional	true	A Boolean value that specifies whether the user can close the window. If true, the window has an X close icon.
draggable	Optional	true	A Boolean value that specifies whether the user can drag the window. To drag the window, click the mouse on the title bar and hold the button down while dragging. If the window does not have a title, users cannot drag it.
headerStyle	Optional		A CSS style specification for the window header. As a general rule, use this attribute to set color and font styles. Using this attribute to set the height and width, for example, can result in distorted output.
height	Optional	300	Height of the window in pixels. If you specify a value greater than the available space, the window occupies the available space and the resize handles do not appear.
initShow	Optional	false	A Boolean value that specifies whether to display the window when the containing page first appears. If this value is false, use the ColdFusion.Window.show JavaScript function to display the window.
minHeight	Optional	0	The minimum height, in pixels, to which users can resize the window.
minWidth	Optional	0	The minimum width, in pixels, to which users can resize the window.
modal	Optional	false	A Boolean value that specifies whether the window is modal, that is, whether the user can interact with the main window while this window is displayed. If true, the user *cannot* interact with the main window.
name	Optional		The name of the window. Must be unique on the pages. This attribute is required to interact with the window, including to dynamically show or hide it.
onBindError	Optional	see Description	The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up appears.
refreshOnShow	Optional	false	• true Refresh the contents of the window by running the source bind expression whenever the window shows (for example, by calling the ColdFusion.Window.show JavaScript function), in addition to when bind events occur • false Refresh the window only when the bind expression is triggered by its bind event. To use this attribute, you must also specify a source attribute.
resizable	Optional	true	A Boolean value specifying whether the user can resize the window.

Attribute	Req/Opt	Default	Description
source	Optional		A URL that returns the window contents. This attribute can use URL parameters to pass data to the page. ColdFusion uses standard page path resolution rules to locate the page. You can use a bind expressions in this attribute; for more information see Usage. **Note:** If a CFML page specified in this attribute contains tags that use Ajax features, such as cfform, cfgrid, and cfpod, you must use a cfajaximport tag on the page with the cfwindow tag. For more information, see cfajaximport.
title	Optional		The text to display in the window's title bar. You can use HTML mark-up to control the title appearance; for example, to show the text in red italic font.
width	Optional	500	Width of the window in pixels. If you specify a value greater than the available space, the window occupies the available space and the resize handles do not appear.
x	Optional		The X (horizontal) coordinate of the upper-left corner of the window, relative to the browser window. ColdFusion ignores this attribute if the center attribute value is true and if you do not set the y attribute value.
y	Optional		The Y (vertical) coordinate of the upper-left corner of the window, relative to the browser window. ColdFusion ignores this attribute if the center attribute value is true and if you do not set the x attribute value.

Usage

You cannot use this tag in a form or as a child of a cflayout, or cflayoutarea tag.

You must define the cfwindow tag on the page that displays it (or a page that is included by using the cfinclude tag). So, you cannot use the cfwindow tag on a page that is specified by a cfmenuitem tag http attribute, cfdiv tag bind attribute, or cflayoutarea or cfpod tag source attribute. Instead, for example, you can display a window when a user clicks a menu item by defining the window on the same page as your menu and using a JavaScript function in the cfmenuitem tag http attribute to call the window's show function. The cfwindow tag uses its source attribute to get its contents from another page.

You can use a source attribute or a tag body to specify the window contents; if you specify both, ColdFusion uses the contents specified by the source attribute and ignores the tag body. If you use a source attribute, an animated icon and the text "Loading..." appears while the contents is being fetched.

If the source attribute specifies a page that defines JavaScript functions, the function definitions on that page must have the following format:

```
functionName = function(arguments) {function body}
```

Function definitions that use the following format may not work:

```
function functionName (arguments) {function body}
```

However, Adobe recommends that you include all custom JavaScript in external JavaScript files and import them on the application's main page, and not write them inline in code that you get by using the source attribute. Imported pages do not have this function definition format restriction.

If you use the source attribute, you can use a *bind expression* to include form field values or other form control attributes as part of the source specification. You can bind to HTML format form controls only. For detailed information on using bind expressions, see "Binding data to form fields" on page 650 in the *ColdFusion Developer's Guide*.

JavaScript functions

You can use the following JavaScript functions to manage an HTML format window:

Function	Description
ColdFusion.Window.create	Creates a window without using a cfwindow tag.
ColdFusion.Window.getWindowObject	Gets the underlying Ext JS - JavaScript Library object for the specified HTML format cfwindow control.
ColdFusion.Window.hide	Closes a window.
ColdFusion.Window.onHide	Specifies a function to run each time a specific window closes.
ColdFusion.Window.onShow	Specifies a function to run each time a specific window opens.
ColdFusion.Window.show	Opens a window.

Example

The following example shows several features of the cfwindow tag and dynamic binding of the cfwindow tag source attribute to form controls. It shows how you can use x and y attributes to position the windows and how several attributes, such as closable and resizable affect the window appearance. It also shows how you can use bind expressions to dynamically update window contents when form control values change, including different ways to trigger updating the window contents.

```
<html>
<head>
</head>

<body>
<cfform name="myform">
    <cfinput type="hidden" name="hiddentext"
        value="This is hidden text from the main page">

    Click the mouse on the control to show its text in window 1.
    <cfinput name="text1"><br />

    Click the button to show the input control text in window 2.
    <cfinput name="text2">
    <cfinput type="button" name="mybutton" value="Show Text"
        onclick="javascript:ColdFusion.Window.show('mywindow2')"><br />

    Click the Checkbox to change and show its status in window 3
    <cfinput name="check1" type="checkbox"><br />

    Click the button to open a window containing the page
        specified by the input control.
    <cfinput name="text3" value="windowsource.cfm">
    <cfinput type="button" name="mybutton3" value="Open Window"
        onclick="javascript:ColdFusion.Window.show('mywindow4')">
</cfform>

<!--- This window shows initially and cannot be closed, dragged, or resized.
        The value of the text URL parameter, and therefore, the contents of the
        text displayed in the window changes each time the user clicks the
        mouse over the text1 control. --->
<cfwindow x="0" y="100" width="200" height="150"
        name="mywindow" title="My First Window"
        closable="false" draggable="false" resizable="false" initshow="true"
        source="windowsource.cfm?text={myform:text1@mousedown}" />

<!--- This window shows initially and cannot be dragged, or resized, but can
        be closed.
```

```
        The text URL parameter represents the text2 input control value. --->
<cfwindow x="0" y="250" width="200" height="150"
        name="mywindow2" title="My Second Window"
        initshow="true" draggable="false" resizable="false"
        source="windowsource.cfm?text={myform:text2}" />

<!--- This window shows initially and cannot be resized, but can be dragged
        or closed.
        The value of the text URL parameter, and therefore, Boolean value
        displayed in the window changes each time the user clicks the mouse
        in the check1 control to change the check box status.
        The bind expression binds to the check box checked attribute;
        it specifies a click event because Internet Explorer does not
        generate a change event when the user clicks the box.--->
<cfwindow x="0" y="400" width="200" height="150"
        name="mywindow3" title="My Third Window"
        initshow="true" resizable="false"
        source="windowsource.cfm?text=<b>Checkbox: </b>{myform:check1.checked@click}" />

<!--- This window does not display initially.
        The Open Window button control opens it.
        It can be closed, dragged, and resized.
        The value text URL parameter represents the value of a hidden text
        field. --->
<cfwindow x="210" y="100" width="500" height="480" name="mywindow4"
        minHeight="400" minWidth="400"
        title="My Fourth Window" initshow="false"
        source="{myform:text3}?text={myform:hiddentext}" />
</body>
</html>
```

The windowsource.cfm page that the `cfwindow` tag `source` attributes specify to display in the windows contains the following code:

```
<h3>Main page input:</h3>
<cfoutput>
    #url.text#
</cfoutput>
```

cfxml

Description

Creates a ColdFusion XML document object that contains the markup in the tag body. This tag can include XML and CFML tags. ColdFusion processes the CFML code in the tag body, and then assigns the resulting text to an XML document object variable, which is always stored in Unicode.

Category

Extensibility tags

Syntax

```
<cfxml
    variable="xmlVarName"
    caseSensitive="yes|no">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`IsXmlDoc`, `IsXmlElem`, `IsXmlRoot`, `ToString`, `XmlChildPos`, `XmlNew`, `XmlParse`, `XmlSearch`, `XmlTransform`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added support for using an XML declaration at the start of the text.

ColdFusion MX: Added this tag.

Attributes

Attribute	Req/Opt	Default	Description
`variable`			Name of the document object.
`caseSensitive`	Optional	`no`	• `yes`: maintains the case of document elements and attributes. • `no`

Usage

If your XML object is case-sensitive, you cannot use dot notation to reference an element or attribute name. Use the name in associative array (bracket) notation, or a reference that does not use the case-sensitive name (such as xmlChildren[1]) instead. In the following code, the first line will work with a case-sensitive XML object. The second and third lines cause errors:

```
MyDoc.xmlRoot.XmlAttributes["Version"] = "12b";
MyDoc.xmlRoot.XmlAttributes.Version = "12b";
MyDoc.MyRoot.XmlAttributes["Version"] = "12b";
```

Use the `XmlFormat` function to escape special characters such as &, > and <.

To convert an XML document object back into a string, use the `ToString` function, at which time ColdFusion automatically prepends the <?xml version="1.0" encoding="UTF-8" ?> XML declaration .

To change the declaration to specify another encoding, use the `Replace` function. To specify the encoding of the text that is returned to a browser or other application, use the `cfcontent` tag.

The following example illustrates this process:

```
<cfprocessingdirective suppresswhitespace="Yes">
<cfcontent type="text/xml; charset=utf-16">
<cfxml variable="xmlobject">
<breakfast_menu>
    <food>
        <name quantity="50">Belgian Waffles</name>
        <description>Our famous Belgian Waffles</description>
    </food>
</breakfast_menu>
</cfxml>

<!--- <cfdump var="#xmlobject#">--->

<cfset myvar=toString(xmlobject)>
<cfset mynewvar=replace(myvar, "UTF-8", "utf-16")>

<!---<cfdump var="#mynewvar#">--->

<cfoutput>#mynewvar#</cfoutput>
</cfprocessingdirective>
```

The `cfprocessingdirective` tag prevents ColdFusion from putting white space characters in front of the XML declaration.

Example

This following example creates a document object whose root element is MyDoc. The object includes text that displays the value of the ColdFusion variable testVar. The code creates four nested child elements, which are generated by an indexed `cfloop` tag. The `cfdump` tag displays the XML document object.

```
<cfset testVar = True>
<cfxml variable="MyDoc">
    <?xml version='1.0' encoding='utf-8' ?>
    <MyDoc>
        <cfif testVar IS True>
            <cfoutput>The value of testVar is True.</cfoutput>
        <cfelse>
            <cfoutput>The value of testVar is False.</cfoutput>
        </cfif>
        <cfloop index = "LoopCount" from = "1" to = "4">
            <childNode>
                This is Child node <cfoutput>#LoopCount#.</cfoutput>
            </childNode>
        </cfloop>
    </MyDoc>
</cfxml>
<cfdump var=#MyDoc#>
```

cfzip

Description

Manipulates ZIP and Java Archive (JAR) files. In addition to the basic zip and unzip functions, use the `cfzip` tag to delete entries from an archive, filter files, read files in binary format, list the contents of an archive, and specify an entry path used in an executable JAR file.

History

ColdFusion 8: Added this tag.

Category

File management tags

Syntax

```
delete
<cfzip
    required
    action = "delete"
    file = "absolute pathname"
    optional
    entrypath = "full pathname"
    filter = "file filter"
    recurse = "yes|no">

list
<cfzip
    required
    action = "list"
    file = "absolute pathname"
    name = "recordset name"
    optional
    filter = "file filter"
    recurse = "yes|no"
    showDirectory= "yes|no">

read
<cfzip
    required
    action = "read"
    entrypath = "full pathname"
    file = "absolute pathname"
    variable = "variable name"
    optional
    charset = "encoding type">

readBinary
<cfzip
    required
    action = "readBinary"
    entrypath = "full pathname"
    file = "absolute pathname"
    variable = "variable name">

unzip
<cfzip
    required
    action = "unzip"
```

```
    destination = "destination directory"
    file = "absolute pathname"
    optional
    entrypath = "full pathname"
    filter = "file filter"
    overwrite = "yes|no"
    recurse = "yes|no"
    storePath = "yes|no">

zip
<cfzip
    required
    file = "absolute pathname"
    One of the following:
    source = "source directory"
    <cfzipparam source = "source directory" ...>
    optional
    action = "zip"
    filter = "file filter"
    overwrite = "yes|no"
    prefix = "string"
    recurse = "yes|no"
    storePath = "yes|no">
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

`cfzipparam`

Attributes

Attribute	Action	Req/Opt	Default	Description
`action`	N/A	Optional	`zip`	Action to take. Must be one of the following: • `delete` • `list` • `read` • `readBinary` • `unzip` • `zip` If you do not specify an action, ColdFusion applies the default action, `zip`.
`charset`	read	Optional	default encoding of the host machine	Character set used to translate the ZIP or JAR entry into a text string. Examples of character sets: • `JIS` • `RFC1345` • `UTF-16`
`destination`	unzip	Required		Destination directory where the ZIP or JAR file is extracted.
`entryPath`	delete read readBinary unzip	Optional		Pathname on which the action is performed.

Attribute	Action	Req/Opt	Default	Description
file	delete list read readBinary unzip zip	Required		Absolute pathname of the file on which the action is performed; for example, the full pathname of the ZIP file: c:\temp\log.zip. If you do not specify the full pathname (for example, file="log.zip"), ColdFusion creates the file in a temporary directory. You can use the GetTempDirectory function to access the ZIP or JAR file.
filter	delete list unzip zip	Optional		File filter applied to the action. The action applies to all files in the specified pathname that match the filter.
name	list	Required		Record set name in which the result of the list action is stored. The record set columns are the following: • name: Filename of the entry in the JAR file. For example, if the entry is help/docs/index.htm, the name is index.htm. • directory: Directory containing the entry. For the preceding example, the directory is help/docs. You can obtain the full entry name by concatenating directory and name. If an entry is at the root level, the directory is empty (" "). • size: Uncompressed size of the entry, in bytes. • compressedSize: Compressed size of the entry, in bytes. • type: Type of entry (directory or file). • dateLastModified: Last modified date of the entry, cfdate object. • comment: Any comment, if present, for the entry. • crc: Crc-32 checksum of the uncompressed entry data.
overwrite	unzip zip	Optional	no	unzip: Specifies whether to overwrite the extracted files: • yes: If the extracted file already exists at the destination specified, the file is overwritten. • no: If the extracted file already exists at the destination specified, the file is not overwritten and that entry is not extracted. The remaining entries are extracted. zip: Specifies whether to overwrite the contents of a ZIP or JAR file: • yes: Overwrites all of the content in the ZIP or JAR file if it exists. • no: Updates existing entries and adds new entries to the ZIP or JAR file if it exists.
prefix	zip	Optional		String added as a prefix to the ZIP or JAR entry. The string is the name of a subdirectory in which the entries are added.
recurse	delete list unzip zip	Optional	yes	Specifies whether the action applies to subdirectories: • yes: Includes subdirectories. • no: Does not include subdirectories.
showDirectory	list	Optional	no	Specifies whether to show the directory structure: • yes: Lists the directories. • no: Does not list directories.
source	zip	Required (see description)		Source directory to be zipped. Not required if the cfzipparam tag is specified.

Attribute	Action	Req/Opt	Default	Description
storePath	unzip zip	Optional	yes	unzip: Specifies whether files are stored at the entry path: • yes: The files are extracted to the entry path. • no: The entry path is ignored and all the files are extracted at the root level. zip: Specifies whether pathnames are stored in the ZIP or JAR file: • yes: Pathnames of entries are stored in the ZIP or JAR file. • no: Pathnames of the entries are not stored in the ZIP or JAR file. All the files are placed at the root level. In case of a name conflict, the last file in the iteration is added.
variable	read readBinary	Required		Variable in which the content is stored.

Usage

Use the cfzip tag to zip and unzip files and manipulate existing ZIP or JAR files in ColdFusion. You can use the cfzip tag independently or with one or more cfzipparam tags to manipulate multiple files or directories. The cfzip tag is the parent tag of the cfzipparam tag.

The ZIP format is the standard format for file archiving and compression. The JAR format is based on the ZIP format. JAR files are platform-independent.

Note: *The cfzip tag does not create directories. If you specify a directory that does not exist, ColdFusion generates an error.*

delete action

Use the delete action to delete entries from a ZIP or JAR file.

```
<!--- This example shows how to delete all the properties in a JAR file.
    --->
<cfzip file="e:\work\soure.jar" action="delete" filter="*.properties, *.props">

<!--- This example shows how to delete all of the entries in a ZIP file with a JPG, GIF, or
PNG extension, including entries in subdirectories. --->
<cfzip file="c:\myApp\images.zip" action="delete" filter="*.jpg, *.gif, *.png"
recurse="yes">

<!--- This example shows how to delete the "images" subdirectory (and its contents) from the
"myApp.zip" file. --->
<cfzip action="delete" file="c:\myApp.zip" entrypath="images">

<!--- This example shows how to delete all Java source entries in the "work/source" directory
and images (*.gif, *.jpg, *.jpeg) from a JAR file. --->
<cfzip file="/downloads/source.jar" action="delete">
    <cfzipparam entrypath="work/source" filter="*.java">
    <cfzipparam filter="*.gif,*.jpg,*.jpeg">
</cfzip>
```

list action

Use the list action to list the entries of a ZIP or JAR file. The following table shows the types of information you can retrieve for entries in the archive:

Field	Description
comment	Text string description saved with the entry source.
compressedSize	Compressed size of the entry in bytes.
crc	Checksum for the entry source.
dateLastModified	Date and time when the source was last modified.
directory	Name of the directory where the entry is stored.
name	Entry pathname.
size	Uncompressed size of the entry source in bytes.
type	Source type for the entry, for example, *file*.

You can use the `cfdump` tag to list all of the information in a ZIP or JAR file, as the following example shows:

```
<cfzip file="c:/myApp.jar" action="list" name="entry">
<cfdump var="#entry#">
```

You can use the `cfoutput` tag to list individual fields for the entries in an archive, as the following example shows:

```
<cfzip file="c:\zipTest\Test.zip" action="list" name="entry">
<table>
    <cfoutput>
    <tr>
        <td><b>Entry Name:</b> #entry.name#</td>
        <td><b>Last Modified Date:</b>
#dateFormat(entry.dateLastModified)#,#timeFormat(entry.dateLastModified)#</td>
        <td><b>Size (uncompressed):</b> #numberFormat(entry.size/1000)# KB
    <br></td>
 </cfoutput>
 </tr>
</table>
```

read action

Use the `read` action to read the content of the ZIP or JAR file entry in human-readable format. The `read` action uses the `charset` value to create the string.

```
<!--- This example shows how to read a text file in a JAR file. --->
<cfzip action="read" file="/home/sam/work/util.jar" entrypath="info.txt" variable="text">
```

readBinary action

Use the `readBinary` action to read the content of a ZIP or JAR file in binary format.

```
<!--- This example shows how to use the readBinary action to copy a ZIP entry from one ZIP
file to another ZIP file. --->
<cfzip file="c:\work\instr.zip" action="readBinary"
    entryPath="com/test/abc.jpg" variable="xyz">
<cfzip file="c:\work\copy_instr.zip">
    <cfzipparam entryPath="com/test/xyz.jpg" content="#xyz#">
</cfzip>
```

unzip action

Use the `unzip` action to extract the entries from a ZIP or JAR file.

```
<!--- This example shows how to extract the class files of a JAR file and save the files to
a local drive. --->
```

```
<cfzip file="e:\work\tools.jar" action="unzip" filter="*.class"
destination="c:\temp\tools\classes"/>

<!--- This example shows how to extract files from a JAR file in multiple directories. --->
<cfzip file="e:\work\images.jar" action="unzip" destination="c:\images">
    <cfzipparam entryPath="toWork\small">
    <cfzipparam entryPath="final\large">
</cfzip>
```

zip action

Use the `zip` action to create or update a ZIP or JAR file. This is the default action; you do not have to specify it explicitly. If you specify a ZIP or JAR file that does not exist, ColdFusion creates it. If the ZIP or JAR file exists, ColdFusion adds new entries from the source and updates existing entries if they have changed. If you set the `overwrite` attribute to `yes`, all of the entries in the ZIP or JAR file are replaced by the new content.

```
<!--- This example shows how to zip the directory "c:\temp" into the ZIP file
"e:\work\abc.zip". --->
<cfzip file="e:\work\abc.zip" source="c:\temp">

<!--- This example shows how to zip all the class files in a directory and add a subdirectory
named "classes" to the JAR file entry name. --->
<cfzip file="e:\work\util.jar" action="zip" source="c:\src\util\" prefix="classes"
filter="*.class">

<!---This example shows how to zip all of the log files in the ColdFusion directory and
create a subdirectory called exception where zipped files are archived.
<cfzip file="c:\zipTest\log2.zip" action="zip" source="c:\ColdFusion\" prefix="exception"
filter="*.log">

<!--- This example shows how to overwrite all of the content of a ZIP file with the entries
specified in the source. --->
<cfzip file="c:\currentApp.zip" source="c:\myApp\work" overwrite="yes">
```

Example

The following example shows how to zip image files chosen from a form and e-mail the ZIP file to the person requesting the images.

The first ColdFusion page populates a pop-up menu with the names of artists generated from a database query:

```
<!--- Create a query to extract artist names from the cfartgallery database. --->
<cfquery name="artist" datasource="cfartgallery">
    SELECT FIRSTNAME || ' ' || LASTNAME AS FULLNAME,ARTISTS.ARTISTID
    FROM ARTISTS
</cfquery>

<!--- Create a form that lists the artists generated by the query. --->
<cfform action="zipArt_action.cfm" method="post">
<h3>Choose an Artist</h3>
<p>Please choose an artist:</p>
<cfselect name="artistName" query="artist" display="FULLNAME" value="ARTISTID"
required="yes" multiple="no" size="8">
</cfselect>
<br/><cfinput type="submit" name="submit" value="OK">
</cfform>
```

The first action page displays the images by the selected artist, zips the files, and writes the ZIP file to a temporary directory. Also, it includes a form to e-mail the ZIP file:

```
<!--- Create a query to extract artwork for the selected artist from the cfartgallery
database. --->
<cfquery name="artwork" datasource="cfartgallery">
    SELECT FIRSTNAME, LASTNAME, LARGEIMAGE
```

```
    FROM ARTISTS, ART
    WHERE ARTISTS.ARTISTID = ART.ARTISTID
    AND ARTISTS.ARTISTID=<cfqueryparam value="#form.artistName#">
    ORDER BY ARTNAME
</cfquery>

<cfoutput>
<p>You have chosen the work of #artwork.FirstName# #artwork.LastName#.</p>
<cfset thisDir = ExpandPath(".")>
<cfset imgDir = ExpandPath("..")>
</cfoutput>
<cfset xctr = 1>
<table border="0" cellpadding="15" cellspacing="0" bgcolor="#FFFFFF">
<cfoutput query="artwork">
    <cfif xctr mod 3 eq 1>
    <tr>
        </cfif>
        <!--- Use the IsImageFile function to verify that the image files
            extracted from the database are valid. Use the ImageNew function to
                create a ColdFusion image from valid image files. --->
        <cfif IsImageFile("#imgdir#/cfdocs/images/artgallery/
                #artwork.largeImage#")>
        <cfset myImage=ImageNew("#imgdir#/cfdocs/images/artgallery/
                #artwork.largeImage#")>
            <td valign="top" align="center" width="200">
            <cfset xctr = xctr + 1>
        <img src="#imgdir#/cfdocs/images/artgallery/#artwork.largeImage#"/>
            </td>

<!---Zip the files by the specified artist. --->
        <cfzip source="#imgDir#/cfdocs/images/artgallery/#artwork.LARGEIMAGE#"
            action="zip" file="#thisDir#/#artwork.lastname#.zip">
            </cfif>
                </cfoutput>
                </tr>
                </table>
<h3>Mail the ZIP File</h3>
<p>Please enter your e-mail address so we can send you the ZIP file as an attachment.</p>
<cfform action = "zipArt_action2.cfm" method="post">
Your e-mail address: <cfinput type = "Text" name = "MailTo">
<!--- Specify the required field. --->
    <cfinput type = "hidden" name = "MailTo_required" value = "You must enter
            your email address">
    <cfinput type="hidden" name="zipPath"
            value="#thisDir#/#artwork.lastname#.zip">
    <p><cfinput type = "Submit" name = "OK" label="Mail">
</cfform>
```

The second action page mails the ZIP file as an attachment:

```
<h3>Mail the ZIP file</h3>
<p>Your file has been mailed to you.</p>
<cfset eMail="#form.MailTo#">
<cfset zipPath="#form.zipPath#">
<cfmail from="coldfusion@adobe.com" to="#eMail#" subject="see zipped attachment">
    The images you requested are enclosed in a ZIP file.
            <cfmailparam file="#zipPath#">
</cfmail>
```

cfzipparam

Description

Provides additional information to the cfzip tag.

The cfzipparam tag is always a child tag of the cfzip tag.

History

ColdFusion 8: Added this tag.

Category

File management tags

Syntax

```
<cfzip ..>
    <cfzipparam
        charset = "encoding type"
        content = "variable name"
        entryPath = "full pathname"
        filter = "file filter"
        prefix = "string"
        recurse = "yes|no"
        source = "source directory">
</cfzip>
```

Note: *You can specify this tag's attributes in an* `attributeCollection` *attribute whose value is a structure. Specify the structure name in the* `attributeCollection` *attribute and use the tag's attribute names as structure keys.*

See also

cfzip

Attributes

Attribute	Req/Opt	Default	Description
charset	Optional	default encoding of the host machine	Converts string content into binary data before putting it into a ZIP or JAR file. Used only when cfzip action="zip" and the cfzipparam content is a string. Examples of character sets: • JIS • RFC1345 • UTF-16
content	Optional		Content written to the ZIP or JAR entry. Used only when cfzip action="zip". Valid content data types are binary and string. If you specify the content attribute, you must specify the entrypath attribute.
entryPath	Optional		Pathname used: • For cfzip action="zip", it is the entry path used. This is valid only when the source is a file. The entry path creates a subdirectory in the ZIP or JAR file. • For cfzip action="unzip", it is the pathname to unzip. • For cfzip action="delete", it is the pathname to delete from the ZIP or JAR file.

Attribute	Req/Opt	Default	Description
filter	Optional		File filter applied to the action. For example, for the zip action, all the files in the source directory that match the filter are zipped.
prefix	Optional		String added as a prefix to the ZIP or JAR entry. Used only when cfzip action="zip".
recurse	Optional	yes	Include the directory to be zipped, unzipped, or deleted, as specified by the cfzip parent tag.
source	Optional		Source directory or file. Used only when cfzip action="zip". Specified files are added to the ZIP or JAR file: • If you specify source attribute for the cfzip tag, the cfzipparam source is relative to it. • If you do not specify a source attribute for the cfzip tag, the cfzipparam source must be an absolute pathname.

Usage

Use the cfzipparam tag with the cfzip tag to zip, extract, or delete multiple files or directories. For example, to zip multiple directories, specify a cfzipparam tag for each source directory.

Example

Example 1

```
<!--- This example shows how to zip class files from one subdirectory and class and property
files from another directory into a JAR file. --->
<cfzip file="c:\util.jar" source="c:\myproj\classes">
    <cfzipparam source="com\abc\util" filter="*.class">
    <cfzipparam source="com\abc\io" filter="*.class, *.properties">
</cfzip>
```

Example 2

```
<!--- This example shows how to update a ZIP file with files from multiple locations, each
with a different filter. --->
<cfzip file="e:\work\test.jar" action="zip">
 <cfzipparam source="c:\temp\abc.txt" prefix="com\abc">
 <cfzipparam source="c:\src\classes" recurse="yes"
            filter="*.class,*.properties" prefix="classes">
 <cfzipparam source="c:\src\Manifest.MF" entrypath="META-INF\MANIFEST">
</cfzip>
```

Example 3

```
<!--- This example shows how to insert the string format for a programmatically generated
XML file into a ZIP file. --->
<!--- Create a variable that specifies a time-out period. --->
<cfset myDoc="<system-config><timeout>1500</timeout><pool-max-size>30
            </pool-max-size></system-config>">
<!--- Zip the file. --->
<cfzip file="e:\work\test.zip" action="zip">
 <cfzipparam source="c:\src\Manifest.MF" entrypath="META-INF\MANIFEST">
 <cfzipparam content="#myDoc#" entrypath="system-config.xml">
</cfzip>
```

Example 4

```
<!--- This example shows how to update a JAR file with a new version of the file and add
some new files to the JAR file. --->
<cfzip file="e:\work\admin.jar">
```

```
    <cfzipparam source="c:\src\classes" recurse="yes"
            filter="*.class,*.properties">
    <cfzipparam source="c:\src\Manifest.MF" entrypath="META-INF\MANIFEST">
</cfzip>
```

Example 5

The following example shows how to zip multiple image files chosen from a form and e-mail the ZIP file to the person requesting the images.

The first ColdFusion page populates a pop-up menu with the names of artists generated from a database query:

```
<!--- The following code creates a form for selecting images. --->
<h3>Select the images</h3>
<p>Please choose the images you would like sent to you.</p>
<!--- Create the ColdFusion form to select the images. --->
<table>
<cfform action="zip2_action.cfm" method="post"
    enctype="multipart/form-data">
    <tr>
        <td><img src="../cfdocs/images/artgallery/elecia01.jpg"/><br/>
        <cfinput type="checkbox" name="ck1" Value=1>Cube</td>
        <td><img src="../cfdocs/images/artgallery/elecia02.jpg"/><br/>
        <cfinput type="checkbox" name="ck2" Value=1>Pentagon</td>
        <td><img src="../cfdocs/images/artgallery/elecia03.jpg"/><br/>
        <cfinput type="checkbox" name="ck3" Value=1>Surfer Dude</td>
        <td><img src="../cfdocs/images/artgallery/elecia04.jpg"/><br/>
        <cfinput type="checkbox" name="ck4" Value=1>Surfer Girl</td></tr>
        <tr><td><cfinput type = "Submit" name = "OK" label="OK"></td></tr>
</table>
</cfform>
```

The first action page zips the files selected from the form, and writes the ZIP file to the hard drive. Also, it includes a form to e-mail the ZIP file:

```
<!--- Determine the absolute pathname on the server. --->
<cfset thisDir = ExpandPath(".")>

<!--- Create a ZIP file based on the selections from the form. Use the cfzipparam tag to
specify the source for each check box selection. --->
<cfzip file="c:\images.zip" source="#thisDir#" action="zip" overwrite="yes">
    <cfif IsDefined("form.ck1")>
        <cfzipparam source="../cfdocs/images/artgallery/elecia01.jpg">
        </cfif>
    <cfif IsDefined("form.ck2")>
        <cfzipparam source="../cfdocs/images/artgallery/elecia02.jpg">
        </cfif>
    <cfif IsDefined("form.ck3")>
        <cfzipparam source="../cfdocs/images/artgallery/elecia03.jpg">
        </cfif>
    <cfif IsDefined("form.ck4")>
        <cfzipparam source="../cfdocs/images/artgallery/elecia04.jpg">
        </cfif>
</cfzip>
<h3>Mail the ZIP File</h3>
<p>Please enter your e-mail address so we can send you the ZIP file as an attachment.</p>
<cfif IsDefined("form.mailto")>
    <cfif form.mailto is not "" >
        <cfoutput>
            <cfmail from="coldfusion@adobe.com" to="#form.mailto#"
                    subject="see zipped attachment">
    The images you requested are enclosed in a ZIP file.
```

```
            <cfmailparam file="#thisDir#/images.zip">
            </cfmail>
        </cfoutput>
        </cfif>
    </cfif>
<cfform action = "zipArt_action2.cfm" method="post">
    Your e-mail address: <cfinput type = "Text" name = "MailTo">
    <!--- Specify the required field. --->
    <cfinput type = "hidden" name = "MailTo_required"
    value = "You must enter your email address">
    <cfinput type = "hidden" name="zipPath" value = "c:\images.zip">
    <p><cfinput type = "Submit" name = "OK" label="Mail">
</cfform>
```

The second action page mails the ZIP file as an attachment:

```
<h3>Mail the ZIP file</h3>
<p>Your file has been mailed to you.</p>
<cfset eMail="#form.MailTo#">
<cfset zipPath="#form.zipPath#">
<cfmail from="coldfusion@adobe.com" to="#eMail#"
    subject="see zipped attachment">
    The images you requested are enclosed in a ZIP file.
    <cfmailparam file="#zipPath#">
</cfmail>
```

Chapter 4: ColdFusion Functions

The following tables list and categorize ColdFusion Markup Language (CFML) functions.

Contents

Function list

ColdFusion Markup Language (CFML) includes a set of functions that you use in ColdFusion 8 pages to perform logical and arithmetic operations and manipulate data.

The following table lists CFML functions:

Abs	ACos	AddSOAPRequestHeader
AddSOAPResponseHeader	AjaxLink	AjaxOnLoad
ArrayAppend	ArrayAvg	ArrayClear
ArrayDeleteAt	ArrayInsertAt	ArrayIsDefined
ArrayIsEmpty	ArrayLen	ArrayMax
ArrayMin	ArraySet	ArraySort
ArraySum	ArraySwap	ArrayToList
Asc	ASin	Atn
BinaryDecode	BinaryEncode	BitAnd
BitMaskClear	BitMaskRead	BitMaskSet
BitNot	BitOr	BitSHLN
BitSHRN	BitXor	Ceiling
CharsetDecode	CharsetEncode	Chr
CJustify	Compare	CompareNoCase
Cos	CreateDate	CreateDateTime
CreateObject	CreateODBCDate	CreateODBCDateTime
CreateODBCTime	CreateTime	CreateTimeSpan
CreateUUID	DateAdd	DateCompare
DateConvert	DateDiff	DateFormat
DatePart	Day	DayOfWeek

DayOfWeekAsString	DayOfYear	DaysInMonth
DaysInYear	DE	DecimalFormat
DecrementValue	Decrypt	DecryptBinary
DeleteClientVariable	DeserializeJSON	DirectoryExists
DollarFormat	DotNetToCFType	Duplicate
Encrypt	EncryptBinary	Evaluate
Exp	ExpandPath	FileClose
FileCopy	FileDelete	FileExists
FileIsEOF	FileMove	FileOpen
FileRead	FileReadBinary	FileReadLine
FileSetAccessMode	FileSetAttribute	FileSetLastModified
FileWrite	Find	FindNoCase
FindOneOf	FirstDayOfMonth	Fix
FormatBaseN	GenerateSecretKey	GetAuthUser
GetBaseTagData	GetBaseTagList	GetBaseTemplatePath
GetClientVariablesList	GetComponentMetaData	GetContextRoot
GetCurrentTemplatePath	GetDirectoryFromPath	GetEncoding
GetException	GetFileFromPath	GetFileInfo
GetFunctionList	GetGatewayHelper	GetHttpRequestData
GetHttpTimeString	GetK2ServerDocCount	GetK2ServerDocCountLimit
GetLocale	GetLocaleDisplayName	GetLocalHostIP
GetMetaData	GetMetricData	GetPageContext
GetPrinterInfo	GetProfileSections	GetProfileString
GetReadableImageFormats	GetSOAPRequest	GetSOAPRequestHeader
GetSOAPResponse	GetSOAPResponseHeader	GetTempDirectory
GetTempFile	GetTemplatePath	GetTickCount
GetTimeZoneInfo	GetToken	GetUserRoles
GetWriteableImageFormats	Hash	Hour
HTMLCodeFormat	HTMLEditFormat	IIf
ImageAddBorder	ImageBlur	ImageClearRect
ImageCopy	ImageCrop	ImageDrawArc
ImageDrawBeveledRect	ImageDrawCubicCurve	ImageDrawLine
ImageDrawLines	ImageDrawOval	ImageDrawPoint
ImageDrawQuadraticCurve	ImageDrawRect	ImageDrawRoundRect

ImageDrawText	ImageFlip	ImageGetBlob
ImageGetBufferedImage	ImageGetEXIFTag	ImageGetHeight
ImageGetIPTCTag	ImageGetWidth	ImageGrayscale
ImageInfo	ImageNegative	ImageNew
ImageOverlay	ImagePaste	ImageRead
ImageReadBase64	ImageResize	ImageRotate
ImageRotateDrawingAxis	ImageScaleToFit	ImageSetAntialiasing
ImageSetBackgroundColor	ImageSetDrawingColor	ImageSetDrawingStroke
ImageSetDrawingTranspar-ency	ImageSharpen	ImageShear
ImageShearDrawingAxis	ImageTranslate	ImageTranslateDrawingAxis
ImageWrite	ImageWriteBase64	ImageXORDrawingMode
IncrementValue	InputBaseN	Insert
Int	IsArray	IsBinary
IsBoolean	IsCustomFunction	IsDate
IsDDX	IsDebugMode	IsDefined
IsImage	IsImageFile	IsInstanceOf
IsJSON	IsLeapYear	IsLocalHost
IsNumeric	IsNumericDate	IsObject
IsPDFFile	IsPDFObject	IsQuery
IsSimpleValue	IsSOAPRequest	IsStruct
IsUserInAnyRole	IsUserInRole	IsUserLoggedIn
IsValid	IsWDDX	IsXML
IsXmlAttribute	IsXmlDoc	IsXmlElem
IsXmlNode	IsXmlRoot	JavaCast
JSStringFormat	LCase	Left
Len	ListAppend	ListChangeDelims
ListContains	ListContainsNoCase	ListDeleteAt
ListFind	ListFindNoCase	ListFirst
ListGetAt	ListInsertAt	ListLast
ListLen	ListPrepend	ListQualify
ListRest	ListSetAt	ListSort
ListToArray	ListValueCount	ListValueCountNoCase
LJustify	Log	Log10
LSCurrencyFormat	LSDateFormat	LSEuroCurrencyFormat

LSIsCurrency	LSIsDate	LSIsNumeric
LSNumberFormat	LSParseCurrency	LSParseDateTime
LSParseEuroCurrency	LSParseNumber	LSTimeFormat
LTrim	Maxfilename	Mid
Min	Minute	Month
MonthAsString	Now	NumberFormat
ParagraphFormat	ParseDateTime	Pi
PrecisionEvaluate	PreserveSingleQuotes	Quarter
QueryAddColumn	QueryAddRow	QueryConvertForGrid
QueryNew	QuerySetCell	QuotedValueList
Rand	Randomize	RandRange
REFind	REFindNoCase	ReleaseComObject
REMatch	REMatchNoCase	RemoveChars
RepeatString	Replace	ReplaceList
ReplaceNoCase	REReplace	REReplaceNoCase
Reverse	Right	RJustify
Round	RTrim	Second
SendGatewayMessage	SerializeJSON	SetEncoding
SetLocale	SetProfileString	SetVariable
Sgn	Sin	Sleep
SpanExcluding	SpanIncluding	Sqr
StripCR	StructAppend	StructClear
StructCopy	StructCount	StructDelete
StructFind	StructFindKey	StructFindValue
StructGet	StructInsert	StructIsEmpty
StructKeyArray	StructKeyExists	StructKeyList
StructKeyList	StructNew	StructSort
StructUpdate	Tan	TimeFormat
ToBase64	ToBinary	ToScript
ToString	Trim	UCase
URLDecode	URLEncodedFormat	URLSessionFormat
Val	ValueList	VerifyClient
Week	Wrap	Wrap
WriteOutput	XmlChildPos	XmlElemNew

XmlFormat	XmlGetNodeType	XmlNew
XmlParse	XmlSearch	XmlTransform
XmlValidate	Year	YesNoFormat

Functions by category

The following tables list functions by their category or purpose.

Array functions

ArrayAppend	ArrayIsDefined	ArrayNew	ArraySum
ArrayAvg	ArrayIsEmpty	ArrayPrepend	ArraySwap
ArrayClear	ArrayLen	ArrayResize	ArrayToList
ArrayDeleteAt	ArrayMax	ArraySet	IsArray
ArrayInsertAt	ArrayMin	ArraySort	ListToArray

Conversion functions

ArrayToList	DotNetToCFType	ToBinary	XmlFormat
BinaryDecode	Hash	ToScript	XmlParse
BinaryEncode	LCase	ToString	XmlTransform
CharsetDecode	ListToArray	URLDecode	
CharsetEncode	SerializeJSON	URLEncodedFormat	
DeserializeJSON	ToBase64	Val	

Date and time functions

CreateDate	DateFormat	GetTimeZoneInfo	MonthAsString
CreateDateTime	DatePart	Hour	Now
CreateODBCDate	Day	IsDate	ParseDateTime
CreateODBCDateTime	DayOfWeek	IsLeapYear	Quarter
CreateODBCTime	DayOfWeekAsString	IsNumericDate	Second
CreateTime	DayOfYear	LSDateFormat	TimeFormat
CreateTimeSpan	DaysInMonth	LSIsDate	Week
DateAdd	DaysInYear	LSParseDateTime	Year
DateCompare	FirstDayOfMonth	LSTimeFormat	
DateConvert	GetHttpTimeString	Minute	
DateDiff	GetTickCount	Month	

Decision functions

DirectoryExists	IsDefined	IsPDFFile	IsXmlDoc
FileExists	IsInstanceOf	IsPDFObject	IsXmlElem
FileIsEOF	IsJSON	IsQuery	IsXmlNode
IIf	IsK2ServerABroker	IsSimpleValue	IsXmlRoot
IsArray	IsK2ServerDocCountExceeded	IsStruct	LSIsCurrency
IsBinary	IsK2ServerOnline	IsUserInAnyRole	LSIsDate
IsBoolean	IsLeapYear	IsValid	LSIsNumeric
IsCustomFunction	IsNumeric	IsWDDX	StructIsEmpty
IsDate	IsNumericDate	IsXML	StructKeyExists
IsDebugMode	IsObject	IsXmlAttribute	YesNoFormat
IsDDX			

Display and formatting functions

AjaxLink	GetLocaleDisplayName	LSIsDate	ParagraphFormat
AjaxOnLoad	HTMLCodeFormat	LSNumberFormat	RJustify
CJustify	HTMLEditFormat	LSParseCurrency	StripCR
DateFormat	LJustify	LSParseDateTime	TimeFormat
DecimalFormat	LSCurrencyFormat	LSParseEuroCurrency	YesNoFormat
DollarFormat	LSDateFormat	LSParseNumber	
FormatBaseN	LSEuroCurrencyFormat	LSTimeFormat	
GetLocale	LSIsCurrency	NumberFormat	

Dynamic evaluation functions

DE	IIf	SetVariable	
Evaluate	PrecisionEvaluate		

Extensibility functions

CreateObject	GetGatewayHelper	SendGatewayMessage
DotNetToCFType	IsInstanceOf	ToScript
GetComponentMetaData	ReleaseComObject	

Full-text search functions

History

ColdFusion MX 6.1: These functions are deprecated. They might not work, and might cause errors, in a future release.

GetK2ServerDocCount	IsK2ServerABroker	IsK2ServerOnline
GetK2ServerDocCountLimit	IsK2ServerDocCountExceeded	

Image functions

ImageAddBorder	ImageGetBlob	ImageScaleToFit
ImageBlur	ImageGetBufferedImage	ImageSetAntialiasing
ImageClearRect	ImageGetEXIFTag	ImageSetBackgroundColor
ImageCopy	ImageGetHeight	ImageSetDrawingColor
ImageCrop	ImageGetIPTCTag	ImageSetDrawingStroke
ImageDrawArc	ImageGetWidth	ImageSetDrawingTransparency
ImageDrawBeveledRect	ImageGrayscale	ImageSharpen
ImageDrawCubicCurve	ImageInfo	ImageShear
ImageDrawLine	ImageNegative	ImageShearDrawingAxis
ImageDrawLines	ImageNew	ImageTranslate
ImageDrawOval	ImageOverlay	ImageTranslateDrawingAxis
ImageDrawPoint	ImagePaste	ImageWrite
ImageDrawQuadraticCurve	ImageRead	ImageWriteBase64
ImageDrawRect	ImageReadBase64	ImageXORDrawingMode
ImageDrawRoundRect	ImageResize	IsImage
ImageDrawText	ImageRotate	IsImageFile
ImageFlip	ImageRotateDrawingAxis	

International functions

DateConvert	LSIsCurrency	LSIsNumeric	SetEncoding
GetEncoding	LSCurrencyFormat	LSNumberFormat	SetLocale
GetHttpTimeString	LSDateFormat	LSParseCurrency	
GetLocale	LSEuroCurrencyFormat	LSParseEuroCurrency	
GetLocaleDisplayName	LSIsDate	LSParseNumber	
GetTimeZoneInfo	LSParseDateTime	LSTimeFormat	

List functions

ArraySort	FindOneOf	ListFind	ListSort
ArrayToList	FormatBaseN	ListFindNoCase	ListToArray
Asc	GetClientVariablesList	ListFirst	ListValueCount
Chr	LCase	ListGetAt	ListValueCountNoCase
CJustify	Left	ListInsertAt	ReplaceList
Compare	Len	ListLast	ValueList
CompareNoCase	ListAppend	ListLen	
Decrypt	ListChangeDelims	ListPrepend	
Encrypt	ListContains	ListQualify	
Find	ListContainsNoCase	ListRest	
FindNoCase	ListDeleteAt	ListSetAt	

Mathematical functions

Abs	BitNot	FormatBaseN	Rand
ACos	BitOr	IncrementValue	Randomize
ArrayAvg	BitSHLN	InputBaseN	RandRange
ArraySum	BitSHRN	Int	Round
ASin	BitXor	Log	Sgn
Atn	Ceiling	Log10	Sin
BitAnd	Cos	Maxfilename	Sqr
BitMaskClear	DecrementValue	Min	Tan
BitMaskRead	Exp	Pi	
BitMaskSet	Fix	PrecisionEvaluate	

Other functions

CreateUUID	GetBaseTemplatePath	PreserveSingleQuotes
DeleteClientVariable	GetClientVariablesList	URLSessionFormat
GetBaseTagData	GetLocalHostIP	WriteOutput
GetBaseTagList	IsLocalHost	

Query functions

IsQuery	QueryAddRow	QueryNew	QuotedValueList
QueryAddColumn	QueryConvertForGrid	QuerySetCell	ValueList

Security functions

Decrypt	GetAuthUser	GetUserRoles	IsUserLoggedIn
DecryptBinary	GenerateSecretKey	Hash	
Encrypt	GetTempDirectory	IsUserInAnyRole	
EncryptBinary	GetTempDirectory	IsUserInRole	

String functions

History

ColdFusion MX: ColdFusion now supports the Java UCS-2 representation of Unicode character values 0–65535. (Earlier releases supported ASCII values.)

String-processing functions process any of these characters (including ASCII 0 (NUL) characters), and continue counting subsequent characters of the string, if any. (In earlier releases, some string-processing functions processed the ASCII 0 (NUL) character, but did not process subsequent characters of the string.)

Asc	HTMLEditFormat	ParagraphFormat	ToBase64
BinaryDecode	Insert	ParseDateTime	ToBinary
BinaryEncode	JavaCast	REFind	ToString
CharsetDecode	JSStringFormat	REFindNoCase	Trim
CharsetEncode	LCase	REMatch	UCase
Chr	Left	REMatchNoCase	URLDecode
CJustify	Len	RemoveChars	URLEncodedFormat
Compare	LJustify	RepeatString	Val
CompareNoCase	ListValueCount	Replace	Wrap
DayOfWeekAsString	LSParseNumber	RTrim	XmlFormat
Decrypt	LTrim	SpanExcluding	
Encrypt	ListValueCountNoCase	ReplaceNoCase	
Find	LSIsDate	REReplace	

FindNoCase	LSIsNumeric	REReplaceNoCase	
FindOneOf	LSParseCurrency	ReplaceList	
FormatBaseN	LSIsCurrency	Reverse	
GenerateSecretKey	LSParseDateTime	Right	
GetToken	LSParseEuroCurrency	RJustify	
Hash	Mid	SpanIncluding	
HTMLCodeFormat	MonthAsString	StripCR	

See also "Conversion functions" on page 641.

Structure functions

Duplicate	StructCount	StructGet	StructKeyList
IsStruct	StructDelete	StructInsert	StructNew
StructAppend	StructFind	StructIsEmpty	StructSort
StructClear	StructFindKey	StructKeyArray	StructUpdate
StructCopy	StructFindValue	StructKeyExists	

System functions

DirectoryExists	FileWrite	GetPageContext
Duplicate	GetBaseTemplatePath	GetPrinterInfo
ExpandPath	GetContextRoot	GetProfileSections
FileClose	GetCurrentTemplatePath	GetProfileString
FileCopy	GetDirectoryFromPath	GetReadableImageFormats
FileDelete	GetDirectoryFromPath	GetTempDirectory
FileExists	GetEncoding	GetTempFile
FileIsEOF	GetException	GetTemplatePath
FileMove	GetFileFromPath	GetTickCount
FileOpen	GetFileInfo	GetWriteableImageFormats
FileRead	GetFunctionList	SetEncoding
FileReadBinary	GetHttpRequestData	SetLocale
FileReadLine	GetLocale	SetProfileString
FileSetAccessMode	GetLocaleDisplayName	Sleep
FileSetAttribute	GetMetaData	WriteOutput
FileSetLastModified	GetMetricData	

XML functions

AddSOAPRequestHeader	IsSOAPRequest	IsXmlRoot	XmlGetNodeType
AddSOAPResponseHeader	IsXML	IsWDDX	XmlNew
GetSOAPRequest	IsXmlAttribute	ToString	XmlParse
GetSOAPRequestHeader	IsXmlDoc	XmlChildPos	XmlSearch
GetSOAPResponse	IsXmlElem	XmlElemNew	XmlTransform
GetSOAPResponseHeader	IsXmlNode	XmlFormat	XmlValidate

Function changes since ColdFusion 5

The following tables list functions, parameters and values that have changed since ColdFusion 5 and indicate the specific release in which the change was made.

New functions, parameters, and values

Function	Parameter or value	Added in this ColdFusion release
AjaxLink	All	ColdFusion 8
AjaxOnLoad	All	ColdFusion 8
ArrayIsDefined	All	ColdFusion 8
BinaryDecode	All	ColdFusion MX 7
BinaryEncode	All	ColdFusion MX 7
CharsetDecode	All	ColdFusion MX 7
CharsetEncode	All	ColdFusion MX 7
CreateObject	.net value of the type parameter and associated assembly, server, port, protocol, and secure parameters. WSDL2Java and argStruct parameters for web service objects	ColdFusion 8
	portName parameter	ColdFusion MX 7
	All	ColdFusion MX
DateAdd	l key of datepart parameter	ColdFusion MX 6.1
DatePart	l key of datepart parameter	ColdFusion MX 6.1
Decrypt	IVorSalt and iterations parameters	ColdFusion MX 7.0.1
	algorithm and encoding parameters	ColdFusion MX 7
DecryptBinary	All	ColdFusion MX 7.0.1
DeserializeJSON	All	ColdFusion 8
DotNetToCFType	All	ColdFusion 8
Encrypt	IVorSalt and iterations parameters	ColdFusion MX 7.0.1
	algorithm and encoding parameters	ColdFusion MX 7
EncryptBinary	All	ColdFusion MX 7.0.1
FileClose	All	ColdFusion 8
FileCopy	All	ColdFusion 8
FileDelete	All	ColdFusion 8

Function	Parameter or value	Added in this ColdFusion release
FileIsEOF	All	ColdFusion 8
FileMove	All	ColdFusion 8
FileOpen	All	ColdFusion 8
FileRead	All	ColdFusion 8
FileReadBinary	All	ColdFusion 8
FileReadLine	All	ColdFusion 8
FileSetAccessMode	All	ColdFusion 8
FileSetAttribute	All	ColdFusion 8
FileSetLastModified	All	ColdFusion 8
FileWrite	All	ColdFusion 8
GenerateSecretKey	All	ColdFusion MX 7
GetGatewayHelper	All	ColdFusion MX 7
GetAuthUser	All	ColdFusion MX
GetComponentMetaData	All	ColdFusion 8
GetContextRoot	All	ColdFusion MX 7
GetEncoding	All	ColdFusion MX
GetFileInfo	All	ColdFusion 8
GetLocaleDisplayName	All	ColdFusion MX 7
GetLocalHostIP	All	ColdFusion MX 7.0.1
GetMetaData	All	ColdFusion MX
GetPageContext	All	ColdFusion MX
GetPrinterInfo	All	ColdFusion 8
GetProfileSections	All	ColdFusion MX
GetReadableImageFormats	All	ColdFusion 8
GetSOAPRequest	All	ColdFusion MX 7
GetSOAPRequestHeader	All	ColdFusion MX 7
GetSOAPResponse	All	ColdFusion MX 7
GetSOAPResponseHeader	All	ColdFusion MX 7
GetUserRoles	All	ColdFusion 8
GetWriteableImageFormats	All	ColdFusion 8
Hash	`algorithm` and `encoding` parameters	ColdFusion MX 7
ImageAddBorder	All	ColdFusion 8
ImageBlur	All	ColdFusion 8

Function	Parameter or value	Added in this ColdFusion release
ImageClearRect	All	ColdFusion 8
ImageCopy	All	ColdFusion 8
ImageCrop	All	ColdFusion 8
ImageDrawArc	All	ColdFusion 8
ImageDrawBeveledRect	All	ColdFusion 8
ImageDrawCubicCurve	All	ColdFusion 8
ImageDrawPoint	All	ColdFusion 8
ImageDrawLine	All	ColdFusion 8
ImageDrawLines	All	ColdFusion 8
ImageDrawOval	All	ColdFusion 8
ImageDrawQuadraticCurve	All	ColdFusion 8
ImageDrawRect	All	ColdFusion 8
ImageDrawRoundRect	All	ColdFusion 8
ImageDrawText	All	ColdFusion 8
ImageFlip	All	ColdFusion 8
ImageGetBlob	All	ColdFusion 8
ImageGetBufferedImage	All	ColdFusion 8
ImageGetEXIFMetadata	All	ColdFusion 8
ImageGetEXIFTag	All	ColdFusion 8
ImageGetHeight	All	ColdFusion 8
ImageGetIPTCMetadata	All	ColdFusion 8
ImageGetIPTCTag	All	ColdFusion 8
ImageGetWidth	All	ColdFusion 8
ImageGrayscale	All	ColdFusion 8
ImageInfo	All	ColdFusion 8
ImageNegative	All	ColdFusion 8
ImageNew	All	ColdFusion 8
ImageOverlay	All	ColdFusion 8
ImagePaste	All	ColdFusion 8
ImageRead	All	ColdFusion 8
ImageReadBase64	All	ColdFusion 8
ImageResize	All	ColdFusion 8
ImageRotate	All	ColdFusion 8

Function	Parameter or value	Added in this ColdFusion release
ImageRotateDrawingAxis	All	ColdFusion 8
ImageScaleToFit	All	ColdFusion 8
ImageSetAntialiasing	All	ColdFusion 8
ImageSetBackgroundColor	All	ColdFusion 8
ImageSetDrawingColor	All	ColdFusion 8
ImageSetDrawingStroke	All	ColdFusion 8
ImageSetDrawingTransparency	All	ColdFusion 8
ImageSharpen	All	ColdFusion 8
ImageShear	All	ColdFusion 8
ImageShearDrawingAxis	All	ColdFusion 8
ImageTranslate	All	ColdFusion 8
ImageTranslateDrawingAxis	All	ColdFusion 8
ImageWrite	All	ColdFusion 8
ImageWriteBase64	All	ColdFusion 8
ImageXORDrawingMode	All	ColdFusion 8
IsDDX	All	ColdFusion 8
IsImage	All	ColdFusion 8
IsImageFile	All	ColdFusion 8
IsInstanceOf	All	ColdFusion 8
IsJSON	All	ColdFusion 8
IsLocalHost	All	ColdFusion MX 7.0.1
IsObject	All	ColdFusion MX
IsPDFFile	All	ColdFusion 8
IsPDFObject	All	ColdFusion 8
IsSOAPRequest	All	ColdFusion MX 7
IsUserInAnyRole	All	ColdFusion 8
IsUserInRole	All	ColdFusion MX
IsUserLoggedIn	All	ColdFusion 8
IsValid	All	ColdFusion MX 7
IsXML	All	ColdFusion MX 7
IsXmlAttribute	All	ColdFusion MX 7
IsXmlDoc	All	ColdFusion MX
IsXmlElem	All	ColdFusion MX

Function	Parameter or value	Added in this ColdFusion release
IsXmlNode	All	ColdFusion MX 7
IsXmlRoot	All	ColdFusion MX
LSTimeFormat	l key of mask parameter	ColdFusion MX 6.1
QueryAddColumn	datatype parameter	ColdFusion MX 7
QueryConvertForGrid	All	ColdFusion 8
QueryNew	columntypelist parameter	ColdFusion MX 7
PrecisionEvaluate	All	ColdFusion 8
Rand	algorithm parameter	ColdFusion MX 7
Randomize	algorithm parameter	ColdFusion MX 7
RandRange	algorithm parameter	ColdFusion MX 7
ReleaseComObject	All	ColdFusion MX 6.1
REMatch	All	ColdFusion 8
REMatchNoCase	All	ColdFusion 8
SerializeJSON	All	ColdFusion 8
SendGatewayMessage	All	ColdFusion MX 7
SetEncoding	All	ColdFusion MX
Sleep	All	ColdFusion 8
TimeFormat	l key of mask parameter	ColdFusion MX 6.1
ToScript	All	ColdFusion MX 7
URLDecode	charset parameter	ColdFusion MX
URLEncodedFormat	charset parameter	ColdFusion MX
URLSessionFormat	All	ColdFusion MX
VerifyClient	All	ColdFusion 8
Wrap	All	ColdFusion MX 6.1
XmlChildPos	All	ColdFusion MX
XmlElemNew	All	ColdFusion MX
XmlElemNew	namespace parameter	ColdFusion MX 7
XmlGetNodeType	All	ColdFusion MX 7
XmlNew	All	ColdFusion MX
XmlParse	All	ColdFusion MX
XmlParse	validator parameter	ColdFusion MX 7
XmlSearch	All	ColdFusion MX

Function	Parameter or value	Added in this ColdFusion release
XmlTransform	All	ColdFusion MX
XmlTransform	parameters parameter	ColdFusion MX 7
XmlValidate	All	ColdFusion MX 7

Deprecated functions, parameters, and values

The following functions, parameters, and values are deprecated. Do not use them in ColdFusion applications. They might not work, and might cause an error, in releases later than ColdFusion MX.

Function	Parameter or value	Deprecated as of this ColdFusion release
GetMetricData	cachepops parameter	ColdFusion MX
GetK2ServerDocCount	All	ColdFusion MX 6.1
GetK2ServerDocCountLimit	All	ColdFusion MX 6.1
GetTemplatePath	All	ColdFusion MX
IsK2ServerABroker	All	ColdFusion MX 6.1
IsK2ServerDocCountExceeded	All	ColdFusion MX 6.1
IsK2ServerOnLine	All	ColdFusion MX 6.1
ParameterExists	All	ColdFusion MX. Use the IsDefined function.
SetLocale	locale = "Spanish (Mexican)" value	ColdFusion MX. Use Spanish (Standard).

Obsolete functions, parameters, and values

The following functions, parameters, and values are obsolete. Do not use them in ColdFusion applications. They do not work in releases later than ColdFusion 5.

Function	Parameter or value	Obsolete as of this ColdFusion release
AuthenticatedContext	All	ColdFusion MX
AuthenticatedUser	All	ColdFusion MX
isAuthenticated	All	ColdFusion MX
isAuthorized	All	ColdFusion MX
isProtected	All	ColdFusion MX

Abs

Description

Absolute-value function. The absolute value of a number is the number without its sign.

Returns

The absolute value of a number.

Category

Mathematical functions

Function syntax

Abs(number)

See also

Sgn

Parameters

Parameter	Description
number	A number

Example

```
<h3>Abs Example</h3>
<p>The absolute value of the following numbers:
1,3,-4,-3.2,6 is
<cfoutput>
#Abs(1)#,#Abs(3)#,#Abs(-4)#,#Abs(-3.2)#,#Abs(6)#
</cfoutput>

<p>The absolute value of a number is the number without its sign.</p>
```

ACos

Description

Arccosine function. The arccosine is the angle whose cosine is *number*.

Returns

The arccosine, in radians, of a number.

Category

Mathematical functions

Function syntax

`ACos(number)`

See also

`Cos`, `Sin`, `ASin`, `Tan`, `Atn`, `Pi`

Parameters

Parameter	Description
`number`	Cosine of an angle. The value must be between -1.0 and 1.0, inclusive.

Usage

The range of the result is 0 to π.

To convert degrees to radians, multiply degrees by π/180. To convert radians to degrees, multiply radians by 180/π.

Example

```
<h3>ACos Example</h3>
<!--- Output the arccosine value. --->
<cfif IsDefined("FORM.CosNum")>
    <cfif IsNumeric(FORM.CosNum)>
        <cfif Abs(FORM.CosNum) LESS THAN OR EQUAL TO 1>
            <cfoutput>ACos(#FORM.CosNum#) = #ACos(FORM.cosNum)# Radians </cfoutput>
            <br>or<br>
            <cfoutput>ACos(#FORM.CosNum#) = #ACos(FORM.cosNum) * 180/PI()#</cfoutput>
        <cfelse>
            <!--- If it is empty, output an error message. --->
            <h4>Enter a number between -1 and 1</h4>
        </cfif>
    </cfif>
</cfif>

<form method="post" action = "acos.cfm">
    <p>Enter a number to get its arccosine in Radians and Degrees.
    <br><input type = "Text" name = "cosNum" size = "25">
    <p><input type = "Submit" name = ""> <input type = "RESET">
</form>
```

AddSOAPRequestHeader

Description
Adds a SOAP header to a web service request before making the request.

Returns
Nothing.

Category
XML functions

Function syntax
AddSOAPRequestHeader(webservice, namespace, name, value [, mustunderstand])

See also
AddSOAPResponseHeader, GetSOAPRequest, GetSOAPRequestHeader, GetSOAPResponse, GetSOAPResponse-Header, IsSOAPRequest; "Basic web service concepts" on page 903 in the *ColdFusion Developer's Guide*

History
ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
webservice	A web service object as returned from the cfobject tag or the CreateObject function.
namespace	A string that is the namespace for the header.
name	A string that contains the name of the SOAP header in the request.
value	The value for the SOAP header; this can be a CFML XML value.
mustunderstand	Optional. True or False (default). Sets the SOAP mustunderstand value for this header.

Usage
Used within CFML code by a consumer of a web service *before* it calls the web service .

If you pass XML in the value parameter, ColdFusion ignores the namespace and name parameters. If you require a namespace, define it within the XML itself.

Example
There are two parts to this example. The first part is the web service CFC that this function (as well as the other ColdFusion SOAP functions) uses to demonstrate its interaction with a web service. To implement the web service for this function, see the example for AddSOAPResponseHeader.

Execute the following example as a client to see how the AddSOAPRequestHeader function operates.

```
<!--- Note that you might need to modify the URL in the CreateObject function
    to match your server and the location of the headerservice.cfc file if it is
    different than shown here. Likewise for the cfinvoke tag at the end. --->

<h3>AddSOAPRequestHeader Example</h3>
<cfscript>
 // Create the web service object.
 ws = CreateObject("webservice", "http://localhost/soapheaders/headerservice.cfc?WSDL");
```

```
 // Set the username header as a string.
 addSOAPRequestHeader(ws, "http://mynamespace/", "username", "tom", false);

 // Set the password header as a CFML XML object.
 doc = XmlNew();
 doc.password = XmlElemNew(doc, "http://mynamespace/", "password");
 doc.password.XmlText = "My Voice is my Password";
 doc.password.XmlAttributes["xsi:type"] = "xsd:string";
 addSOAPRequestHeader(ws, "ignoredNameSpace", "ignoredName", doc);

 // Invoke the web service operation.
 ret = ws.echo_me("argument");

 // Get the first header as an object (string) and as XML.
 header = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader");
 XMLheader = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader", true);

 // Get the second header as an object (string) and as XML.
 header2 =getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader2");
 XMLheader2 = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader2", true);
</cfscript>
<hr>
<cfoutput>
    Soap Header value: #HTMLCodeFormat(header)#<br>
    Soap Header XML value: #HTMLCodeFormat(XMLheader)#<br>
    Soap Header 2 value: #HTMLCodeFormat(header2)#<br>
    Soap Header 2 XML value: #HTMLCodeFormat(XMLheader2)#<br>
    Return value: #HTMLCodeFormat(ret)#<br>
</cfoutput>
<hr>

<cfinvoke component="soapheaders.headerservice" method="echo_me" returnvariable="ret"
in_here="hi">
</cfinvoke>
<cfoutput>The cfinvoke tag returned: #ret#</cfoutput>
```

AddSOAPResponseHeader

Description

Adds a SOAP response header to a web service response. Call only from within a CFC web service function that is processing a request as a SOAP web service.

Returns

Nothing

Category

XML functions

Function syntax

`AddSOAPResponseHeader(namespace, name, value [, mustunderstand])`

See also

`AddSOAPRequestHeader`, `GetSOAPRequest`, `GetSOAPRequestHeader`, `GetSOAPResponse`, `GetSOAPResponse-Header`, `IsSOAPRequest`; "Basic web service concepts" on page 903 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`namespace`	A string that is the namespace for the header.
`name`	A string that contains the name of the SOAP header in the request.
`value`	The value for the SOAP header; this can be a CFML XML value.
`mustunderstand`	Optional. True or False (default). Sets the SOAP mustunderstand value for this header.

Usage

Call this function only from within a CFC web service function. It throws an error if it is invoked in a context that is not a web service request.

If you pass XML in the `value` parameter, ColdFusion ignores the namespace and name parameters. If you require a namespace, define it within the XML itself.

Use the `IsSOAPRequest` function to determine if the CFC is running as a web service.

Example

This example creates a CFC web service that illustrates the operation of the `AddSOAPResponseHeader` function and also provides a web service that illustrates the operation of other ColdFusion SOAP functions.

Save the following code as headerservice.cfc in a folder called soapheaders under your web root. Test its operation, and specifically the operation of the `AddSOAPResponseHeader` function, by executing the examples that invoke this web service. For example, see the example for `AddSOAPRequestHeader`.

```
<h3>AddSOAPResponseHeader Example</h3>
<!--- The headerservice.cfc CFC Web Service.--->
<cfcomponent displayName="tester" hint="Test for SOAP headers">
<cffunction name="echo_me"
            access="remote"
```

```
            output="false"
            returntype="string"
            displayname="Echo Test" hint="Header test">

<cfargument name="in_here" required="true" type="string">

<cfset isSOAP = isSOAPRequest()>
<cfif isSOAP>
    <!--- Get the first header as a string and as XML. --->
    <cfset username = getSOAPRequestHeader("http://mynamespace/", "username")>
    <cfset return = "The service saw username: " & username>
    <cfset xmlusername = getSOAPRequestHeader("http://mynamespace/", "username", "TRUE")>
    <cfset return = return & "<br> as XML: " & xmlusername>

    <!--- Get the second header as a string and as XML. --->
    <cfset password = getSOAPRequestHeader("http://mynamespace/", "password")>
    <cfset return = return & "The service saw password: " & password>
    <cfset xmlpassword = getSOAPRequestHeader("http://mynamespace/", "password", "TRUE")>
    <cfset return = return & "<br> as XML: " & xmlpassword>

    <!--- Add a header as a string. --->
    <cfset addSOAPResponseHeader("http://www.tomj.org/myns",
        "returnheader", "AUTHORIZED VALUE", false)>

    <!--- Add a second header using a CFML XML value. --->
    <cfset doc = XmlNew()>
    <cfset x = XmlElemNew(doc, "http://www.tomj.org/myns", "returnheader2")>
    <cfset x.XmlText = "hey man, here I am in XML">
    <cfsetx.XmlAttributes["xsi:type"] = "xsd:string">
    <cfset tmp = addSOAPResponseHeader("ignoredNameSpace", "ignoredName", x)>
<cfelse>
    <!--- Add a header as a string - Must generate error!
    <cfset addSOAPResponseHeader("http://www.tomj.org/myns",
        "returnheader", "AUTHORIZED VALUE", false)>
    --->
    <cfset return = "Not invoked as a web service">
</cfif>

<cfreturn return>
</cffunction>
</cfcomponent>
```

AjaxLink

Description

Causes an HTML `href` attribute to display link results in the current Ajax container. When the browser follows a link that is specified by this function, the HTTP response does not replace the current page; instead, it populates the containing `cfdiv`, `cflayoutarea`, `cfpod`, or `cfwindow` control.

Returns

Code that causes the linked page to be displayed in the containing control.

Category

Display and formatting functions

Function syntax

`AjaxLink(URL)`

See also

`cfdiv`, `cflayoutarea`, `cfpod`, `cfwindow`, "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
URL	The URL of the link.

Usage

This function has an effect only when it is used to specify the URL of an `href` attribute when the HTML `a` tag is inside a `cfdiv`, `cflayoutarea`, `cfpod`, or `cfwindow` control. Otherwise, the link has its normal effect.

To prevent cross-site scripting, ColdFusion does not load a remote web page.

Example

```
<cfpod height="600" width="600" name="podTest">
    <a href="<cfoutput>#AjaxLink('HelloWorld.cfm')#</cfoutput>">Click me</a>
</cfpod>
```

AjaxOnLoad

Description
Causes the specified JavaScript function to run when the page loads.

Returns
This function does not return a value.

Category
Display and formatting functions

Function syntax
`AjaxOnLoad(functionName)`

See also
`cfdiv`, `cflayoutarea`, `cfpod`, `cfwindow`, "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`functionName`	The name of the function to run when the page loads. The specified function does not take a parameter.

Usage
This function causes a JavaScript function to run when a page loads in the browser. The JavaScript function can perform any initialization actions that are required for the page to function properly. For example, a login window might open on a page if the user is not already logged in. You can use the `AjaxOnLoad` function to specify a JavaScript function that determines the login status and opens the window only if needed.

You can use this function on top-level pages, or on pages that you dynamically include in your application by using the `source` attribute of the `cfpod`, `cfwindow`, `cfpod`, and `cfwindow` tags.

Example
The following example uses the `AjaxOnLoad` function to call an init function each time the page loads. The `init` function displays a login window.

```
<html>
<head>
<title>Enter Mail Login Details</title>

<script>
init = function() {
    ColdFusion.Window.show('loginwindow');
}
</script>
</head>

<body>
<cfwindow name="loginwindow" title="Enter Login Details"
    draggable="false" closable="false" resizable="false"
    width="450" height="200">
```

```
<cfoutput>
<form action="#cgi.script_name#" method="post" name="loginform">
    <table width="400" class="loginTable" cellpadding="5">
        <tr>
            <td style="text-align: right">mail server:</td>
            <td><input type="text" name="server"></td>
        </tr>
        <tr>
            <td style="text-align: right">username:</td>
            <td><input type="text" name="username"></td>
        </tr>
        <tr>
            <td style="text-align: right">password:</td>
            <td><input type="password" name="password"></td>
        </tr>
        <tr>
            <td> </td>
            <td><input type="submit" name="login" value="Login"></td>
        </tr>
    </table>
</form>
</cfoutput>
</cfwindow>

<cfset AjaxOnLoad("init")>
</body>
</html>
```

ArrayAppend

Description
Appends an array element to an array.

Returns
True, on successful completion.

Category
Array functions

Function syntax
`ArrayAppend(array, value)`

See also
`ArrayPrepend`; "Adding elements to an array" on page 73 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
`array`	Name of an array
`value`	Value to add at end of array

Example
```
<h3>ArrayAppend Example</h3>
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
    SELECT FirstName, LastName FROM Employees
</cfquery>
<!--- Create an array> --->
<cfset myArray = ArrayNew(1)>
<!--- Set element one to show where we are. --->
<cfset myArray[1] = "Test Value">
<!--- Loop through the query; append these names successively to the last
    element. --->
<cfloop query = "GetEmployeeNames">
    <cfoutput>#ArrayAppend(myArray, "#FirstName# #LastName#")#
    </cfoutput>, Array was appended<br>
</cfloop>
<!--- Show the resulting array as a list. --->
<cfset myList = ArrayToList(myArray, ",")>
<!--- Output the array as a list. --->
<cfoutput>
    <p>The contents of the array are as follows:</p>
    <p>#myList#</p>
</cfoutput>
```

ArrayAvg

Description
Calculates the average of the values in an array.

Returns
Number. If the `array` parameter value is an empty array, returns zero.

Category
Array functions, Mathematical functions

Function syntax
`ArrayAvg(array)`

See also
`ArraySum`; "Basic array techniques" on page 70 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
array	Name of an array

Usage
The following example uses the ColdFusion built-in variable `Form.fieldNames`, which is available on the action page of a form. It contains a comma-delimited list of the names of the fields on the form.

Example
```
<!--- This example shows the use of ArrayAvg. --->
<!-- The following lines of code keep track of the form fields that can
be dynamically generated on the screen. It uses the Fieldnames variable
with the ListLen function to determine the number of fields on the form. --->
<cfset FormElem = 2>
    <cfif Isdefined("Form.Submit")>
        <cfif Form.Submit is "More">
            <cfset FormElem = ListLen(Form.Fieldnames)>
        </cfif>
    </cfif>

<html>
<head>
<title>ArrayAvg Example</title>
</head>
<body>
<h3>ArrayAvg Example</h3>
<p> This example uses ArrayAvg to find the average of the numbers that you enter
    into an array.<br>
To enter more than two numbers click the <b>more</b> button.
</p>
<form action = "arrayavg.cfm">
<!--- The following code initially creates two fields. It adds fields if the
    user presses MORE. FormElem is initialized to two at the beginning of this
    code to show that the form has two fields. ----->
<input type = "submit" name = "submit" value = "more">
<table cellspacing = "2" cellpadding = "2" border = "0">
<cfloop index = "LoopItem" from = "1" to = "#FormElem#">
```

```
    <tr>
        <cfoutput>
        <th align = "left">Number #LoopItem#</th>
        <td><input type = "text" name = "number#LoopItem#"></td>
        </cfoutput>
    </tr>
</cfloop>
</table>
<input type = "submit" name = "submit" value = "get the average">
</form>

<!--- Create an array. --->
<cfif IsDefined("FORM.submit")>
    <cfset myNumberArray = ArrayNew(1)>
    <cfset Count = 1>
    <cfloop index = "ListItem" list = "#Form.Fieldnames#">
        <cfif Left(ListItem,3) is "Num">
            <cfset myNumberArray[Count] = Val("number#Count#")>
            <cfset count = IncrementValue(Count)>
        </cfif>
    </cfloop>

    <cfif Form.Submit is "get the average">
        <!--- use ArrayAvg to get the average of the two numbers --->
        <p>The average of the numbers that you entered is
        <cfoutput>#ArrayAvg(myNumberArray)#.</cfoutput>
    <cfelse>
        <cfoutput>Try again. You must enter at least two numeric values.
        </cfoutput>
    </cfif>
</cfif>
</body>
</html>
```

ArrayClear

Description
Deletes the data in an array.

Returns
True, on successful completion.

Category
Array functions

Function syntax
`ArrayClear(array)`

See also
`ArrayDeleteAt`; "Functions for XML object management" on page 879 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: This function can be used on XML objects.

Parameters

Parameter	Description
`array`	Name of an array

Example
```
<h3>ArrayClear Example</h3>
<!--- Create a new array. --->
<cfset MyArray = ArrayNew(1)>
<!--- Populate an element or two. --->
<cfset MyArray[1] = "Test">
<cfset MyArray[2] = "Other Test">
<!--- Output the contents of the array. --->
<p>Your array contents are:
<cfoutput>#ArrayToList(MyArray)#</cfoutput>
<!--- Check to see if the array is empty. --->
<p>Is the array empty?:
<cfoutput>#ArrayIsEmpty(MyArray)#</cfoutput>
<p>Now, clear the array:
<!--- Now clear the array. --->
<cfset Temp = ArrayClear(MyArray)>
<!--- Check to see if the array is empty. --->
<p>Is the array empty?:
<cfoutput>#ArrayIsEmpty(MyArray)#</cfoutput>
```

ArrayDeleteAt

Description

Deletes an element from an array.

When an element is deleted, ColdFusion recalculates index positions. For example, in an array that contains the months of the year, deleting the element at position 5 removes the entry for May. After this, to delete the entry for June, you would delete the element at position 5 (not 6).

Returns

True, on successful completion.

Category

Array functions

Function syntax

`ArrayDeleteAt(array, position)`

See also

`ArrayInsertAt`; "Functions for XML object management" on page 879 in the *ColdFusion Developer's Guide*

History

ColdFusion MX:

- Changed behavior: This function can be used on XML objects.
- Changed thrown exceptions: This function can throw the InvalidArrayIndexException error.

Parameters

Parameter	Description
`array`	Name of an array
`position`	Array position

Throws

If this function attempts to delete an element at position 0, or specifies a value for `position` that is greater than the size of `array`, this function throws an InvalidArrayIndexException error.

Example

```
<h3>ArrayDeleteAt Example</h3><p>
<!--- Create an array. --->
<cfset DaysArray = ArrayNew(2)>
<!--- Populate an element or two. --->
<cfset DaysArray[1][1] = "Monday">
<cfset DaysArray[2][1] = "Tuesday">
<cfset DaysArray[3][1] = "Wednesday">
<cfset DaysArray[1][2] = "April 12">
<cfset DaysArray[2][2] = "April 13">
<cfset DaysArray[3][2] = "April 14">
<p>This is what the array looks like before delete:<br>
<cfoutput>
#DaysArray[1][1]#  #DaysArray[1][2]#<br>
#DaysArray[2][1]#  #DaysArray[2][2]#<br>
#DaysArray[3][1]#  #DaysArray[3][2]#<br>
```

```
</cfoutput>

<cfoutput>
We delete this element of the array:<br>
#ArrayDeleteAt(DaysArray,2)#<br>
</cfoutput>
<!--- The formerly third element, "Wednesday" is now the second element. --->
<p>This is what the array looks like after delete:<br>
<cfoutput>
#DaysArray[1][1]#  #DaysArray[1][2]#<br>
#DaysArray[2][1]#  #DaysArray[2][2]#<br>
</cfoutput>
```

ArrayInsertAt

Description

Inserts a value into an array. Array elements whose indexes are equal to or greater than the new position are incremented by one. The array length increases by one.

Returns

True, on successful completion.

Category

Array functions

Function syntax

`ArrayInsertAt(array, position, value)`

See also

`ArrayDeleteAt`; "Functions for XML object management" on page 879 in the *ColdFusion Developer's Guide*

History

ColdFusion MX:

- Changed behavior: This function can be used on XML objects.
- Changed thrown exceptions: This function can throw the InvalidArrayIndexException error.

Parameters

Parameter	Description
`array`	Name of an array
`position`	Index position at which to insert value
`value`	Value to insert

Usage

To apply the `ArrayInsertAt()` function to a multidimensional array, you must specify all but the last index in the `array` parameter. The following example inserts an element at `myarray`[2][4]:

```
<cfset ArrayInsertAt(myarray[2], 4, "test")>
```

Throws

If this function attempts to insert an element at position 0, or specifies a value for `position` that is greater than the size of `array`, this function throws an InvalidArrayIndexException error.

Example

```
<h3>ArrayInsertAt Example</h3><p>
<!--- Create a new array. --->
<cfset DaysArray = ArrayNew(1)>
<!--- Populate an element or two. --->
<cfset DaysArray[1] = "Monday">
<cfset DaysArray[2] = "Tuesday">
<cfset DaysArray[3] = "Thursday">
<!--- Add an element before position 3. --->
<p>Add an element before position 3:
    <cfoutput>#ArrayInsertAt(DaysArray,3,"Wednesday")#</cfoutput>
```

```
<p>Now output the array as a list:
<cfoutput>#ArrayToList(DaysArray)#</cfoutput>
<!--- The array now has four elements. Element 3, "Thursday", has become element four. --->
```

ArrayIsDefined

Description
Determines whether an array element is defined.

Returns
True, if the array element is defined (exists); false, otherwise.

Category
Array functions

Function syntax
```
ArrayIsDefiend(array, elementIndex)
```

See also
ArrayIsEmpty

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
array	Name of a one dimensional array, or the array name and indexes into higher-order dimensions of a multidimensional array.
elementIndex	Index of the element in a one dimensional array, or the index of the element in the final dimension of a multidimensional array.

Usage
To test the existence of an element in a multidimensional array, specify all but the last dimension of the array in the first parameter. For example, the following line tests the existence of element MyArray[2][4][1]:

```
ArrayIsDefined(MyArray[2][4], 1)
```

Example
```
<h3>ArrayIsDefined Example</h3>
<!--- Create a sparse new array. --->
<cfset MyArray = ArrayNew(1)>
<!--- Populate an element or two. --->
<cfset MyArray[1] = "Test">
<cfset MyArray[3] = "Other Test">

<cfoutput>
    <!--- Display the contents of the array. --->
    <p>Your array contents are:
    <cfdump var="#MyArray#"></p>

    <!--- Check if an existing element is defined. --->
    <p>Does element 3 exist?: 
    #ArrayIsDefined(MyArray, 3)#</p>

    <!--- Check if a non-existent element is defined. --->
    <p>Does element 2 exist? 
    #ArrayIsDefined(MyArray, 2)#
</cfoutput>
```

ArrayIsEmpty

Description
Determines whether an array is empty of data elements.

Returns
True, if the array is empty; False, otherwise.

Category
Array functions

Function syntax
`ArrayIsEmpty(array)`

See also
`ArrayIsDefined`, `ArrayLen`, "Functions for XML object management" on page 879 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
`array`	Name of an array

Usage
You can test whether an element of a higher level dimension of a multidimensional array is empty by specifying the element in the `ArrayIsEmpty` function. To test whether the first row of the two-dimensional array MyArray is empty, use the following function:

```
ArrayIsEmpty(MyArray2[1])
```

Example
```
<h3>ArrayIsEmpty Example</h3>
<!--- Create a new array. --->
<cfset MyArray = ArrayNew(1)>
<!--- Populate an element or two. --->
<cfset MyArray[1] = "Test">
<cfset MyArray[2] = "Other Test">
<!--- Output the contents of the array. --->
<p>Your array contents are:
<cfoutput>#ArrayToList(MyArray)#</cfoutput>
<!--- Check to see if the array is empty. --->
<p>Is the array empty?:
<cfoutput>#ArrayIsEmpty(MyArray)#</cfoutput>
<p>Now, clear the array:
<!--- Now clear the array. --->
<cfset Temp = ArrayClear(MyArray)>
<!--- Check to see if the array is empty. --->
<p>Is the array empty?:
<cfoutput>#ArrayIsEmpty(MyArray)#</cfoutput>
```

ArrayLen

Description
Determines the number of elements in an array.

Returns
The number of elements in an array.

Category
Array functions

Function syntax
`ArrayLen(array)`

See also
`ArrayIsEmpty`; "Functions for XML object management" on page 879 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: This function can be used on child XML objects.

Parameters

Parameter	Description
`array`	Name of an array

Example
```
<h3>ArrayLen Example</h3>
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
    SELECT FirstName, LastName FROM Employees
</cfquery>
<!--- Create an array. --->
<cfset myArray = ArrayNew(1)>
<!--- Set element one to show where we are. --->
<cfset myArray[1] = "Test Value">
<!--- Loop through the query and append these names
    successively to the last element. --->
<cfloop query = "GetEmployeeNames">
    <cfset temp = ArrayAppend(myArray, "#FirstName# #LastName#")>
</cfloop>
<!--- Show the resulting array as a list. --->
<cfset myList = ArrayToList(myArray, ",")>
<!--- Output the array as a list. --->
<cfoutput>
    <p>The contents of the array are as follows:</p>
    <p>#myList#</p>
    <p>This array has #ArrayLen(MyArray)# elements.</p>
</cfoutput>
```

ArrayMax

Description

Array maximum function.

Returns

The largest numeric value in an array. If the `array` parameter value is an empty array, returns zero.

Category

Array functions

Function syntax

`ArrayMax(array)`

Parameters

Parameter	Description
`array`	Name of an array

Example

```
<h3>ArrayMax Example</h3>
<p>This example uses ArrayMax to find the largest number in an array.<br> </p>
<!--- After checking whether the form has been submitted, the code creates an array
    and assigns the form fields to the first two elements in the array. ---->
<cfif IsDefined("FORM.submit")>
    <cfset myNumberArray = ArrayNew(1)>
    <cfset myNumberArray[1] = number1>
    <cfset myNumberArray[2] = number2>
    <cfif Form.Submit is "Maximum Value">
        <!--- Use ArrayMax to find the largest number in the array. --->
        <p>The largest number that you entered is
        <cfoutput>#ArrayMax(myNumberArray)#.</cfoutput>
    </cfif>
</cfif>
<!---- The following form provides two numeric fields that are compared when the
    form is submitted. --->
<form action = "arraymax.cfm">
    <input type = "hidden" name = "number1_Float">
    <input type = "hidden" name = "number2_Float">
    <input type = "text" name = "number1"><br>
    <input type = "text" name = "number2"><br>
    <input type = "submit" name = "submit" value = "Maximum Value">
</form>
```

ArrayMin

Description

Array minimum function.

Returns

The smallest numeric value in an array. If the `array` parameter value is an empty array, returns zero.

Category

Array functions

Function syntax

`ArrayMin(array)`

Parameters

Parameter	Description
`array`	Name of an array

Example

```
<h3>ArrayMin Example</h3>
<p>This example uses ArrayMin to find the smallest number in an array.<br></p>
<!--- After checking whether the form has been submitted, this code creates an
    array and assigns the form fields to the first two elements. ----->
<cfif IsDefined("FORM.submit")>
    <cfset myNumberArray = ArrayNew(1)>
    <cfset myNumberArray[1] = FORM.number1>
    <cfset myNumberArray[2] = FORM.number2>
    <cfif Form.Submit is "Minimum Value">
        <!--- Use ArrayMin to find the smallest number in the array. --->
        <p>The smallest number that you entered is
        <cfoutput>#ArrayMin(myNumberArray)#.</cfoutput>
    </cfif>
</cfif>
<!---- The following form provides two numeric fields that are compared when the form is
    submitted. ----->
<form action = "arraymin.cfm">
    <input type = "hidden" name = "number1_Float">
    <input type = "hidden" name = "number2_Float">
    <input type = "text" name = "number1"><br>
    <input type = "text" name = "number2"><br>
    <input type = "submit" name = "submit" value = "Minimum Value">
</form>
```

ArrayNew

Description

Creates an array of 1–3 dimensions. Index array elements with square brackets: [ ].

ColdFusion arrays expand dynamically as data is added.

Returns

An array

Category

Array functions

Function syntax

`ArrayNew(dimension)`

Parameters

Parameter	Description
`dimension`	Number of dimensions in new array: 1, 2, or 3

Example

```
<h3>ArrayNew Example</h3>
<!--- Create an array. --->
<cfset MyNewArray = ArrayNew(1)>
<!--- ArrayToList does not function properly if the Array is not initialized with
    ArraySet --->
<cfset temp = ArraySet(MyNewArray, 1,6, "")>

<!--- Set some elements. --->
<cfset MyNewArray[1] = "Sample Value">
<cfset MyNewArray[3] = "43">
<cfset MyNewArray[6] = "Another Value">

<!--- Is it an array? --->
<cfoutput>
    <p>Is this an array? #IsArray(MyNewArray)#</p>
    <p>It has #ArrayLen(MyNewArray)# elements.</p>
    <p>Contents: #ArrayToList(MyNewArray)#</p>
<!--- The array has expanded dynamically to six elements with the use of ArraySet,
    even though we only set three values. --->
</cfoutput>
```

ArrayPrepend

Description

Inserts an array element at the beginning of an array.

Returns

True, on successful completion.

Category

Array functions

Function syntax

`ArrayPrepend(array, value)`

See also

`ArrayAppend`; "Adding elements to an array" on page 73 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`array`	Name of an array
`value`	Value to insert at beginning of array

Example

```
<h3>ArrayPrepend Example</h3>
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
    SELECT FirstName, LastName FROM Employees
</cfquery>
<!--- Create an array. --->
<cfset myArray = ArrayNew(1)>
<!--- Set element one to show where we are. --->
<cfset myArray[1] = "Test Value">
<!--- Loop through query. Append names successively before last element.
    (The list reverses itself from the standard queried output, becuase it keeps
    prepending the array entry.) --->
<cfloop query = "GetEmployeeNames">
    <cfoutput>#ArrayPrepend(myArray, "#FirstName# #LastName#")#
    </cfoutput>, Array was prepended<br>
</cfloop>
<!--- Show the resulting array as a list. --->
<cfset myList = ArrayToList(myArray, ",")>
<!--- Output the array as a list. --->
<cfoutput>
    <p>The contents of the array are as follows:
    <p>#myList#
</cfoutput>
```

ArrayResize

Description

Resets an array to a specified minimum number of elements. This can improve performance, if used to size an array to its expected maximum. For more than 500 elements, use `ArrayResize` immediately after using the `ArrayNew` tag.

ColdFusion arrays expand dynamically as data is added.

Returns

True, on successful completion.

Category

Array functions

Function syntax

`ArrayResize(array, minimum_size)`

Parameters

Parameter	Description
`array`	Name of an array
`minimum_size`	Minimum array size

Example

```
<h3>ArrayResize Example</h3>
<!--- Perform a query to get the list. --->
<cfquery name = "GetCourses" datasource = "cfdocexamples">
    SELECT * FROM Courses
</cfquery>
<!--- Create a new array. --->
<cfset MyArray = ArrayNew(1)>
<!--- Resize that array to the number of records in the query. --->
<cfset temp = ArrayResize(MyArray, GetCourses.RecordCount)>
<cfoutput>
    The array is now #ArrayLen(MyArray)# elements, to match
    the query of #GetCourses.RecordCount# records.
</cfoutput>
```

ArraySet

Description

In a one-dimensional array, sets the elements in a specified index range to a value. Useful for initializing an array after a call to ArrayNew.

Returns

True, on successful completion.

Category

Array functions

Function syntax

ArraySet(*array*, *start_pos*, *end_pos*, *value*)

See also

ArrayNew; "Populating arrays with data" on page 75 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Changed behavior: This function can be used on XML objects.

Parameters

Parameter	Description
array	Name of an array.
start_pos	Starting index position of range to set.
end_pos	Ending index position of range to set. If this value is greater than array length, ColdFusion adds elements to array.
value	Value to which to set each element in the range.

Example

```
<h3>ArraySet Example</h3>

<!--- Create an array. --->
<cfset MyNewArray = ArrayNew(1)>
<!--- ArrayToList does not function properly if the Array has not been initialized
    with ArraySet. --->
<cfset temp = ArraySet(MyNewArray, 1,6, "Initial Value")>

<!--- Set some elements. --->
<cfset MyNewArray[1] = "Sample Value">
<cfset MyNewArray[3] = "43">
<cfset MyNewArray[6] = "Another Value">
...
```

ArraySort

Description
Sorts array elements numerically or alphanumerically.

Returns
True, if sort is successful; False, otherwise.

Category
Array functions, List functions

Function syntax
`ArraySort(array, sort_type [, sort_order ])`

History
ColdFusion MX:

- Changed thrown exceptions: This function can throw the ArraySortSimpleValueException error and ValueNotNumeric error.
- Changed the order in which sorted elements are returned: In a `textnocase`, descending sort, this function might return elements in a different sort order than in earlier releases. If `sort_type = "textnocase"` and `sort_order = "desc"`, ColdFusion processes elements that *differ only in case* differently from earlier releases, as follows:
  - ColdFusion reverses the elements' original order.
  - Releases earlier than ColdFusion MX do not change the elements' original order.

  For example, in a `textnocase`, `desc` sort of `d,a,a,b,A`, the following occurs:
  - ColdFusion MX and later returns `d,b,A,a,a`
  - Releases earlier than ColdFusion MX return `d,b,a,a,A`

Parameters

Parameter	Description
`array`	Name of an array

Parameter	Description
sort_type	• numeric: sorts numbers • text: sorts text alphabetically, taking case into account (also known as case sensitive). All letters of one case precede the first letter of the other case: - aabzABZ, if sort_order = "asc" (ascending sort) - ZBAzbaa, if sort_order = "desc" (descending sort) • textnocase: sorts text alphabetically, without regard to case (also known as case-insensitive). A letter in varying cases precedes the next letter: - aAaBbBzzZ, in an ascending sort; preserves original intra-letter order - ZzzBbBaAa, in a descending sort; reverses original intra-letter order
sort_order	• asc - ascending sort order. Default. - aabzABZ or aAaBbBzzZ, depending on value of sort_type, for letters - from smaller to larger, for numbers • desc - descending sort order. - ZBAzbaa or ZzzBbBaAa, depending on value of sort_type, for letters - from larger to smaller, for numbers

Throws

If an array element is something other than a simple element, this function throws an ArraySortSimpleValueException error. If sort_type is numeric and an array element is not numeric, this function throws a ValueNotNumeric error.

Example

```
<!--- This example shows ArraySort. --->
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
    SELECT FirstName, LastName FROM Employees
</cfquery>
<!--- Create an array. --->
<cfset myArray = ArrayNew(1)>
<!--- Loop through the query and append these names successively to the last element. --->
<cfloop query = "GetEmployeeNames">
    <cfset temp = ArrayAppend(myArray, "#FirstName# #LastName#")>
</cfloop>
<!--- Show the resulting array as a list. --->
<cfset myList = ArrayToList(myArray, ",")>
<!--- Sort that array in descending order alphabetically. --->
<cfset isSuccessful = ArraySort(myArray, "textnocase", "desc")>
...
```

ArraySum

Description

Array sum function.

Returns

The sum of values in an array. If the `array` parameter value is an empty array, returns zero.

Category

Array functions, Mathematical functions

Function syntax

`ArraySum(array)`

Parameters

Parameter	Description
`array`	Name of an array

Example

```
<h3>ArraySum Example</h3>
<p>This example uses ArraySum to add two numbers.<br> </p>
<!--- After checking whether the form has been submitted, the code creates
    an array and assigns the form fields to the first two elements in the array. --->
<cfif IsDefined("FORM.submit")>
    <cfset myNumberArray = ArrayNew(1)>
    <cfset myNumberArray[1] = number1>
    <cfset myNumberArray[2] = number2>

    <cfif Form.Submit is "Add">
        <!--- Use ArraySum to add the number in the array. --->
        <p>The sum of the numbers is
        <cfoutput>#ArraySum(myNumberArray)#.</cfoutput>
    </cfif>
</cfif>
<!--- This form provides two numeric fields that are added when the form is submitted. --->
<form action = "arraysum.cfm" method="post">
    <input type = "hidden" name = "number1_Float">
    <input type = "hidden" name = "number2_Float">
    <input type = "text" name = "number1">
    <br>
    <input type = "text" name = "number2">
    <br>
    <input type = "submit" name = "submit" value = "Add">
</form>
```

ArraySwap

Description

Swaps array values of an array at specified positions. This function is more efficient than multiple `cfset` tags.

Returns

True, on successful completion.

Category

Array functions

Function syntax

`ArraySwap(array, position1, position2)`

See also

"Functions for XML object management" on page 879 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`array`	Name of an array
`position1`	Position of first element to swap
`position2`	Position of second element to swap

Example

```
<h3>ArraySwap Example</h3>

<cfset month = ArrayNew(1)>
<cfset month[1] = "February">
<cfset month[2] = "January">
<cfset temp = ArraySwap(month, 1, 2)>
<cfset temp = ArrayToList(month)>

<p>Show the results: <cfoutput>#temp#</cfoutput>
```

ArrayToList

Description

Converts a one-dimensional array to a list.

Returns

Delimited list, as a string.

Category

Array functions, Conversion functions, List functions

Function syntax

```
ArrayToList(array [, delimiter ])
```

Parameters

Parameter	Description
array	Name of array
delimiter	Character or multicharacter string to separate list elements. The default value is comma.

Example

```
<h3>ArrayToList Example</h3>

<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
    SELECT FirstName, LastName FROM Employees
</cfquery>

<!--- Create an array. --->
<cfset myArray = ArrayNew(1)>

<!--- Loop through the query, append names successively to the last element. --->
<cfloop query = "GetEmployeeNames">
    <cfset temp = ArrayAppend(myArray, "#FirstName# #LastName#")>
</cfloop>

<!--- Show the resulting array as a list. --->
<cfset myList = ArrayToList(myArray, ",")>

<!--- Sort that array in descending order alphabetically. --->
<cfset myAlphaArray = ArraySort(myArray, "textnocase", "desc")>

<!--- Show the resulting alphabetized array as a list. --->
<cfset myAlphaList = ArrayToList(myArray, ",")>

<!--- Output the array as a list. --->
<cfoutput>
    <p>The contents of the array are as follows:
    <p>#myList#
    <p>This array, alphabetized by first name (descending):
    <p>#myAlphaList#
    <p>This array has #ArrayLen(MyArray)# elements.
</cfoutput>
```

Asc

Description
Determines the value of a character.

Returns
The value of the first character of a string; if the string is empty, returns zero.

Category
String functions

Function syntax
`Asc(string)`

See also
`Chr`

History
ColdFusion MX: Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode characters, up to a value of 65536. (Earlier releases supported 1-255.)

Parameters

Parameter	Description
`string`	A string

Example
```
<h3>Asc Example</h3>
<!--- If the character string is not empty, output its ASCII value. --->
<cfif IsDefined("FORM.charVals")>

    <cfif FORM.charVals is not "">
        <cfoutput>#Left(FORM.charVals,1)# =
 #Asc(FORM.charVals)#</cfoutput>
    <cfelse>
<!--- If it is empty, output an error message. --->
        <h4>Enter a character</h4>
    </cfif>
</cfif>

<form action = "asc.cfm" method=post>
    <p>Enter a character to see its ASCII value
    <br><input type = "Text" name = "CharVals" size = "1" maxlength = "1"></p>
    <p><input type = "Submit" name = ""> <input type = "RESET"></p>
</form>
```

ASin

Description

Determines the arcsine of a number. The arcsine is the angle whose sine is *number*.

Returns

The arcsine, in radians, of a number.

Category

Mathematical functions

Function syntax

`ASin(number)`

See also

`Sin`, `Cos`, `ACos`, `Tan`, `Atn`, `Pi`

Parameters

Parameter	Description
`number`	Sine of an angle. The value must be between -1 and 1, inclusive.

Usage

The range of the result is -π/2 to π/2 radians. To convert degrees to radians, multiply degrees by π/180. To convert radians to degrees, multiply radians by 180/π.

Example

```
<h3>ASin Example</h3>
<!--- Output its arcsine value. --->
<cfif IsDefined("FORM.SinNum")>
    <cfif IsNumeric(FORM.SinNum)>
        <cfif FORM.SinNum LESS THAN OR EQUAL TO 1>
            <cfif FORM.SinNum GREATER THAN OR EQUAL TO -1>
                ASin(<cfoutput>#FORM.SinNum#</cfoutput>) =
                 <cfoutput>#ASin(FORM.sinNum)# Radians</cfoutput>
                 <br> or <br>ASin(<cfoutput>#FORM.SinNum#</cfoutput>) =
                <cfoutput>
                    #ASin(FORM.sinNum) * 180/Pi()# Degrees
                </cfoutput>
            <cfelse>
<!--- If it is less than negative one, output an error message. --->
                <h4>Enter the sine of the angle to calculate, in degrees and radians.
                The value must be between 1 and -1, inclusive.</h4>
            </cfif>
        <cfelse>
<!--- If it is greater than one, output an error message. --->
        <h4>Enter the sine of the angle to calculate, in degrees and radians. The
value must be between 1 and -1, inclusive.</h4>
        </cfif>
    <cfelse>
<!--- If it is empty, output an error message. --->
        <h4>Enter the sine of the angle to calculate, in degrees and radians. The
value must be between 1 and -1,inclusive.</h4>
    </cfif>
</cfif>
<form action = "./evaltest.cfm" method="post">
```

```
<p>Enter a number to get its arcsine in Radians and Degrees.
<br><input type = "Text" name = "SinNum" size = "25">
<p><input type = "Submit" name = ""> <input type = "RESET">
</form>
```

Atn

Description

Arctangent function. The arctangent is the angle whose tangent is *number*.

Returns

The arctangent, in radians, of a number.

Category

Mathematical functions

Function syntax

`Atn(number)`

See also

`Atn`, `Sin`, `ASin`, `Cos`, `ACos`, `Pi`

Parameters

Parameter	Description
`number`	Tangent of an angle

Usage

The range of the result is -π/2 to π/2 radians. To convert degrees to radians, multiply degrees by π/180. To convert radians to degrees, multiply radians by 180/π.

Example

```
<h3>Atn Example</h3>
<!--- Output its Atn value. --->
<cfif IsDefined("FORM.AtnNum")>
    <cfif IsNumeric(FORM.AtnNum)>
        Atn(<cfoutput>#FORM.AtnNum#</cfoutput>) is
 <cfoutput>#Atn(FORM.AtnNum)# radians is #Atn(FORM.AtnNum * 180/PI())# Degrees</cfoutput>
    <cfelse>
<!--- If it is empty, output an error message. --->
        <h4>Enter a number</h4>
    </cfif>
</cfif>
<form action = "evaltest.cfm" method="post">
    <p>Enter a number to get its arctangent in Radians and Degrees
    <br><input type = "Text" name = "atnNum" size = "25"></p>
    <p><input type = "Submit" name = ""> <input type = "RESET"></p>
</form>
```

AuthenticatedContext

Description

This function is obsolete. Use the newer security tools; see "Conversion functions" on page 641 and "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*.

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later ColdFusion releases.

AuthenticatedUser

Description

This function is obsolete. Use the newer security tools; see "Conversion functions" on page 641 and "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*.

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later ColdFusion releases.

BinaryDecode

Description

Converts a string to a binary object. Used to convert binary data that has been encoded into string format back into binary data.

Returns

A binary object.

Category

Conversion functions, String functions

Function syntax

```
BinaryDecode(string, binaryencoding)
```

See also

BinaryEncode, CharsetEncode, CharsetDecode

History

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
string	A string containing encoded binary data.
binaryencoding	A string that specifies the algorithm used to encode the original binary data into a string; must be one of the following: • Hex: the characters 0-9 and A-F represent the hexadecimal value of each byte; for example, 3A. • UU: data is encoded using the UNIX UUencode algorithm. • Base64: data is encoded using the Base64 algorithm, as specified by IETF RFC 2045, at www.ietf.org/rfc/rfc2045.txt.

Usage

Use this function to convert a binary-encoded string representation of binary data back to a binary object for use in your application. Binary data is often encoded as a string for transmission over many Internet protocols, such as HTTP and SMTP, or for storage in a database.

Adobe recommends that you use the BinaryDecode function, not the ToBinary(*base64data*) function, to convert Base64-encoded data to binary data in all new applications.

See the following pages for additional information on handling binary data:

- cffile for loading and reading binary data in files
- cfwddx for serializing and deserializing binary data
- IsBinary for checking variables for binary format
- Len for determining the length of a binary object

Example

The following example reads a GIF file as binary data, converts it to a binary-encoded string, converts the binary-encoded data back to binary data and writes the result to a file. It displays the encoded string and the image in the output file.

```
<h3>Binary Encoding Conversion Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.binEncoding")>

    <!--- Read in a binary data file. --->
    <cffile action="readbinary"
file="C:\CFusionMX7\wwwroot\CFIDE\administrator\images\help.gif" variable="binimage">

    <!--- Convert the read data to binary encoding and back to binary data. --->
    <cfscript>
        binencode=BinaryEncode(binimage, Form.binEncoding);
        bindecode=BinaryDecode(binencode, Form.binEncoding);
    </cfscript>

    <!---Write the converted results to a file. --->
    <cffile action="write" file="C:\temp\help.gif" output="#bindecode#" addnewline="No" >

    <!--- Display the results. --->
    <cfoutput>
        <p><b>The binary encoding:</b> #Form.binEncoding#</p>

        <p><b>The image converted into a binary-encoded string by BinaryEncode
            </b><br>
            #binencode#</p>
        <p><b>The image as written back to a file after converting back to binary
            using BinaryDecode</b><br>
        <img src="C:\temp\help.gif"><br>
    </cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select binary encoding</b><br>
    <select size="1" name="binEncoding" >
        <option selected>UU</option>
        <option>Base64</option>
        <option>Hex</option>
    </select><br>
    <br>
    <input type = "Submit" value = "convert my data">
</form>
```

BinaryEncode

Description
Converts binary data to a string.

Returns
An encoded string representing the binary data.

Category
Conversion functions, String functions

Function syntax
`BinaryEncode(binarydata, encoding)`

See also
`BinaryDecode`, `CharsetEncode`, `CharsetDecode`

History
ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`binarydata`	A variable containing the binary data to encode.
`encoding`	A string that specifies the encoding method to use to represent the data; one of the following: • Hex: use the characters 0-9 and A-F to represent the hexadecimal value of each byte; for example, 3A. • UU: use the UNIX UUencode algorithm to convert the data. • Base64: use the Base64 algorithm to convert the data, as specified by IETF RFC 2045, at www.ietf.org/rfc/rfc2045.txt.

Usage
Binary objects and, in some cases, 8-bit characters, cannot be transported over many Internet protocols, such as HTTP and SMTP, and might not be supported by some database systems. By Binary encoding the data, you convert the data into a format that you can transfer over any Internet protocol or store in a database as character data. To convert the data back to a binary format, use the `BinaryDecode` function.

Adobe recommends that you use the `BinaryEncode` function, and not the `ToBase64(binarydata)` function, to convert binary data to Base64 data in all new applications.

This function provides a superset of the functionality of the `ToBase64(binarydata)` function.

See the following pages for additional information on handling binary data:

- `cffile` for loading and reading binary data
- `cfwddx` for serializing and deserializing binary data
- `IsBinary` for checking variables for binary format
- `Len` for determining the length of a binary object

Example

The following example reads a GIF file as binary data, converts it to a binary-encoded string, converts the binary-encoded data back to binary data, and writes the result to a file. It displays the encoded string and the image in the output file.

```
<h3>Binary Encoding Conversion Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.binEncoding")>

    <!--- Read in a binary data file. --->
    <cffile action="readbinary"
        file="C:\CFusionMX7\wwwroot\CFIDE\administrator\images\help.gif"
        variable="binimage">

    <!--- Convert the read data to binary encoding and back to binary data. --->
    <cfscript>
        binencode=BinaryEncode(binimage, Form.binEncoding);
        bindecode=BinaryDecode(binencode, Form.binEncoding);
    </cfscript>

    <!---Write the converted results to a file. --->
    <cffile action="write" file="C:\temp\help.gif" output="#bindecode#" addnewline="No" >

    <!--- Display the results. --->
    <cfoutput>
        <p><b>The binary encoding:</b> #Form.binEncoding#</p>

        <p><b>The image converted into a binary-encoded string by BinaryEncode
            </b><br>
            #binencode#</p>
        <p><b>The image as written back to a file after converting back to binary
            using BinaryDecode</b><br>
        <img src="C:\temp\help.gif"><br></p>
    </cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select binary encoding</b><br>
    <select size="1" name="binEncoding" >
        <option selected>UU</option>
        <option>Base64</option>
        <option>Hex</option>
    </select><br>
    <br>
    <input type = "Submit" value = "convert my data">
</form>
```

BitAnd

Description
Performs a bitwise logical AND operation.

Returns
The bitwise AND of two long integers.

Category
Mathematical functions

Function syntax
`BitAnd(number1, number2)`

See also
`BitNot`, `BitOr`, `BitXor`

Parameters

Parameter	Description
`number1`	32-bit signed integer
`number2`	32-bit signed integer

Usage
Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example
```
<h3>BitAnd Example</h3>

<p>Returns the bitwise AND of two long integers.</p>
<p>BitAnd(5,255): <cfoutput>#BitAnd(5,255)#</cfoutput></p>
<p>BitAnd(5,0): <cfoutput>#BitAnd(5,0)#</cfoutput></p>
<p>BitAnd(128,128): <cfoutput>#BitAnd(128,128)#</cfoutput></p>
```

BitMaskClear

Description
Performs a bitwise mask clear operation.

Returns
A number, bitwise cleared, with *length* bits beginning at *start*.

Category
Mathematical functions

Function syntax
```
BitMaskClear(number, start, length)
```

See also
BitMaskRead, BitMaskSet

Parameters

Parameter	Description
number	32-bit signed integer
start	Integer, in the range 0-31, inclusive; start bit for mask
length	Integer, in the range 0-31, inclusive; length of mask

Usage
Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example
```
<h3>BitMaskClear Example</h3>

<p>Returns number bitwise cleared with length bits beginning from start.</p>

<p>BitMaskClear(255, 4, 4): <cfoutput>#BitMaskClear(255, 4, 4)#</cfoutput></p>
<p>BitMaskClear(255, 0, 4): <cfoutput>#BitMaskClear(255, 0, 4)#</cfoutput></p>
<p>BitMaskClear(128, 0, 7): <cfoutput>#BitMaskClear(128, 0, 7)#</cfoutput></p>
```

BitMaskRead

Description
Performs a bitwise mask read operation.

Returns
An integer, created from *length* bits of *number*, beginning at *start*.

Category
Mathematical functions

Function syntax
`BitMaskRead(number, start, length)`

See also
`BitMaskClear`, `BitMaskSet`

Parameters

Parameter	Description
`number`	32-bit signed integer to mask
`start`	Integer, in the range 0-31, inclusive; start bit for read
`length`	Integer, in the range 0-31, inclusive; length of mask

Usage
Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example
```
<h3>BitMaskRead Example</h3>
<p>Returns integer created from <em>length</em> bits of <em>number</em>, beginning
    with <em>start</em>.</p>

<p>BitMaskRead(255, 4, 4): <cfoutput>#BitMaskRead(255, 4, 4)#</cfoutput></p>
<p>BitMaskRead(255, 0, 4): <cfoutput>#BitMaskRead(255, 0, 4)#</cfoutput></p>
<p>BitMaskRead(128, 0, 7): <cfoutput>#BitMaskRead(128, 0, 7)#</cfoutput></p>
```

BitMaskSet

Description
Performs a bitwise mask set operation.

Returns
A number, bitwise masked with *length* bits of *mask* beginning at *start*.

Category
Mathematical functions

Function syntax
`BitMaskSet(number, mask, start, length)`

See also
`BitMaskClear`, `BitMaskRead`

Parameters

Parameter	Description
`number`	32-bit signed integer
`mask`	32-bit signed integer; mask
`start`	Integer, in the range 0-31, inclusive; start bit for mask
`length`	Integer, in the range 0-31, inclusive; length of mask

Usage
Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example
```
<h3>BitMaskSet Example</h3>
<p>Returns number bitwise masked with length bits of mask beginning at start.</p>

<p>BitMaskSet(255, 255, 4, 4): <cfoutput>#BitMaskSet(255, 255, 4, 4)#</cfoutput></p>
<p>BitMaskSet(255, 0, 4, 4): <cfoutput>#BitMaskSet(255, 0, 4, 4)#</cfoutput></p>
<p>BitMaskSet(0, 15, 4, 4): <cfoutput>#BitMaskSet(0, 15, 4, 4)#</cfoutput></p>
```

BitNot

Description
Performs a bitwise logical NOT operation.

Returns
A number; the bitwise NOT of a long integer.

Category
Mathematical functions

Function syntax
`BitNot(number)`

See also
`BitAnd`, `BitOr`, `BitXor`

Parameters

Parameter	Description
`number`	32-bit signed integer

Usage
Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example
```
<h3>BitNot Example</h3>
<p>Returns the bitwise NOT of a long integer.</p>

<p>BitNot(0): <cfoutput>#BitNot(0)#</cfoutput></p>
<p>BitNot(255): <cfoutput>#BitNot(255)#</cfoutput></p>
```

BitOr

Description
Performs a bitwise logical OR operation.

Returns
A number; the bitwise OR of two long integers.

Category
Mathematical functions

Function syntax
`BitOr(number1, number2)`

See also
`BitAnd`, `BitNot`, `BitXor`

Parameters

Parameter	Description
`number1`	32-bit signed integer
`number2`	32-bit signed integer

Usage
Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example
```
<h3>BitOr Example</h3>
<p>Returns the bitwise OR of two long integers.</p>

<p>BitOr(5,255): <cfoutput>#BitOr(5,255)#</cfoutput></p>
<p>BitOr(5,0): <cfoutput>#BitOr(5,0)#</cfoutput></p>
<p>BitOr(7,8): <cfoutput>#BitOr(7,8)#</cfoutput></p>
```

BitSHLN

Description

Performs a bitwise shift-left, no-rotation operation.

Returns

A number, bitwise shifted without rotation to the left by *count* bits.

Category

Mathematical functions

Function syntax

`BitSHLN(number, count)`

See also

`BitSHRN`

Parameters

Parameter	Description
`number`	32-bit signed integer
`count`	Integer, in the range 0-31, inclusive; number of bits to shift the number

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example

```
<h3>BitSHLN Example</h3>
<p>Returns the number, bitwise shifted, without rotation, to the left by count bits.</p>

<p>BitSHLN(1,1): <cfoutput>#BitSHLN(1,1)#</cfoutput></p>
<p>BitSHLN(1,30): <cfoutput>#BitSHLN(1,30)#</cfoutput></p>
<p>BitSHLN(1,31): <cfoutput>#BitSHLN(1,31)#</cfoutput></p>
<p>BitSHLN(2,31): <cfoutput>#BitSHLN(2,31)#</cfoutput></p>
```

BitSHRN

Description

Performs a bitwise shift-right, no-rotation operation.

Returns

A number, bitwise shifted, without rotation, to the right by *count* bits.

Category

Mathematical functions

Function syntax

```
BitSHRN(number, count)
```

See also

BitSHLN

Parameters

Parameter	Description
number	32-bit signed integer
count	Integer, in the range 0-31, inclusive. Number of bits to shift the number

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example

```
<h3>BitSHRN Example</h3>
<p>Returns a number, bitwise shifted, without rotation, to the right, by count bits.</p>

<p>BitSHRN(1,1): <cfoutput>#BitSHRN(1,1)#</cfoutput></p>
<p>BitSHRN(255,7): <cfoutput>#BitSHRN(255,7)#</cfoutput></p>
<p>BitSHRN(-2147483548,1): <cfoutput>#BitSHRN(-2147483548,1)#</cfoutput></p>
```

BitXor

Description

Performs a bitwise logical XOR operation.

Returns

Bitwise XOR of two long integers.

Category

Mathematical functions

Function syntax

`BitXor(number1, number2)`

See also

`BitAnd`, `BitNot`, `BitOr`

Parameters

Parameter	Description
`number1`	32-bit signed integer
`number2`	32-bit signed integer

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 – 2147483647.

Example

```
<h3>BitXOr Example</h3>
<p>Returns the bitwise XOR of two long integers.</p>

<p>BitXOr(5,255): <cfoutput>#BitXOr(5,255)#</cfoutput></p>
<p>BitXOr(5,0): <cfoutput>#BitXOr(5,0)#</cfoutput></p>
<p>BitXOr(128,128): <cfoutput>#BitXOr(128,128)#</cfoutput></p>
```

Ceiling

Description

Determines the closest integer that is greater than a specified number.

Returns

The closest integer that is greater than a given number.

Category

Mathematical functions

Function syntax

`Ceiling(number)`

See also

`Int`, `Fix`, `Round`

Parameters

Parameter	Description
`number`	A real number

Example

```
<h3>Ceiling Example</h3>

<cfoutput>
    <p>The ceiling of 3.4 is #ceiling(3.4)#</p>
    <p>The ceiling of 3 is #ceiling(3)#</p>
    <p>The ceiling of 3.8 is #ceiling(3.8)#</p>
    <p>The ceiling of -4.2 is #ceiling(-4.2)#</p>
</cfoutput>
```

CharsetDecode

Description
Converts a string to a binary representation.

Returns
A binary object that represents the string.

Category
Conversion functions, String functions

Function syntax
`CharsetDecode(string, encoding)`

See also
`BinaryDecode`, `BinaryEncode`, `CharsetEncode`; "Determining the page encoding of server output" on page 344 in the *ColdFusion Developer's Guide*

History
ColdFusion MX 7: Added this function.

Parametersc

Parameter	Description
`string`	A string containing data to encode in binary format.
`encoding`	A string that specifies encoding of the input data. Must be a character encoding name recognized by the Java runtime. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For a complete list of character encoding names supported by Sun Java runtimes, see http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html and http://java.sun.com/j2se/1.4/docs/guide/intl/encoding.doc.html.

Usage
This function converts a string directly to a binary object. In releases of ColdFusion through ColdFusion MX 6.1, you had to use the `ToBase64` function to convert the string to Base64 and then use the `ToBinary` function to convert strings to binary data.

Example

The following example uses the `CharsetDecode` function to convert a string from a form to a binary object, and uses the `CharsetEncode` function to convert it back to the original value. You can change the character encoding that ColdFusion uses for the conversion. If you select the Asian language encodings, characters that are not in the specified character set are successfully converted.

```
<h3>Character Encoding Conversion Example</h3>
<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>

    <!--- Do the conversions. --->
    <cfscript>
        chardecode=CharsetDecode(Form.myString, Form.charEncoding);
        charencode=CharsetEncode(chardecode, Form.charEncoding);
    </cfscript>

    <!--- Display the input values and results. --->
    <cfoutput>
        <h3>Parameter Settings</h3>
        <p><b>The string:</b><br>
             #Form.myString#</p>
        <p><b>The character encoding:</b> #Form.charEncoding#</p>

        <h3>Results of the operations:</h3>
        <p><b>Dump of the string converted to a binary object by CharsetDecode:
            </b><br>
            <cfdump var="#chardecode#"></p>
        <p><b>The binary object converted back to a string by CharsetEncode:
            </b><br>
        #charencode#</p>
    </cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select the character encoding</b><br>
    <!--- This is a subset, additional encodings are available. --->
    <select size="1" name="charEncoding" >
        <option selected>UTF-8</option>
        <option>ASCII</option>
        <option>ISO8859_1</option>
        <option>CP1252</option>
        <option>SJIS</option>
        <option>MS932</option>
        <option>EUC_CN</option>
        <option>Big5</option>
    </select><br>
    <br>
    <b>Enter a string</b><br>
    <textarea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">
    The following four characters are not in all character encodings: ½àç÷
    </textarea><br>
    <br>
    <input type = "Submit" value = "convert my data">
</form>
```

CharsetEncode

Description
Uses the specified encoding to convert binary data to a string.

Returns
A string representation of the binary object.

Category
Conversion functions, String functions

Function syntax
`CharsetEncode(binaryobject, encoding)`

See also
`BinaryDecode`, `BinaryEncode`, `CharsetDecode`; "Determining the page encoding of server output" on page 344 in the *ColdFusion Developer's Guide*

History
ColdFusion MX 7: Added this function.

Parametersc

Parameter	Description
`binaryobject`	A variable containing binary data to decode into text.
`encoding`	The character encoding that was used to encode the string into binary format. It must be a character encoding name recognized by the Java runtime. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For a complete list of character encoding names supported by Sun Java runtimes, see http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html and http://java.sun.com/j2se/1.4/docs/guide/intl/encoding.doc.html.

Usage
Adobe recommends that you use this function, and not the `ToString` function, to convert binary data to strings in all new applications.

Example

The following example uses the `CharsetDecode` function to convert a string from a form to a binary object, and uses the `CharsetEncode` function to convert it back to the original value. You can change the character encoding that ColdFusion uses for the conversion. If you select the Asian language encodings, characters that are not in the specified character set are successfully converted.

```
<h3>Character Encoding Conversion Example</h3>
<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>

    <!--- Do the conversions. --->
    <cfscript>
        chardecode=CharsetDecode(Form.myString, Form.charEncoding);
        charencode=CharsetEncode(chardecode, Form.charEncoding);
    </cfscript>

    <!--- Display the input values and results. --->
    <cfoutput>
        <h3>Parameter Settings</h3>
        <p><b>The string:</b><br>
             #Form.myString#</p>
        <p><b>The character encoding:</b> #Form.charEncoding#</p>

        <h3>Results of the operations:</h3>
        <p><b>Dump of the string converted to a binary object by CharsetDecode:
            </b><br>
            <cfdump var="#chardecode#"></p>
        <p><b>The binary object converted back to a string by CharsetEncode:
            </b><br>
        #charencode#</p>
    </cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select the character encoding</b><br>
    <!--- This is a subset, additional encodings are available. --->
    <select size="1" name="charEncoding" >
        <option selected>UTF-8</option>
        <option>ASCII</option>
        <option>ISO8859_1</option>
        <option>CP1252</option>
        <option>SJIS</option>
        <option>MS932</option>
        <option>EUC_CN</option>
        <option>Big5</option>
    </select><br>
    <br>
    <b>Enter a string</b><br>
    <textArea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">
    The following four characters are not in all character encodings: ½âç÷
    </textArea><br>
    <br>
    <input type = "Submit" value = "convert my data">
</form>
```

Chr

Converts a numeric value to a UCS-2 character.

Returns
A character with the specified UCS-2 character code.

Category
String functions

Function syntax
`Chr(number)`

See also
`Asc`

History
ColdFusion MX: Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode characters, up to a value of 65535. (Earlier releases supported 1-255.)

Parameters

Parameter	Description
`number`	A value (a number in the range 0 to 65535, inclusive)

Usage
The values 0 – 31 are standard, nonprintable codes. For example:

- `Chr(10)` returns a linefeed character
- `Chr(13)` returns a carriage return character
- The two-character string `Chr(13) & Chr(10)` returns a Windows newline

Note: *For a complete list of the Unicode characters and their codes, see www.unicode.org/charts/*

Example
```
<!--- If the character string is not empty, output its Chr value. --->
<cfif IsDefined("form.charVals")>
        <cfoutput>#form.charVals# = #Chr(form.charVals)#</cfoutput>
</cfif>

<cfform action="#CGI.script_name#" method="POST">
    <p>Type an integer character code from 1 to 65535<br>
        to see its corresponding character.<br>
    <cfinput type="Text"
        name="CharVals"
        range="1,65535"
        message="Enter an integer from 1 to 65535"
        validate="integer"
        required="Yes"
        size="5"
        maxlength="5"
        >
<p><input type="Submit" name=""> <input type="RESET">
</cfform>
```

CJustify

Description

Centers a string in a field length.

Returns

String, center-justified by adding spaces before or after the input parameter. If *length* is less than the length of the input parameter string, the string is returned unchanged.

Category

Display and formatting functions, String functions

Function syntax

`Cjustify(string, length)`

See also

`LJustify`, `RJustify`

Parameters

Parameter	Description
`string`	A string or a variable that contains one. May be empty. If it is a variable that is defined as a number, the function processes it as a string.
`length`	A positive integer or a variable that contains one. Length of field. Can be coded as: • A number; for example, `6` • A string representation of a number; for example, `"6"` Any other value causes ColdFusion to throw an error.

Example

```
<!--- This example shows how to use CJustify. --->
<CFPARAM name = "jstring" DEFAULT = "">

<cfif IsDefined("FORM.submit")>
<cfdump var="#Form#">
    <cfset jstring = Cjustify("#FORM.justifyString#", 35)>
</cfif>
<html>
<head>
<title>CJustify Example</title>
</head>
<body>
<h3>CJustify</h3>
<p>Enter a string; it will be center-justified within the sample field.
<form action = "cjustify.cfm" method="post">
<p><input type = "Text" value = "<cfoutput>#jString#</cfoutput>"
        size = 35 name = "justifyString">
<p><input type = "Submit" name = "submit">
<input type = "RESET">
</form>
</body>
</html>
```

Compare

Description

Performs a case-sensitive comparison of two strings.

Returns

- -1, if *string1* is less than *string2*
- 0, if *string1* is equal to *string2*
- 1, if *string1* is greater than *string2*

Category

String functions

Function syntax

```
Compare(string1, string2)
```

See also

CompareNoCase, Find

Parameters

Parameter	Description
string1	A string or a variable that contains one
string2	A string or a variable that contains one

Usage

Compares the values of corresponding characters in *string1* and *string2*.

Example

```
<h3>Compare Example</h3>
<p>The compare function performs a <I>case-sensitive</I> comparison of two strings.</p>

<cfif IsDefined("FORM.string1")>
 <cfset comparison = Compare(FORM.string1, FORM.string2)>
 <!--- Switch on the variable to give various responses. --->
 <cfswitch expression = #comparison#>
    <cfcase value = "-1">
        <h3>String 1 is less than String 2</h3>
        <I>The strings are not equal</I>
    </cfcase>
    <cfcase value = "0">
        <h3>String 1 is equal to String 2</h3>
        <I>The strings are equal!</I>
    </cfcase>
    <cfcase value = "1">
        <h3>String 1 is greater than String 2</h3>
        <I>The strings are not equal</I>
    </cfcase>
    <cfdefaultcase>
        <h3>This is the default case</h3>
    </cfdefaultcase>
    </cfswitch>
</cfif>
<form action = "compare.cfm" method="post">
```

```
<p>String 1
<br><input type = "Text" name = "string1">
<p>String 2
<br><input type = "Text" name = "string2">
<p><input type = "Submit" value = "Compare these Strings" name = "">
    <input type = "RESET">
</form>
```

CompareNoCase

Description

Performs a case-insensitive comparison of two strings.

Returns

An indicator of the difference:

- A negative number, if *string1* is less than *string2*
- 0, if *string1* is equal to *string2*
- A positive number, if *string1* is greater than *string2*

Category

String functions

Function syntax

`CompareNoCase(string1, string2)`

See also

`Compare`, `FindNoCase`; "Ambiguous type expressions and strings" on page 40 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`string1`	A string or a variable that contains one
`string2`	A string or a variable that contains one

Example

```
<H3>CompareNoCase Example</H3>
<P>This function performs a <I>case-insensitive</I> comparison of two strings.
<cfif IsDefined("form.string1")>
<cfset comparison = Comparenocase(form.string1, form.string2)>
<!--- switch on the variable to give various responses --->
<cfswitch expression=#comparison#>
    <cfcase value="-1">
        <H3>String 1 is less than String 2</H3>
        <I>The strings are not equal</I>
    </cfcase>
    <cfcase value="0">
        <H3>String 1 is equal to String 2</H3>
        <I>The strings are equal!</I>
    </cfcase>
    <cfcase value="1">
        <H3>String 1 is greater than String 2</H3>
        <I>The strings are not equal</I>
    </cfcase>
    <cfdefaultcase>
        <H3>This is the default case</H3>
    </cfdefaultcase>
</cfswitch>
</cfif>
<form action="comparenocase.cfm" method="POST">
<P>String 1
<BR><input type="Text" name="string1">
```

```
<P>String 2
<BR><input type="Text" name="string2">
<P><input type="Submit" value="Compare these Strings" name="">
    <input type="RESET">
</form>
```

Cos

Description

Calculates the cosine of an angle that is entered in radians.

Returns

A number; the cosine of the angle.

Category

Mathematical functions

Function syntax

`Cos(number)`

See also

`ACos`, `Sin`, `ASin`, `Tan`, `Atn`, `Pi`

Parameters

Parameter	Description
`number`	Angle, in radians, for which to calculate the cosine

Usage

The range of the result is -1 to 1.

To convert degrees to radians, multiply degrees by π/180. To convert radians to degrees, multiply radians by 180/π.

Note: *Because the function uses floating point arithmetic, it returns a very small number (such as 6.12323399574E-017) for angles that should produce 0. To test for a 0 value, check whether the value is less than 0.0000000000001.*

Example

```
<h3>Cos Example</h3>
<!--- Calculate cosine if form has been submitted. --->
<cfif IsDefined("FORM.cosNum")>
<!--- Make sure input is a number. --->
    <cfif IsNumeric(#FORM.cosNum#)>
<!--- Convert degrees to radians, call the Cos function. --->
<cfset cosValue=#Cos((Form.cosNum * PI()) / 180)#>
<!--- 0.0000000000001 is the function's precision limit. If the absolute value
    of the returned cosine value is less, set result to 0 .--->
<cfif Abs(cosValue) LT 0.0000000000001>
 <cfset cosValue=0>
</cfif>
<cfoutput>Cos(#FORM.cosNum#) = #cosValue#<br><br></cfoutput>
    <cfelse>
<!--- If input is not a number, show an error message. --->
        <h4>You must enter a numeric angle in degrees.</h4>
    </cfif>
</cfif>
<form action = "#CGI.script_name#" method="post">
Enter an angle in degrees to get its cosine:
<br><input type = "Text" name = "cosNum" size = "15"><br><br>
<input type = "Submit" name = "">  
<input type = "RESET"
</form>
```

CreateDate

Description
Creates a date/time object.

Returns
A date/time value.

Category
Date and time functions

Function syntax
```
CreateDate(year, month, day)
```

See also
CreateDateTime, CreateODBCDate; "Date-time functions and queries when ODBC is not supported" on page 40 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
year	Integer in the range 0-9999. Integers in the range 0-29 are converted to 2000-2029. Integers in the range 30-99 are converted to 1930-1999. You cannot specify dates before AD 100.
month	Integer in the range 1 (January)-12 (December)
day	Integer in the range 1-31

Usage
CreateDate is a subset of CreateDateTime.

The time in the returned object is set to 00:00:00.

Example
```
<h3>CreateDate Example</h3>
<cfif IsDefined("form.year")>
<p>Your date value, generated with CreateDate:</p>
<cfset yourDate = CreateDate(form.year, form.month, form.day)>
<cfoutput>
<ul>
    <li>Formatted with CreateDate: #CreateDate(form.year, form.month, form.day)#</li>
    <li>Formatted with CreateDateTime: #CreateDateTime(form.year, form.month,
        form.day, 12,13,0)#</li>
    <li>Formatted with CreateODBCDate: #CreateODBCDate(yourDate)#</li>
    <li>Formatted with CreateODBCDateTime: #CreateODBCDateTime(yourDate)#</li>
</ul>

<p>The same value can be formatted with DateFormat:
<ul>
    <li>Formatted with CreateDate and DateFormat:
        #DateFormat(CreateDate(form.year, form.month, form.day), "mmm-dd-yyyy")#</li>
    <li>Formatted with CreateDateTime and DateFormat:
        #DateFormat(CreateDateTime(form.year, form.month, form.day, 12,13,0))#</li>
    <li>Formatted with CreateODBCDate and DateFormat:
        #DateFormat(CreateODBCDate(yourDate), "mmmm d, yyyy")#</li>
    <li>Formatted with CreateODBCDateTime and DateFormat:
```

```
        #DateFormat(CreateODBCDateTime(yourDate), "d/m/yy")#</li>
</ul>
</cfoutput>
</cfif>
<cfform action="createdate.cfm" METHOD="POST">
<p>Enter the year, month and day, as integers:
<pre>
Year<cfinput type="Text" name="year" value="1998" validate="integer" required="Yes">
Month<cfinput type="Text" name="month" value="6" validate="integer" required="Yes">
Day<cfinput type="Text" name="day" value="8" validate="integer" required="Yes">
</pre>
<p><input type="Submit" name=""> <input type="RESET">
</cfform>
```

CreateDateTime

Description

Creates a date-time object.

Returns

A date/time value.

Category

Date and time functions

Function syntax

```
CreateDateTime(year, month, day, hour, minute, second)
```

See also

CreateDate, CreateTime, CreateODBCDateTime, Now; "Date-time functions and queries when ODBC is not supported" on page 40 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
year	Integer in the range 0-9999. Integers in the range 0-29 are converted to 2000-2029. Integers in the range 30-99 are converted to 1930-1999. You cannot specify dates before AD 100.
month	Integer in the range 1 (January)–12 (December)
day	Integer in the range 1–31
hour	Integer in the range 0–23
minute	Integer in the range 0–59
second	Integer in the range 0–59

Example

```
<h3>CreateDateTime Example</h3>

<cfif IsDefined("form.year")>
Your date value, generated with CreateDateTime:
<cfset yourDate = CreateDateTime(form.year, form.month, form.day,
    form.hour, form.minute, form.second)>

<cfoutput>
<ul>
    <li>Formatted with CreateDate: #CreateDate(form.year, form.month, form.day)#</li>
    <li>Formatted with CreateDateTime: #CreateDateTime(form.year, form.month,
    form.day, form.hour, form.minute, form.second)#</li>
    <li>Formatted with CreateODBCDate: #CreateODBCDate(yourDate)#</li>
    <li>Formatted with CreateODBCDateTime: #CreateODBCDateTime(yourDate)#</li>
</ul>

<p>The same value can be formatted with DateFormat:
<ul>
    <li>Formatted with CreateDate and DateFormat:
        #DateFormat(CreateDate(form.year, form.month, form.day), "mmm-dd-yyyy")#</li>
    <li>Formatted with CreateDateTime and DateFormat:
        #DateFormat(CreateDateTime(form.year, form.month, form.day,
    form.hour, form.minute, form.second))#</li>
```

```
    <li>Formatted with CreateODBCDate and DateFormat:
        #DateFormat(CreateODBCDate(yourDate), "mmmm d, yyyy")#</li>
    <li>Formatted with CreateODBCDateTime and DateFormat:
        #DateFormat(CreateODBCDateTime(yourDate), "d/m/yy")#</li>
</ul>
</cfoutput>
</cfif>

<CFFORM ACTION="createdatetime.cfm" METHOD="POST">
<p>Please enter the year, month, and day, in integer format, for a date to view:
<PRE>
Year<CFINPUT TYPE="Text" NAME="year" VALUE="1998" VALIDATE="integer"
        REQUIRED="Yes">
Month<CFINPUT TYPE="Text" NAME="month" VALUE="6" RANGE="1,12"
        MESSAGE="Please enter a month (1-12)" VALIDATE="integer"
        REQUIRED="Yes">
Day <CFINPUT TYPE="Text" NAME="day" VALUE="8" RANGE="1,31"
        MESSAGE="Please enter a day of the month (1-31)" VALIDATE="integer"
        REQUIRED="Yes">
Hour<CFINPUT TYPE="Text" NAME="hour" VALUE="16" RANGE="0,23"
        MESSAGE="You must enter an hour (0-23)" VALIDATE="integer"
        REQUIRED="Yes">
Minute<CFINPUT TYPE="Text" NAME="minute" VALUE="12" RANGE="0,59"
        MESSAGE="You must enter a minute value (0-59)" VALIDATE="integer"
        REQUIRED="Yes">
Second<CFINPUT TYPE="Text" NAME="second" VALUE="0" RANGE="0,59"
        MESSAGE="You must enter a value for seconds (0-59)" VALIDATE="integer"
        REQUIRED="Yes">
</PRE>
<p><INPUT TYPE="Submit" NAME=""> <INPUT TYPE="RESET">
</cfform>
```

CreateObject

Description
Creates a ColdFusion object, of a specified type.

Returns
An object, of the specified type.

Category
Extensibility functions

History
ColdFusion 8:

- Added the .NET/dotnet type
- For web service object, added the `WSDL2Java` parameter

ColdFusion MX 7: For web service object: added the `portName` parameter, which specifies a port named in the `service` element of the WSDL.

ColdFusion MX:

1 Changed instantiation behavior: this function, and the `cfobject` tag, can instantiate ColdFusion components and web services. Executing operations on a CFC object executes CFML code that implements the CFC's method in the CFC file.

For more information, see the *ColdFusion Developer's Guide*.

2 For CORBA object: changed the Naming Service separator format for addresses from a dot to a forward slash. For example, if "`context=NameService`", for a class, use either of the following formats for the `class` parameter:

- "`/Eng/CF`"
- "`.current/Eng.current/CF`"

(In earlier releases, the format was "`.Eng.CF`".)

3 For CORBA object: changed the `locale` parameter; it specifies the Java config that contains the properties file.

CreateObject object types
For information about using this function, see these sections:

- "CreateObject: .NET object" on page 721
- "CreateObject: COM object" on page 723
- "CreateObject: component object" on page 725
- "CreateObject: CORBA object" on page 726
- "CreateObject: Java or EJB object" on page 728
- "CreateObject: web service object" on page 729

Note: *On UNIX, this function does not support COM objects.*

CreateObject: .NET object

Description

Creates a .NET object, that is, a ColdFusion proxy for accessing a class in a local or remote .NET assembly.

Returns

A .NET object, that is, a ColdFusion reference to a local or remote .NET assembly class.

Function syntax

`CreateObject(type, class, assembly[, server, port, protocol, secure])`

See also

`cfobject: .NET object`, `DotNetToCFType`, "Using Microsoft .NET Assemblies" on page 952 in the *ColdFusion Developer's Guide*

Parameters

Attribute	Default	Description
`type`		Object type. Must be `.NET` or `dotnet` for .NET objects.
`class`		Name of the .NET class to represent as an object.
`assembly`	`mscorlib.dll which contains the .NET core classes`	**For local .NET assemblies**, the absolute path or paths to the assembly or assemblies (.exe or .dll files) from which to access the .NET class and its supporting classes. If a class in an assembly requires supporting classes that are in other assemblies, you must also specify those assemblies. You can, however, omit the supporting assemblies for the following types of supporting classes: • .NET core classes (classes in mscorlib.dll) • classes in assemblies that are in the global assembly cache (GAC) To specify multiple assemblies, use a comma-delimited list. **For remote .NET assemblies**, you must specify the absolute path or paths of the local proxy JAR file or files that represent the assemblies. If you omit this parameter, and there is no local .NET installation, the function fails without generating an error. If you omit this parameter, there is a local .NET installation, and the specified class is not in the .NET core classes, ColdFusion generates an error.
`server`	`localhost`	Host name or IP address of the server where the .NET-side agent is running. Can be in any of these forms: • server name (for example, myserver) • IP address (for example, 127.0.0.1) You must specify this attribute to access .NET components on a remote server.
`port`	`6086`	Port number at which the .NET-side agent is listening.
`protocol`	`tcp`	Protocol to use to use for communication between ColdFusion and .NET. Must be one of the following values: • `http`: Use HTTP/SOAP communication protocol. This option is slower than tcp, but might be required for access through a firewall. • `tcp`: Use binary TCP/IP protocol. This method is more efficient than HTTP.
`secure`	`false`	Whether to secure communications with the .NET-side agent. If `true`, ColdFusion uses SSL to communicate with .NET.

Usage

The `CreateObject` function and `cfobject` tag differ only in syntax. For more information on creating ColdFusion .NET objects, see "cfobject: .NET object" on page 410. For detailed information on using the .NET assemblies in ColdFusion, see "Using Microsoft .NET Assemblies" on page 952 in the *ColdFusion Developer's Guide*.

CreateObject: COM object

Description

The `CreateObject` function can create a Component Object Model (COM) object.

To create a COM object, you must provide this information:

- The object's program ID or filename
- The methods and properties available to the object through the IDispatch interface
- The arguments and return types of the object's methods

For most objects, you can get this information from the OLEView utility.

Note: *On UNIX, this function does not support COM objects.*

Returns

A COM object.

Function syntax

`CreateObject(type, class, context, serverName)`

See also

`ReleaseComObject`, `cfobject`; "Integrating COM and CORBA Objects in CFML Applications" on page 974 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`type`	Type of object to create. • com • corba • java • component • webservice
`class`	Component ProgID for the object to invoke.
`context`	• `InProc` • `Local` • `Remote`
`serverName`	Server name, using UNC or DNS convention, in one of these forms: • \\lanserver • lanserver • http://www.servername.com • www.servername.com • 127.0.0.1 If `context = "remote"`, this parameter is required.

Usage

The following example creates the Windows Collaborative Data Objects (CDO) for NTS NewMail object to send mail. You use this code in a `cfscript` tag.

```
Mailer = CreateObject("COM", "CDONTS.NewMail");
```

CreateObject: component object

Description

The `CreateObject` function can create an instance of a ColdFusion component (CFC) object.

Returns

A component object.

Function syntax

`CreateObject(type, component-name)`

See also

"Building and Using ColdFusion Components" on page 158 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`type`	Type of object to create. • com • corba • java • component • webservice
`component-name`	The CFC name; corresponds to the name of the file that defines the component; for example, use engineComp to specify the component defined in the engineComp.cfc file

Usage

On UNIX systems, ColdFusion searches first for a file with a name that matches the specified component name, but is all lowercase. If it does not find the file, it looks for a filename that matches the component name exactly, with the identical character casing.

In the following example, the CFScript statements assign the `tellTimeCFC` variable to the `tellTime` component using the `CreateObject` function. The `CreateObject` function references the component in another directory. To invoke component methods, you use function syntax.

```
<b>Server's Local Time:</b>
<cfscript>
    tellTimeCFC=CreateObject("component","appResources.components.
        tellTime");
    tellTimeCFC.getLocalTime();
</cfscript>
<br>
<b>Calculated UTC Time:</b>
<cfscript>
    tellTimeCFC.getUTCTime();
</cfscript>
```

CreateObject: CORBA object

Description

The `CreateObject` function can call a method on a CORBA object. The object must be defined and registered for use.

Returns

A handle to a CORBA interface.

Function syntax

`CreateObject(type, context, class, locale)`

See also

"Integrating COM and CORBA Objects in CFML Applications" on page 974 in the *ColdFusion Developer's Guide*

History

See the History section of the main `CreateObject` function page.

Parameters

Parameter	Description
`type`	Type of object to create. • com • corba • java • component • webservice
`context`	• IOR: ColdFusion uses IOR to access CORBA server • NameService: ColdFusion uses naming service to access server. Valid only with the InitialContext of a VisiBroker ORB.
`class`	• If `context = "ior"`: absolute path of file that contains string version of the Interoperable Object Reference (IOR). ColdFusion must be able to read file; it should be local to ColdFusion server or accessible on network • If `context = "nameservice"`: forward slash-delimited naming context for naming service. For example: `Allaire//Doc/empobject`.
`locale`	The name of the Java config that holds the properties file. For more information, see *Configuring and Administering ColdFusion*.

Usage

In the *`class`* parameter, if "`context=NameService`", use a dot separator for the first part of the string. Use either of the following formats:

- "`/Eng/CF`"
- "`.current/Eng.current/CF`"

ColdFusion supports CORBA through the Dynamic Invocation Interface (DII). To use this function with CORBA objects, you must provide the name of the file that contains a string version of the IOR, or the object's naming context in the naming service. You must provide the object's attributes, method names and method signatures.

This function supports user-defined types (structures, arrays, and sequences).

Example

```
myobj = CreateObject("corba", "d:\temp\tester.ior", "ior",
    "visibroker") //    uses IOR

myobj = CreateObject("corba", "/Eng/CF", "nameservice",
    "visibroker")    // uses nameservice

myobj = CreateObject("corba", "d:\temp\tester.ior", "nameservice")
       // uses nameservice and default configuration
```

CreateObject: Java or EJB object

Description

The `CreateObject` function can create a Java object, and, by extension, an EJB object.

Returns

A Java object.

Function syntax

`CreateObject(type, class)`

Parameters

Parameter	Description
`type`	Type of object to create. • com • corba • java • component • webservice
`class`	A Java class name

Usage

Any Java class available in the class path that is specified in the ColdFusion Administrator can be loaded and used from ColdFusion with the `CreateObject` function.

To access Java methods and fields:

1. Call the `CreateObject` function or the `cfobject` tag to load the class.
2. Use the `init` method, with appropriate arguments, to call an instance of the class. For example:

```
<cfset ret = myObj.init(arg1, arg2)>
```

Calling a public method on the object without first calling the `"init"` method invokes a static method. Arguments and return values can be any Java type (simple, array, object). If strings are passed as arguments, ColdFusion does the conversions; if strings are received as return values, ColdFusion does no conversion.

Overloaded methods are supported if the number of arguments is different. Future enhancements will let you use cast functions that allow method signatures to be built more accurately.

CreateObject: web service object

Description

This function can create a web service object.

Returns

A web service object.

Function syntax

```
CreateObject(type, urltowsdl[, portname, wsdl2JavaArgs])

OR

CreateObject(type, urltowsdl, argStruct)
```

Parameters

Parameter	Description
type	Type of object to create. • com • corba • java • component • webservice
urltowsdl	Specifies the URL to web service WSDL file. One of the following: • The absolute URL of the web service • The Name (string) assigned in the ColdFusion Administrator to the web service
portname	The port name for the web service. This value is case-sensitive and corresponds to the port element's name attribute under the service element. Specify this parameter if the web service contains multiple ports. If no port name is specified, ColdFusion uses the first port found in the WSDL.
wsdl2JavaArgs	A string containing a space-delimited list of arguments to pass to the WSDL2Java tool that generates Java stubs for the web services. Useful arguments include the following: • -W or --noWrapped: Turns off the special treatment of wrapped document/literal style operations. • -a or --all: Generates code for all elements in the WSDL, even unreferenced ones. • -w or --wrapArrays: Prefers building beans to straight arrays for wrapped XML array types. This switch is not in included in the Axis documentation. For detailed information on valid arguments, see the Apache Axis WSDL2Java Reference.
argStruct	A structure containing web service configuration arguments. For more information see Usage

Usage

You can use the CreateObject function to create a web service.

The argStruct structure can contain any combination of the following values:

Name	Default	Description
password	Password set in the Administrator, if any	The password to use to access the web service. If the webservice attribute specifies a web service name configured in the Administrator, overrides any user name specified in the Administrator entry.
port		See portname in the Syntax Parameter table.
proxyPassword	http.proxyPassword system property, if any	The user's password on the proxy server.
proxyPort	http.proxyPort system property, if any	The port to use on the proxy server.
proxyServer	http.proxyHost system property, if any	The proxy server required to access the webservice URL.
proxyUser	http.proxyUser system property, if any	The user ID to send to the proxy server.
refreshWSDL	no	• yes: Reload the WSDL file and regenerate the artifacts used to consume the web service • no
saveJava	no	• yes: Save the Java generated by the WSDL2Java converter that generates Java web server stubs. This code can be useful in debugging errors. • no
timeout	0 (no time-out)	The time-out for retrieving the web service WSDL, in seconds.
username	User name set in the Administrator, if any	The user name to use to access the web service. If the webservice attribute specifies a web service name configured in the Administrator, overrides any user name specified in the Administrator entry.
wsdl2javaArgs		See the Syntax parameter table.

Example

```
<cfscript>
    ws = CreateObject("webservice",
            "http://www.xmethods.net/sd/2001/TemperatureService.wsdl");
    xlatstring = ws.getTemp(zipcode = "55987");
    writeoutput("The temperature at 55987 is " & xlatstring);
</cfscript>
```

CreateODBCDate

Description
Creates an ODBC date object.

Returns
A date object, in normalized ODBC date format.

Category
Date and time functions

Function syntax
`CreateODBCDate(date)`

See also
`CreateDate`, `CreateODBCDateTime`

Parameters

Parameter	Description
`date`	Date or date/time object in the range 100 AD–9999 AD.

Usage
This function does not parse or validate values. To ensure that dates are entered and processed correctly (for example, to ensure that a day/month/year entry is not confused with a month/day/year entry, and so on), Adobe recommends that you parse entered dates with the `DateFormat` function, using the mm-dd-yyyy mask, into three elements. Ensure that values are within appropriate ranges; for example, to validate a month value, use the attributes `validate = "integer"` and `range = "1,12"`.

Example
```
<h3>CreateODBCDate Example</h3>
<cfif IsDefined("form.year")>
<p>Your date value, generated with CreateDateTime:</p>
<cfset yourDate = CreateDateTime(form.year, form.month, form.day, form.hour,
    form.minute, form.second)>
<cfoutput>
<ul>
    <li>Formatted with CreateDate: #CreateDate(form.year, form.month, form.day)#</li>
    <li>Formatted with CreateDateTime:
        #CreateDateTime(form.year, form.month, form.day, form.hour, form.minute,
        form.second)#</li>
    <li>Formatted with CreateODBCDate: #CreateODBCDate(yourDate)#</li>
    <li>Formatted with CreateODBCDateTime: #CreateODBCDateTime(yourDate)#</li>
</ul>
<p>The same value can be formatted with DateFormat:
<ul>
    <li>Formatted with CreateDate and DateFormat:
        #DateFormat(CreateDate(form.year,form.month, form.day), "mmm-dd-yyyy")#</li>
    <li>Formatted with CreateDateTime and DateFormat:
        #DateFormat(CreateDateTime(form.year, form.month, form.day, form.hour,
        form.minute, form.second))#</li>
    <li>Formatted with CreateODBCDate and DateFormat:
        #DateFormat(CreateODBCDate(yourDate), "mmmm d, yyyy")#</li>
    <li>Formatted with CreateODBCDateTime and DateFormat:
        #DateFormat(CreateODBCDateTime(yourDate), "d/m/yy")#</li>
```

```
</ul>
</cfoutput>
</cfif>
<cfform action="createodbcdate.cfm" method="POST">
<p>Enter the year, month and day, as integers:
<pre>
Year    <cfinput type="Text" name="year" value="1998" validate="integer"
        required="Yes">
Month   <cfinput type="Text" name="month" value="6" range="1,12"
        message="Please enter a month (1-12)" validate="integer"
        required="Yes">
Day     <cfinput type="Text" name="day" value="8" range="1,31"
        message="Please enter a day of the month (1-31)" validate="integer"
        required="Yes">
Hour    <cfinput type="Text" NAME="hour" value="16" range="0,23"
        message="You must enter an hour (0-23)" validate="integer"
        required="Yes">
Minute  <cfinput type="Text" name="minute" value="12" range="0,59"
        message="You must enter a minute value (0-59)" validate="integer"
        required="Yes">
Second  <cfinput type="Text" name="second" value="0" range="0,59"
        message="You must enter a value for seconds (0-59)" validate="integer"
        required="Yes">
</pre>
<p><input type="Submit" name=""> <input type="Reset">
</cfform>
```

CreateODBCDateTime

Description
Creates an ODBC date-time object.

Returns
A date-time object, in ODBC timestamp format.

Category
Date and time functions

Function syntax
`CreateODBCDateTime(date)`

See also
`CreateDateTime`, `CreateODBCDate`, `CreateODBCTime`, `Now`; "Evaluation and type conversion issues" on page 39 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`date`	Date-time object in the range 100 AD–9999 AD.

Usage
When passing a date-time value as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date-time object.

Example
```
<!--- This example shows how to use CreateDate, CreateDateTime, CreateODBCDate, and
CreateODBCDateTime --->
<h3>CreateODBCDateTime Example</h3>

<cfif IsDefined("form.year")>
Your date value, generated using CreateDateTime:
<cfet yourDate = CreateDateTime (form.year, form.month, form.day,
    form.hour,form.minute, form.second)>
<cfoutput>
<ul>
    <li>Formatted with CreateDate: #CreateDate(form.year, form.month,form.day)#
    <li>Formatted with CreateDateTime: #CreateDateTime(form.year,form.month,
        form.day,form.hour,form.minute,form.second)#
    <li>Formatted with CreateODBCDate: #CreateODBCDate(yourDate)#
    <li>Formatted with CreateODBCDateTime: #CreateODBCDateTime(yourDate)#
</ul>
<p>The same value can be formatted with DateFormat:
<ul>
    <li>Formatted with CreateDate and DateFormat:
        #DateFormat(CreateDate(form.year,form.month,form.day), "mmm-dd-yyyy")#
    <li>Formatted with CreateDateTime and DateFormat:
        #DateFormat(CreateDateTime(form.year,form.month,form.day,
        form.hour,form.minute,form.second))#
    <li>Formatted with CreateODBCDate and DateFormat:
        #DateFormat(CreateODBCDate(yourDate), "mmmm d, yyyy")#
    <li>Formatted with CreateODBCDateTime and DateFormat:
        #DateFormat(CreateODBCDateTime(yourDate), "d/m/yy")#
```

```
</ul>
</cfoutput>
</cfif>
<CFFORM ACTION="createodbcdatetime.cfm" METHOD="POST">
<p>Enter a year, month and day, as integers:
<PRE>

Year    <CFINPUT
            TYPE="Text"NAME="year" VALUE="1998" VALIDATE="integer" REQUIRED="Yes">

Month   <CFINPUT
            TYPE="Text" NAME="month"VALUE="6"RANGE="1,12"
            MESSAGE="Enter a month (1-12)" VALIDATE="integer" REQUIRED="Yes">

Day     <CFINPUT TYPE="Text" NAME="day" VALUE="8" RANGE="1,31"
            MESSAGE="Enter a day of the month (1-31)" VALIDATE="integer" REQUIRED="Yes">

Hour    <CFINPUT TYPE="Text" NAME="hour" VALUE="16" RANGE="0,23"
            MESSAGE="You must enter an hour (0-23)" VALIDATE="integer" REQUIRED="Yes">

Minute  <CFINPUT TYPE="Text" NAME="minute" VALUE="12" RANGE="0,59"
            MESSAGE="You must enter a minute value (0-59)" VALIDATE="integer"
REQUIRED="Yes">

Second  <CFINPUT TYPE="Text" NAME="second" VALUE="0" RANGE="0,59"
            MESSAGE="You must enter a seconds value (0-59)" VALIDATE="integer"
REQUIRED="Yes">
</PRE>
<p><INPUT TYPE="Submit" NAME=""> <INPUT TYPE="RESET">
</cfform>
```

CreateODBCTime

Description
Creates an ODBC time object.

Returns
A time object, in ODBC timestamp format.

Category
Date and time functions

Function syntax
`CreateODBCTime(date)`

See also
`CreateODBCDateTime`, `CreateTime`, "Evaluation and type conversion issues" on page 39 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`date`	Date/time object in the range 100 AD–9999 AD.

Usage
When passing a date-time value as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date-time object.

Example
```
<h3>CreateODBCTime Example</h3>
<CFIF IsDefined("form.hour")>
Your time value, created with CreateTime...
<CFSET yourTime = CreateTime(form.hour, form.minute, form.second)>
<cfoutput>
<ul>
    <li>Formatted with CreateODBCTime: #CreateODBCTime(yourTime)#
    <li>Formatted with TimeFormat: #TimeFormat(yourTime)#
</ul></cfoutput>
</CFIF>
<CFFORM action="createodbctime.cfm" METHOD="post">
<PRE>
Hour    <CFINPUT TYPE="Text" NAME="hour" VALUE="16" RANGE="0,23" MESSAGE="You must
    enter an hour (0-23)" VALIDATE="integer" REQUIRED="Yes">
Minute  <CFINPUT TYPE="Text" NAME="minute" VALUE="12" RANGE="0,59" MESSAGE="You must
    enter a minute value (0-59)" VALIDATE="integer" REQUIRED="Yes">
Second  <CFINPUT TYPE="Text" NAME="second" VALUE="0" RANGE="0,59" MESSAGE="You must
    enter a value for seconds (0-59)" VALIDATE="integer" REQUIRED="Yes">
</PRE>
<p><INPUT TYPE="Submit" NAME=""> <INPUT TYPE="RESET">
</cfform>
```

CreateTime

Description
Creates a time variable.

Returns
A time variable.

Category
Date and time functions

Function syntax
`CreateTime(hour, minute, second)`

See also
`CreateODBCTime`, `CreateDateTime`; "Evaluation and type conversion issues" on page 39 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`hour`	Number in the range 0–23
`minute`	Number in the range 0–59
`second`	Number in the range 0–59

Usage
`CreateTime` is a subset of `CreateDateTime`.

A time variable is a special case of a date-time variable. The date part of a time variable is set to December 30, 1899.

Example
```
<h3>CreateTime Example</h3>
<cfif IsDefined("FORM.hour")>
Your time value, presented using CreateTime time function:
<cfset yourTime = CreateTime(FORM.hour, FORM.minute, FORM.second)>
<cfoutput><ul>
    <li>Formatted with timeFormat: #TimeFormat(yourTime)#</li>
    <li>Formatted with timeFormat and hh:mm:ss: #TimeFormat(yourTime, 'hh:mm:ss')#</li>
</ul></cfoutput>
</cfif>
<CFFORM action="createtime.cfm" METHOD="post">
<PRE>Hour CFINPUT TYPE="Text" NAME="hour" VALUE="16" RANGE="0,23"
    MESSAGE="You must enter an hour (0-23)" VALIDATE="integer" REQUIRED="Yes">
Minute <CFINPUT TYPE="Text" NAME="minute" VALUE="12" RANGE="0,59"
    MESSAGE="You must enter a minute value (0-59)" VALIDATE="integer"
    REQUIRED="Yes">
Second <CFINPUT TYPE="Text" NAME="second" VALUE="0" RANGE="0,59"
    MESSAGE="You must enter a value for seconds (0-59)" VALIDATE="integer"
    REQUIRED="Yes">
</PRE>
<p><INPUT TYPE="Submit" NAME=""> <INPUT TYPE="RESET"></p>
</cfform>
```

CreateTimeSpan

Description

Creates a date/time object that defines a time period. You can add or subtract it from other date-time objects and use it with the `cachedWithin` attribute of `cfquery`.

Returns

A date-time object.

Category

Date and time functions

Function syntax

`CreateTimeSpan(days, hours, minutes, seconds)`

See also

`CreateDateTime`, `DateAdd`, `DateConvert`; "Defining application-level settings and variables" on page 226 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`days`	Integer in the range 0–32768; number of days in time period
`hours`	Number of hours in time period
`minutes`	Number of minutes in time period
`seconds`	Number of seconds in time period

Usage

Creates a special date-time object that should be used only to add and subtract from other date-time objects or with the `cfquery cachedWithin` attribute.

If you use the `cachedWithin` attribute of `cfquery`, and the original query date falls within the time span you define, cached query data is used. In this case, the `CreateTimeSpan` function is used to define a period of time from the present backwards. The `cachedWithin` attribute takes effect only if you enable query caching in the ColdFusion Administrator. For more information, see `cfquery`.

Example

```
<!--- This example shows the use of CreateTimeSpan with cfquery --->
<h3>CreateTimeSpan Example</h3>
<!--- define startrow and maxrows to facilitate 'next N' style browsing --->
<CFPARAM name = "MaxRows" default = "10">
<CFPARAM name = "StartRow" default = "1">
<!--- Query database for information, if cached database information has not been updated
in the last six hours. -------->
<cfoutput>
<cfquery name = "GetParks" datasource = "cfdocexamples"
    cachedWithin = "#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT PARKNAME, REGION, STATE
    FROM Parks
    ORDER by ParkName, State
</cfquery>
</cfoutput>
<!--- build HTML table to display query --->
```

```
<TABLE cellpadding = 1 cellspacing = 1>
<TR>
    <TD colspan = 2 bgcolor = f0f0f0>
    <B><I>Park Name</I></B>
    </TD>
    <TD bgcolor = f0f0f0>
    <B><i>Region</I></B>
    </TD>
    <TD bgcolor = f0f0f0>
    <B><I>State</I></B>
    </TD>
</TR>
<!--- Output query, define startrow and maxrows. Use query variable CurrentCount to track
the row you are displaying. --->
<cfoutput query = "GetParks" StartRow = "#StartRow#"
    MAXROWS = "#MaxRows#">
<TR>
    <TD valign = top bgcolor = ffffed>
    <B>#GetParks.CurrentRow#</B>
    </TD>
    <TD valign = top>
    <FONT SIZE = "-1">#ParkName#</FONT>
    </TD>
    <TD valign = top>
    <FONT SIZE = "-1">#Region#</FONT>
    </TD>
    <TD valign = top>
    <FONT SIZE = "-1">#State#</FONT>
    </TD>
</TR>
</cfoutput>
<!--- If number of records is less than or equal to number of rows, offer link to same page,
with startrow value incremented by maxrows (in this example,
    incremented by 10). --->
<TR>
    <TD colspan = 4>
    <cfif (StartRow + MaxRows) LTE GetParks.RecordCount>
        <a href = "cfquery.cfm?startrow = <cfoutput>#StartRow + MaxRows#
        </cfoutput>">See next <cfoutput>#MaxRows#</cfoutput> rows</A>
    </cfif>
    </TD>
</TR>
</TABLE>
```

CreateUUID

Description

Creates a Universally Unique Identifier (UUID). A UUID is a 35-character string representation of a unique 128-bit integer.

Returns

A ColdFusion format UUID, in the format `xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx`, where `x` is a hexadecimal digit (0-9 or A-F). (The character groups are 8-4-4-16.)

Category

Other functions

Function syntax

```
CreateUUID()
```

Usage

The ColdFusion UUID generation algorithm uses the unique time-of-day value, the IEEE 802 Host ID, and a cryptographically strong random number generator to generate UUIDs that conform to the principles laid out in the draft IEEE RFC "*UUIDs and GUIDs.*"

The ColdFusion UUID format is as follows:

`xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx` (8-4-4-16).

This does not conform to the Microsoft/DCE standard, which is as follows:

`xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` (8-4-4-4-12)

There are UUID test tools and a user-defined function called CreateGUID, which converts CFML UUIDs to UUID/Microsoft GUID format, available on the web at www.cflib.org.

Use this function to generate a persistent identifier in a distributed environment. To a very high degree of certainty, this function returns a unique value; no other invocation on the same or any other system returns the same value.

UUIDs are used by distributed computing frameworks, such as DCE/RPC, COM+, and CORBA. In ColdFusion, you can use UUIDs as primary table keys for applications in which data is stored in shared databases. In such cases, using numeric keys can cause primary-key constraint violations during table merges. Using UUIDs, you can eliminate these violations.

Example

```
<h3>CreateUUID Example</h3>
<p> This example uses CreateUUID to generate a UUID when you submit the form.
    You can submit the form more than once. </p>
<!--- Checks whether the form was submitted; if so, creates UUID. --->
<cfif IsDefined("Form.CreateUUID") Is True>
    <hr>
    <p>Your new UUID is: <cfoutput>#CreateUUID()#</cfoutput></p>
</cfif>
<form action = "createuuid.cfm">
<p><input type = "Submit" name = "CreateUUID"> </p>
</form>
```

DateAdd

Description
Adds units of time to a date.

Returns
A date/time object.

Category
Date and time functions

Function syntax
```
DateAdd("datepart", number, "date")
```

See also
DateConvert, DatePart, CreateTimeSpan

History
ColdFusion MX 6.1: Added the *datepart* character L or l to represent milliseconds.

Parameters

Parameter	Description
datepart	String: • yyyy: Year • q: Quarter • m: Month • y: Day of year • d: Day • w: Weekday • ww: Week • h: Hour • n: Minute • s: Second • l: Millisecond
number	Number of units of *datepart* to add to *date* (positive, to get dates in the future; negative, to get dates in the past). Number must be an integer.
date	Date/time object, in the range 100 AD–9999 AD.

Usage
The *datepart* specifiers y, d, and w add a number of days to a date.

When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Example
```
<!--- This example shows the use of DateAdd --->
<cfparam name="value" default="70">
```

```
<cfparam name="type" default="m">

<!--- If numbers passed, then use those. --->
<cfif IsDefined("form.value")>
    <cfset value = form.value>
</cfif>
<cfif IsDefined("form.type")>
    <cfset type = form.type>
</cfif>


<cfquery name="GetMessages" datasource="cfdocexamples">
    SELECT UserName, Subject, Posted
    FROM Messages
</cfquery>

<p>This example uses DateAdd to determine when a message in
the database will expire. Currently, messages older
than <cfoutput>#value#</cfoutput>

<cfswitch expression="#type#">
    <cfcase value="yyyy">years</cfcase>
    <cfcase value="q">quarters</cfcase>
    <cfcase value="m">months</cfcase>
    <cfcase value="y">days of year</cfcase>
    <cfcase value="w">weekdays</cfcase>
    <cfcase value="ww">weeks</cfcase>
    <cfcase value="h">hours</cfcase>
    <cfcase value="n">minutes</cfcase>
    <cfcase value="s">seconds</cfcase>
    <cfdefaultcase>years</cfdefaultcase>
</cfswitch>
 are expired.

<table>
<tr>
    <td>UserName</td>
    <td>Subject</td>
    <td>Posted</td>
</tr>
<cfoutput query="GetMessages">
<tr>
    <td>#UserName#</td>
    <td>#Subject#</td>
    <td>#Posted# <cfif DateAdd(type, value, posted) LT Now()><font
color="red">EXPIRED</font></cfif></td>
</tr>
</cfoutput>
</table>

<cfform action="#CGI.Script_Name#" method="post">

Select an expiration value:
<cfinput type="Text" name="value" value="#value#" message="Please enter whole numbers only"
validate="integer" required="Yes">
<select name="type">
    <option value="yyyy">years
    <option value="m" selected>months
    <option value="d">days
    <option value="ww">weeks
    <option value="h">hours
```

```
    <option value="n">minutes
    <option value="s">seconds
</select>

<input type="Submit" value="Submit">
</cfform>
```

DateCompare

Description

Performs a full date/time comparison of two dates.

Returns

- -1, if *date1* is earlier than *date2*
- 0, if *date1* is equal to *date2*
- 1, if *date1* is later than *date2*

Category

Date and time functions

Function syntax

```
DateCompare("date1", "date2" [, "datePart"])
```

See also

CreateDateTime, DatePart

Parameters

Parameter	Description
date1	Date/time object, in the range 100 AD–9999 AD.
date2	Date/time object, in the range 100 AD–9999 AD.
datePart	Optional. String. Precision of the comparison. • s Precise to the second (default) • n Precise to the minute • h Precise to the hour • d Precise to the day • m Precise to the month • yyyy Precise to the year

Usage

When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Example

```
<h3>DateCompare Example</h3>
<p>The DateCompare function compares two date/time values.
<cfif IsDefined("FORM.date1")>
    <cfif IsDate(FORM.date1) and IsDate(FORM.date2)>
        <cfset comparison = DateCompare(FORM.date1, FORM.date2, FORM.precision)>

<!--- Switch on the variable to give various responses. --->
        <cfswitch expression = #comparison#>
            <cfcase value = "-1">
                <h3><cfoutput>#DateFormat(FORM.date1)#
                #TimeFormat(FORM.date1)#</cfoutput> (Date 1) is
                earlier than <cfoutput>#DateFormat(FORM.date2)#
                #TimeFormat(FORM.date2)#</cfoutput> (Date 2)</h3>
```

```
            <I>The dates are not equal</I>
        </cfcase>
        <cfcase value = "0">
            <h3><cfoutput>#DateFormat(FORM.date1)#
            #TimeFormat(FORM.date1)#</cfoutput> (Date 1) is equal
            to <cfoutput>#DateFormat(FORM.date2)#
            #TimeFormat(FORM.date2)#</cfoutput> (Date 2)</h3>
            <I>The dates are equal!</I>
        </cfcase>
        <cfcase value = "1">
            <h3><cfoutput>#DateFormat(FORM.date1)#
            #TimeFormat(FORM.date1)#</cfoutput> (Date 1) is later
            than <cfoutput>#DateFormat(FORM.date2)#
            #TimeFormat(FORM.date2)#</cfoutput> (Date 2)</h3>
            <I>The dates are not equal</I>
        </cfcase>
        <CFDEFAULTCASE>
            <h3>This is the default case</h3>
        </CFDEFAULTCASE>
    </cfswitch>
<cfelse>
    <h3>Enter two valid date values</h3>
</cfif>
</cfif>

<form action = "datecompare.cfm" method="post">
<hr size = "2" color = "#0000A0">
<p>Date 1
<br><input type = "Text" name = "date1"
    value = "<cfoutput>#DateFormat(Now())# #TimeFormat(Now())#
</cfoutput>">
<p>Date 2
<br><input type = "Text" name = "date2"
    value = "<cfoutput>#DateFormat(Now())# #TimeFormat(Now())#
</cfoutput>">
<p>Specify precision to the:
<br><select name = "precision">
        <option value = "s">
            Second
        </option>
        <option value = "n">
            Minute
        </option>
        <option value = "h">
            Hour
        </option>
        <option value = "d">
            Day
        </option>
        <option value = "m">
            Month
        </option>
        <option value = "yyyy">
            Year
        </option>
</select>
<p><input type = "Submit" value = "Compare these dates" name = "">
<input type = "reset">
</form>
```

DateConvert

Description

Converts local time to Coordinated Universal Time (UTC), or UTC to local time. The function uses the daylight savings settings in the executing computer to compute daylight savings time, if required.

Returns

UTC- or local-formatted time object.

Category

Date and time functions

Function syntax

`DateConvert("conversion-type", "date")`

See also

`GetTimeZoneInfo`, `CreateDateTime`, `DatePart`

Parameters

Parameter	Description
`conversion-type`	• local2Utc: Converts local time to UTC time. • utc2Local: Converts UTC time to local time.
`date`	Date and time string or a variable that contains one. To create, use `CreateDateTime`.

Usage

When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Note: *You can pass the `CreateDate` or `Now` function as the date parameter of this function; for example:*

```
#DateConvert(CreateDate(2007, 4, 10))#
```

Example

```
<h3>DateConvert Example</h3>
<!--- This shows conversion of current date - time to UTC and back. --->
<cfset curDate = Now()>
<p><cfoutput>The current date and time: #curDate#. </cfoutput></p>
<cfset utcDate = DateConvert("local2utc", curDate)>
<cfoutput>
    <p>The current date and time converted to UTC time: #utcDate#.</p>
</cfoutput>
<!--- This code checks whether form was submitted. If so, the code generates
    the CFML date with the CreateDateTime function. --->
<cfif IsDefined("FORM.submit")>
    <cfset yourDate = CreateDateTime(FORM.year, FORM.month, FORM.day,
        FORM.hour,FORM.minute, FORM.second)>
    <p><cfoutput>Your date value, presented as a ColdFusion date/time
        string:#yourdate#.</cfoutput></p>
<cfset yourUTC = DateConvert("local2utc", yourDate)>
    <p><cfoutput>Your date and time value, converted to Coordinated Universal Time
    (UTC): #yourUTC#.</cfoutput></p>
    <p><cfoutput>Your UTC date and time, converted back to local date and time:
```

```
    #DateConvert("utc2local", yourUTC)#.
    </cfoutput></p>
<cfelse>
    Type the date and time, and press Enter to see the conversion.
</cfif>
<Hr size = "2" color = "#0000A0">
<form action = "dateconvert.cfm">
<p>Enter year, month and day in integer format for date value to view:
<table cellspacing = "2" cellpadding = "2" border = "0">
<tr>
<td>Year</td>
<td><input type = "Text" name = "year" value = "1998"
        validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Month</td>
<td><input type = "Text" name = "month" value = "6"
        range = "1,12" message = "Enter a month (1-12)"
        validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Day</td>
<td><input type = "Text" name = "day" value = "8"
        range = "1,31"
        message = "Enter a day of the month (1-31)"
        validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Hour</td>
<td><input type = "Text" name = "hour" value = "16"
        range = "0,23"
        message = "You must enter an hour (0-23)"
        validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Minute</td>
<td><input type = "Text" name = "minute" value = "12"
        range = "0,59"
        message = "You must enter a minute value (0-59)"
        validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Second</td>
<td><input type = "Text" name = "second" value = "0"
        range = "0,59"
        message = "You must enter a value for seconds (0-59)"
        validate = "integer" required = "Yes"></td></tr>
<tr>
<td><input type = "Submit" name = "submit" value = "Submit"></td>
<td><input type = "RESET"></td></tr>
</table>
```

DateDiff

Description
Determines the integer number of units by which *date1* is less than *date2*.

Returns
A number of units, of type *datepart*.

Category
Date and time functions

Function syntax
```
DateDiff("datepart", "date1", "date2")
```

See also
DateAdd, DatePart, CreateTimeSpan

History
ColdFusion MX:

- Changed how negative date differences are calculated: this function calculates negative date differences correctly; its output may be different from that in earlier releases.
- Changed the w and ww masks; they determine the number of full weeks between the two dates.

Parameters

Parameter	Description
datepart	String that specifies the units in which to count; for example yyyy requests a date difference in whole years. • yyyy: Years • q: Quarters • m: Months • y: Days of year (same as d) • d: Days • w: Weekdays (same as ww) • ww: Weeks • h: Hours • n: Minutes • s: Seconds
date1	Date/time object, in the range 100 AD–9999 AD.
date2	Date/time object, in the range 100 AD–9999 AD.

Usage
The DateDiff function determines the number of complete *datepart* units between the two dates; for example, if the *datepart* parameter is "m" and the dates differ by 55 days, the function returns 1.

Enclose string constant dates in quotation marks. If the text contains only numbers (such 1932), and is not surrounded by quotation marks, ColdFusion interprets it as a date/time object, resulting in an incorrect value.

Example

```
<cfif IsDefined("form.value")>
    <cfset value = form.value>
</cfif>
<cfif IsDefined("form.type")>
    <cfset type = form.type>
</cfif>

<cfif IsDefined("form.date1") and IsDefined("form.date2")>

    <cfif IsDate(form.date1) and IsDate(form.date2)>

        <p>This example uses DateDiff to determine the difference
        in
        <cfswitch expression = "#form.type#">
            <cfcase value="yyyy">years</cfcase>
            <cfcase value="q">quarters</cfcase>
            <cfcase value="m">months</cfcase>
            <cfcase value="y">days</cfcase>
            <cfcase value="d">days</cfcase>
            <cfcase value="w">weekdays</cfcase>
            <cfcase value="ww">weeks</cfcase>
            <cfcase value="h">hours</cfcase>
            <cfcase value="n">minutes</cfcase>
            <cfcase value="s">seconds</cfcase>
            <cfdefaultcase>years</cfdefaultcase>
        </cfswitch>
             dateparts between date1 and date2.

        <cfif DateCompare("#form.date1#","#form.date2#") is not 0>
        <p>The difference is <cfoutput>#Abs(DateDiff(type, form.date2,
form.date1))#</cfoutput>
        <cfswitch expression = "#form.type#">
            <cfcase value="yyyy">years</cfcase>
            <cfcase value="q">quarters</cfcase>
            <cfcase value="m">months</cfcase>
            <cfcase value="y">days</cfcase>
            <cfcase value="d">days</cfcase>
            <cfcase value="w">weekdays</cfcase>
            <cfcase value="ww">weeks</cfcase>
            <cfcase value="h">hours</cfcase>
            <cfcase value="n">minutes</cfcase>
            <cfcase value="s">seconds</cfcase>
            <cfdefaultcase>years</cfdefaultcase>
        </cfswitch>.
        <cfelse>
        <p>The two dates are equal!Try changing one of the values ...
        </cfif>

    <cfelse>
    <p>Please enter two valid date/time values, formatted like this:
    <cfoutput>#DateFormat(Now())#</cfoutput>
    </cfif>

</cfif>
<form action="index.cfm" method="post">

<pre>
Date 1
<input type="Text" name="date1" value="<cfoutput>#DateFormat(Now())#</cfoutput>">
Date 2
```

```
<input type="Text" name="date2" value="<cfoutput>#DateFormat(Now())#</cfoutput>">
What kind of unit to show difference?
    <select name="type">
        <option value="yyyy" selected>years
        <option value="q">quarters
        <option value="m">months
        <option value="y">days of year
        <option value="d">days
        <option value="w">weekdays
        <option value="ww">weeks
        <option value="h">hours
        <option value="n">minutes
        <option value="s">seconds
    </select>
</pre>

<input type="Submit" name=""><input type="Reset">
</form>
```

DateFormat

Description

Formats a date value using U.S. date formats. For international date support, use `LSDateFormat`.

Returns

A text string representing the date formatted according to the mask. If no mask is specified, returns the value in *dd-mmm-yy* format.

Category

Date and time functions

Function syntax

```
DateFormat("date" [, "mask" ])
```

See also

`Now`, `CreateDate`, `LSDateFormat`, `LSParseDateTime`, `LSTimeFormat`, `TimeFormat`, `ParseDateTime`

History

ColdFusion MX: Added support for the following `mask` parameter options: `short`, `medium`, `long`, and `full`.

Parameters

Parameter	Description
`date`	Date/time object, in the range 100 AD–9999 AD.
`mask`	Characters that show how ColdFusion displays a date: • d: Day of the month as digits; no leading zero for single-digit days. • dd: Day of the month as digits; leading zero for single-digit days. • ddd: Day of the week as a three-letter abbreviation. • dddd: Day of the week as its full name. • m: Month as digits; no leading zero for single-digit months. • mm: Month as digits; leading zero for single-digit months. • mmm: Month as a three-letter abbreviation. • mmmm: Month as its full name. • yy: Year as last two digits; leading zero for years less than 10. • yyyy: Year represented by four digits. • gg: Period/era string. Ignored. Reserved. The following masks tell how to format the full date and cannot be combined with other masks: • short: equivalent to m/d/y • medium: equivalent to mmm d, yyyy • long: equivalent to mmmm d, yyyy • full: equivalent to dddd, mmmm d, yyyy

Usage

When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Note: *You can pass the* `CreateDate` *function or* `Now` *function as the date parameter of this function; for example:*
`#DateFormat(CreateDate(2001, 3, 3))#`

Date and time values in database query results can vary in sequence and formatting unless you use functions to format them. To ensure that application users correctly understand displayed dates and times, Adobe recommends that you use this function and the `LSDateFormat`, `TimeFormat`, and `LSTimeFormat` functions to format resultset values. For more information and examples, see TechNote 22183, "*ColdFusion Server (5 and 4.5.x) with Oracle: Formatting Date and Time Query Results*," on the website at www.coldfusion.com/Support/KnowledgeBase/SearchForm.cfm.

Note: *The* `DateFormat` *function is best used for formatting output, not for formatting input. For formatting input, use one of the date/time creation functions (for example,* `CreateDate`*) instead.*

Example

```
<cfset todayDate = Now()>
<body>
<h3>DateFormat Example</h3>
<p>Today's date is <cfoutput>#todayDate#</cfoutput>.
<p>Using DateFormat, we can display that date in different ways:
<cfoutput>
<ul>
    <li>#DateFormat(todayDate)#
    <li>#DateFormat(todayDate, "mmm-dd-yyyy")#
    <li>#DateFormat(todayDate, "mmmm d, yyyy")#
    <li>#DateFormat(todayDate, "mm/dd/yyyy")#
    <li>#DateFormat(todayDate, "d-mmm-yyyy")#
    <li>#DateFormat(todayDate, "ddd, mmmm dd, yyyy")#
    <li>#DateFormat(todayDate, "short")#
    <li>#DateFormat(todayDate, "medium")#
    <li>#DateFormat(todayDate, "long")#
    <li>#DateFormat(todayDate, "full")#
</ul>
</cfoutput>
```

DatePart

Description
Extracts a part from a date value.

Returns
Part of a date, as an integer.

Category
Date and time functions

Function syntax
```
DatePart("datepart", "date")
```

See also
DateAdd, DateConvert

History
ColdFusion MX 6.1: Added the *datepart* character L or l to represent milliseconds.

Parameters

Parameter	Description
datepart	String: • yyyy: Year • q: Quarter • m: Month • y: Day of year • d: Day • w: Weekday • ww: Week • h: Hour • n: Minute • s: Second • l: Millisecond
date	Date/time object, in the range 100 AD–9999 AD.

Usage
When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Example
```
<!--- This example shows information available from DatePart --->
<cfset todayDate = Now()>
<h3>DatePart Example</h3>
<p>Today's date is <cfoutput>#todayDate#</cfoutput>.
<p>Using datepart, we extract an integer representing the dateparts from that value
<cfoutput>
```

```
<ul>
    <li>year: #DatePart("yyyy", todayDate)#</li>
    <li>quarter: #DatePart("q", todayDate)#</li>
    <li>month: #DatePart("m", todayDate)#</li>
    <li>day of year: #DatePart("y", todayDate)#</li>
    <li>day: #DatePart("d", todayDate)#</li>
    <li>weekday: #DatePart("w", todayDate)#</li>
    <li>week: #DatePart("ww", todayDate)#</li>
    <li>hour: #DatePart("h", todayDate)#</li>
    <li>minute: #DatePart("n", todayDate)#</li>
    <li>second: #DatePart("s", todayDate)#</li>
</ul>
</cfoutput>
```

Day

Description
Determines the day of the month, in a date.

Returns
The ordinal for the day of the month, ranging from 1 to 31.

Category
Date and time functions

Function syntax
`Day("date")`

See also
DayOfWeek, DayOfWeekAsString, DayOfYear, DaysInMonth, DaysInYear, FirstDayOfMonth

Parameters

Parameter	Description
date	Date/time object, in the range 100 AD–9999 AD.

Usage
When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Note: *You can pass the CreateDate function or Now function as the date parameter of this function; for example:*
`#Day(CreateDate(2001, 3, 3))#`

Example
```
<h3>Day Example</h3>
<cfif IsDefined("FORM.year")>
    <p>More information about your date:
    <cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
    <cfoutput>
        <p>Date: #DateFormat(yourDate)#. <br>
    It is #DayofWeekAsString(DayOfWeek(yourDate))#,
    day #DayOfWeek(yourDate)# in the week.
        <br>This is day #Day(YourDate)#
    in the month of #MonthAsString(Month(yourDate))#,
    which has #DaysInMonth(yourDate)# days.
        <br>We are in week #Week(yourDate)# of #Year(YourDate)#
        (day #DayofYear(yourDate)# of #DaysinYear(yourDate)#).
        <br><cfif IsLeapYear(Year(yourDate))>This is a leap year
                <cfelse>This is not a leap year</cfif>
    </cfoutput>
</cfif>
```

DayOfWeek

Description
Determines the day of the week, in a date.

Returns
The ordinal for the day of the week, as an integer in the range 1 (Sunday) to 7 (Saturday).

Category
Date and time functions

Function syntax
```
DayOfWeek("date")
```

See also
Day, DayOfWeekAsString, DayOfYear, DaysInMonth, DaysInYear, FirstDayOfMonth

Parameters

Parameter	Description
date	Date/time object, in the range 100 AD–9999 AD.

Usage
When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Note: *You can pass the CreateDate function or Now function as the date parameter of this function; for example,*
```
#DayOfWeek(CreateDate(2001, 3, 3))#
```

Example
```
<h3>DayOfWeek Example</h3>
<cfif IsDefined("FORM.year")>
    More information about your date:
    <cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
    <cfoutput>
    <p>Your date, #DateFormat(yourDate)#.
    <br>It is #DayofWeekAsString(DayOfWeek(yourDate))#, day
         #DayOfWeek(yourDate)# in the week.
    <br>This is day #Day(YourDate)# in the month of
         #MonthAsString(Month(yourDate))#, which has
     #DaysInMonth(yourDate)# days.
    <br>We are in week #Week(yourDate)# of #Year(YourDate)# (day
     #DayofYear(yourDate)# of #DaysinYear(yourDate)#).
    <br><cfif IsLeapYear(Year(yourDate))>This is a leap year
        <cfelse>This is not a leap year</cfif>
    </cfoutput>
</cfif>
```

DayOfWeekAsString

Description

Determines the day of the week, in a date, as a string function.

Returns

The day of the week, as a string in the current locale, that corresponds to *day_of_week*.

Category

Date and time functions, String functions

Function syntax

```
DayOfWeekAsString(day_of_week [, locale])
```

See also

Day, DayOfWeek, DayOfYear, DaysInMonth, DaysInYear, FirstDayOfMonth

History

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX 7: Changed behavior. The returned string is now in the language of the current locale.

Parameters

Parameter	Description
`day_of_week`	Integer, in the range 1 (Sunday) - 7 (Saturday).
`locale`	Locale to use instead of the locale of the page when processing the function

Example

The following example shows the use of the `DayOfWeekAsString` function. It is the action page for a form that submits year, month, and day fields.

```
<h3>DayOfWeekAsString Example</h3>
<cfif IsDefined("FORM.year")>
More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>

<cfoutput>
<p>Your date, #DateFormat(yourDate)#.
<br>It is #DayofWeekAsString(DayOfWeek(yourDate))#, day
     #DayOfWeek(yourDate)# in the week.
<br>This is day #Day(YourDate)# in the month of
    #MonthAsString(Month(yourDate))#, which has
     #DaysInMonth(yourDate)# days.
<br>We are in week #Week(yourDate)# of #Year(YourDate)# (day #DayofYear(yourDate)#
    of #DaysinYear(yourDate)#).
<br><cfif IsLeapYear(Year(yourDate))>This is a leap year
    <cfelse>This is not a leap year</cfif>
</cfoutput>
</cfif>
```

DayOfYear

Description
Determines the day of the year, in a date.

Returns
The ordinal value of day of the year, as an integer.

Category
Date and time functions

Function syntax
```
DayOfYear("date")
```

See also
Day, DayOfWeek, DayOfWeekAsString, DaysInMonth, DaysInYear, FirstDayOfMonth

Parameters

Parameter	Description
date	Date/time object, in the range 100 AD–9999 AD.

Usage
This function accounts for leap years.

When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Note: *You can pass the CreateDate function or Now function as the date parameter of this function; for example,* `#DayOfYear(CreateDate(2001, 3, 3))#`

Example
```
<h3>Day7OfYear Example</h3>
<cfif IsDefined("FORM.year")>
    More information about your date:
    <cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
    <cfoutput>
    <p>Your date, #DateFormat(yourDate)#.
    <br>It is #DayofWeekAsString(DayOfWeek(yourDate))#,
         day #DayOfWeek(yourDate)# in the week.
    <br>This is day #Day(yourDate)# in the month of
     #MonthAsString(Month(yourDate))#, which has
     #DaysInMonth(yourDate)# days.
    <br>We are in week #Week(yourDate)# of #Year(yourDate)#
    (day #DayofYear(yourDate)# of #DaysinYear(yourDate)#).
    <br><cfif IsLeapYear(Year(yourDate))>This is a leap year
        <cfelse>This is not a leap year</cfif>
    </cfoutput>
</cfif>
```

DaysInMonth

Description
Determines the number of days in a month.

Returns
The number of days in the month in *Date*.

Category
Date and time functions

Function syntax
```
DaysInMonth("date")
```

See also
Day, DayOfWeek, DayOfWeekAsString, DayOfYear, DaysInYear, FirstDayOfMonth

Parameters

Parameter	Description
date	Date/time object, in the range 100 AD–9999 AD.

Usage
When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Note: *You can pass the Now function or the CreateDate function as the date parameter of this function; for example:*
```
#DaysInMonth(CreateDate(2001, 3, 3))#
```

Example
```
<h3>DaysInMonth Example</h3>
<cfif IsDefined("FORM.year")>
    More information about your date:
    <cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
    <cfoutput>
    <p>Your date, #DateFormat(yourDate)#.
    <br>It is #DayofWeekAsString(DayOfWeek(yourDate))#, day
         #DayOfWeek(yourDate)# in the week.
    <br>This is day #Day(YourDate)# in the month of
         #MonthAsString(Month(yourDate))#, which has
     #DaysInMonth(yourDate)# days.
    <br>We are in week #Week(yourDate)# of #Year(YourDate)#
     (day #DayofYear(yourDate)# of #DaysinYear(yourDate)#).
    <br><cfif IsLeapYear(Year(yourDate))>This is a leap year
        <cfelse>This is not a leap year</cfif>
    </cfoutput>
</cfif>
```

DaysInYear

Description
Determines the number of days in a year.

Returns
The number of days in a year.

Category
Date and time functions

Function syntax
```
DaysInYear("date")
```

See also
Day, DayOfWeek, DayOfWeekAsString, DayOfYear, DaysInMonth, DaysInYear, FirstDayOfMonth, IsLeapYear

Parameters

Parameter	Description
date	Date/time object, in the range 100 AD–9999 AD.

Usage
DaysInYear accounts for leap years.

When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

Note: *You can pass the CreateDate function or the Now function as the date parameter of this function; for example:*
```
#DaysInYear(CreateDate(2001, 3, 3))#
```

Example
```
<h3>DaysInYear Example</h3>
<cfif IsDefined("FORM.year")>
    More information about your date:
    <cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
    <cfoutput>
    <p>Your date, #DateFormat(yourDate)#.
    <br>It is #DayofWeekAsString(DayOfWeek(yourDate))#, day
         #DayOfWeek(yourDate)# in the week.
    <br>This is day #Day(YourDate)# in the month of
         #MonthAsString(Month(yourDate))#, which has
     #DaysInMonth(yourDate)# days.
    <br>We are in week #Week(yourDate)# of #Year(yourDate)# (day
     #DayofYear(yourDate)# of #DaysinYear(yourDate)#).
</cfif>
```

DE

Description
Escapes any double-quotation marks in the parameter and wraps the result in double-quotation marks.

Returns
Parameter, surrounded by double-quotation marks, with any inner double-quotation marks escaped.

Category
Dynamic evaluation functions

Function syntax
DE(*string*)

See also
Evaluate, IIf, PrecisionEvaluate, "Using Expressions and Number Signs" on page 50 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
string	String to evaluate, after delay

Usage
The DE function postpones evaluation of a string that is passed as a parameter to the IIf, Evaluate, or PrecisionEvaluate functions.

This function is especially useful with the IIf function, which automatically evaluates its second and third parameters as expressions. You can use the DE function to prevent the function from evaluating a string parameter that is to be output as a variable, and should not be treated as an expression. The following example show this use; it uses IIF to alternate table-row background colors, white and gray, and uses the DE function to prevent ColdFusion from evaluating the color strings.

```
<cfoutput>
<table border="1" cellpadding="3">
<cfloop index="i" from="1" to="10">
    <tr bgcolor="#IIF( i mod 2 eq 0, DE("white"), DE("gray") )#">
        <td>
            hello #i#
        </td>
    </tr>
</cfloop>
</table>
</cfoutput>
```

The DE function does not delay evaluation of variable names that are surrounded by number signs (#). The ColdFusion function evaluates the variable regardless of whether the DE function is present.

The following example shows how you can use the DE function and number signs together, and shows how the function works with an IIF function:

```
<cfoutput>
<cfset var1=Blue>
<cfset var2=Green>
<cfset myresult=IIf( 1 eq 2, DE(#Var1#), DE(#Var2#))>
```

```
The expression is #myresult#
</cfoutput>
```

ColdFusion processes this code as follows:

1 ColdFusion sets the variables `var1` and `var2` to be the strings Blue and Green.

2 In the fourth line, ColdFusion evaluates the variables surrounded by number signs first, replacing them with the strings Blue and Green, the values of the variables.

3 The `IIF` function evaluates the test expression, determines that it is False, and then evaluates the third parameter.

4 The third parameter is a `DE` function, which takes the string Green and surrounds it in quotation marks

5 The `IIF` function returns the string "Green", including the quotation marks.

6 The `cfset` tag gets the expression result="Green", and sets the value of the `myresult` variable to the string Green.

7 ColdFusion evaluates `#myresult#` in the output text, replaces the variable with its value, the string Green, and displays the result.

Example

```
<!--- This example shows the use of DE and Evaluate --->
<h3>DE Example</h3>
<cfif IsDefined("FORM.myExpression")>
    <cftry>
        <!--- Show the expression and the results of evaluating it --->
        <cfoutput>
            <h3>Evaluate the Expression #FORM.MyExpression#</h3>
        </cfoutput>
        The code:<br>
        #Evaluate(FORM.myExpression)#
        <br><br>
        The result:<br>
        <cfoutput>
            #Evaluate(FORM.myExpression)#
        </cfoutput>

        <h3>Use DE to prevent the Evaluate function from evaluating</h3>
        The code:<br>
        #Evaluate(DE(FORM.MyExpression))#<br><br>
        The result:<br>
        <cfoutput>
            #Evaluate(DE(FORM.MyExpression))#
        </cfoutput>
        <!--- Error handling code for bad expressions and any other error.--->
        <cfcatch type = "Any">
        <!--- the message to display --->
            <h3>Sorry, there's been an <B>Error</B>.
            Try a simple expression, such as "2+2".</h3>
            <cfoutput>
            <!--- Display the diagnostic message from ColdFusion. --->
                <p>#cfcatch.message#
            </cfoutput>
        </cfcatch>
    </cftry>
</cfif>

<h3>Enter any valid ColdFusion expression</h3>
<cfform>
    <cfinput name="myExpression" type="Text" size="40">
```

```
    <cfinput type="submit" name="submitit">
</cfform>
```

DecimalFormat

Description

Converts a number to a decimal-formatted string.

Returns

A *number* as a string formatted with two decimal places and a thousands separator.

Category

Display and formatting functions

Function syntax

`DecimalFormat(number)`

See also

`DollarFormat`, `NumberFormat`

Parameters

Parameter	Description
`number`	Number to format

Example

```
<h3>DecimalFormat Function</h3>
<p>Returns a number to two decimal places.
<p>
<cfloop FROM = 1 TO = 20 INDEX = "counter">
    <cfoutput>
    #counter# * Square Root of 2:
        #DecimalFormat(counter * sqr(2))#
    </cfoutput>
    <br>
</cfloop>
```

DecrementValue

Description

Decrements the integer part of a number.

Returns

Integer part of *number,* decremented by one.

Category

Mathematical functions

Function syntax

`DecrementValue(number)`

See also

`IncrementValue`

Parameters

Parameter	Description
`number`	Number to decrement

Example

```
<h3>DecrementValue Example</h3>
<p>Returns the integer part of a number decremented by one.</p>
<p>DecrementValue(0):
    <cfoutput>#DecrementValue(0)#</cfoutput></p>
<p>DecrementValue("1"):
    <cfoutput>#DecrementValue("1")#</cfoutput></p>
<p>DecrementValue(123.35):
    <cfoutput>#DecrementValue(123.35)#</cfoutput></p>
```

Decrypt

Description

Decrypts a string that is encrypted using a standard encryption technique, including strings encrypted by the `Encrypt` function.

Returns

An unencrypted string.

Category

Security functions, String functions

Function syntax

`Decrypt(encrypted_string, key[, algorithm, encoding, IVorSalt, iterations])`

See also

`Duplicate`, `Encrypt`

History

ColdFusion 8: Added support for encryption using the RSA BSafe Crypto-J library on Enterprise Edition.

ColdFusion MX 7.01: Added the *`IVorSalt`* and *`iterations`* parameters.

ColdFusion MX 7: Added the *`algorithm`* and *`encoding`* parameters.

Parameters

Parameter	Description
`encrypted_string`	String to decrypt.
`key`	String. For the CFMX_COMPAT algorithm, the seed that was used to encrypt the string; for all other algorithms, the string generated by the `generateSecretKey()` method.
`algorithm`	(Optional) The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. For a list of algorithms, see the Encrypt function. The Standard Edition of ColdFusion installs a cryptography library with the following algorithms: • CFMX_COMPAT: the algorithm used in ColdFusion MX 7 and prior releases. This algorithm is the least secure option (default). • AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197. • BLOWFISH: the Blowfish algorithm defined by Bruce Schneier. • DES: the Data Encryption Standard algorithm defined by NIST FIPS-46-3. • DESEDE: the "Triple DES" algorithm defined by NIST FIPS-46-3. If you install a security provider with additional cryptography algorithms, you can also specify any of its string encryption and decryption algorithms.
`encoding`	(Optional; if you specify this parameter, you must also specify the *`algorithm`* parameter.) The binary encoding used to represent the data as a string. Must be the same as the algorithm used to encrypt the string. • Base64: the Base64 algorithm, as specified by IETF RFC 2045. • Hex: the characters A-F and 0-9 represent the hexadecimal byte values. • UU: the UNIX standard UUEncode algorithm (default).

Parameter	Description
IVorSalt	(Optional) Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, you must also specify the algorithm parameter. • For Block Encryption Algorithms: This is the binary Initialization Vector value to use with the algorithm. The algorithm must contain a Feedback Mode other than ECB. This must be a binary value that is exactly the same size as the algorithm block size. • For Password Based Encryption Algorithms: This is the binary Salt value to transform the password into a key.
iterations	(Optional) The number of iterations to transform the password into a binary key. Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, you must also specify the algorithm parameter with a Password Based Encryption (PBE) algorithm. Do not specify this parameter for Block Encryption Algorithms. You must use the same value to encrypt and decrypt the data.

Usage

This function uses a symmetric key-based algorithm, in which the same key is used to encrypt and decrypt a string. The parameter values must match the values used to encode string. The security of the encrypted string depends on maintaining the secrecy of the key.

ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

Example

```
<h3>Decrypt Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>
    <cfscript>
        /* GenerateSecretKey does not generate key for the CFMX_COMPAT algorithm,
        so use the key from the form.
        */
        if (Form.myAlgorithm EQ "CFMX_COMPAT")
            theKey=Form.MyKey;
        // For all other encryption techniques, generate a secret key.
        else
            theKey=generateSecretKey(Form.myAlgorithm);
        //Encrypt the string
        encrypted=encrypt(Form.myString, theKey, Form.myAlgorithm,
            Form.myEncoding);
        //Decrypt it
        decrypted=decrypt(encrypted, theKey, Form.myAlgorithm, Form.myEncoding);
    </cfscript>

    <!--- Display the values used for encryption and decryption,
            and the results. --->
    <cfoutput>
        <b>The algorithm:</b> #Form.myAlgorithm#<br>
        <b>The key:</B> #theKey#<br>
        <br>
        <b>The string:</b> #Form.myString# <br>
        <br>
        <b>Encrypted:</b> #encrypted#<br>
        <br>
        <b>Decrypted:</b> #decrypted#<br>
    </cfoutput>
</cfif>
```

```
<!--- The input form.--->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select the encoding</b><br>
    <select size="1" name="myEncoding">
        <option selected>UU</option>
        <option>Base64</option>
        <option>Hex</option>
    </select><br>
    <br>
    <b>Select the algorithm</b><br>
    <select size="1" name="myAlgorithm">
        <option selected>CFMX_COMPAT</option>
        <option>AES</option>
        <option>DES</option>
        <option>DESEDE</option>
    </select><br>
    <br>
    <b>Input your key</b> (used for CFMX_COMPAT encryption only)<br>
    <input type = "Text" name = "myKey" value = "MyKey"><br>
    <br>
    <b>Enter string to encrypt</b><br>
    <textArea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">
        This string will be encrypted (you can replace it with more typing).
    </textArea><br>
    <input type = "Submit" value = "Encrypt my String">
</form>
```

DecryptBinary

Description

Decrypts encrypted binary data with the specified key, value, algorithm, salt, and iterations.

Returns

Unencrypted binary data.

Category

Security functions, String functions

Function syntax

```
DecryptBinary(bytes, key[, algorithm, IVorSalt, iterations])
```

See also

Duplicate, Encrypt, Decrypt

History

ColdFusion 8: Added support for encryption using the RSA BSafe Crypto-J library on Enterprise Edition.

ColdFusion MX 7.01: Added this function.

Parameters

Parameter	Description
bytes	Binary data to decrypt.
key	String. For the CFMX_COMPAT algorithm, the seed that was used to encrypt the binary data; for all other algorithms, the string generated by the `generateSecretKey()` method.
algorithm	(Optional) The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. For a list of algorithms, see the Encrypt function. The Standard Edition of ColdFusion installs a cryptography library with the following algorithms: • CFMX_COMPAT: the algorithm used in ColdFusion and prior releases. This algorithm is the least secure option (default). • AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197. • BLOWFISH: the Blowfish algorithm defined by Bruce Schneier. • DES: the Data Encryption Standard algorithm defined by NIST FIPS-46-3. • DESEDE: the "Triple DES" algorithm defined by NIST FIPS-46-3. If you install a security provider with additional cryptography algorithms, you can also specify any of its string encryption and decryption algorithms.

Parameter	Description
IVorSalt	(Optional) Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, you must also specify the `algorithm` parameter. • For Block Encryption Algorithms: This is the binary Initialization Vector value to use with the algorithm. The algorithm must contain a Feedback Mode other than ECB. This must be a binary value that is exactly the same size as the algorithm block size. • For Password Based Encryption Algorithms:- This is the binary Salt value to transform the password into a key.
iterations	(Optional) The number of iterations to transform the password into a binary key. Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, you must also specify the `algorithm` parameter with a Password Based Encryption (PBE) algorithm. Do not specify this parameter for Block Encryption Algorithms. You must use the same value must to encrypt and decrypt the data.

Usage

This function uses a symmetric key-based algorithm, in which the same key is used to encrypt and decrypt data. The parameter values must match the values used to encode the string. The security of the encrypted string depends on maintaining the secrecy of the key.

ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

Example

```
<h3>DecryptBinary Example</h3>
< !--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myfile")>
<cffile file="#Form.myfile#" action="readBinary" variable="myData">
<cfscript>
/* GenerateSecretKey does not generate key for the CFMX_COMPAT algorithm,
so use the key from the form.
*/
if (Form.myAlgorithm EQ "CFMX_COMPAT")
theKey=Form.MyKey;
// For all other encryption techniques, generate a secret key.
else
theKey=generateSecretKey(Form.myAlgorithm);
//Encrypt the string
encrypted=encryptBinary(myData, theKey, Form.myAlgorithm);
//Decrypt it
decrypted=decryptBinary(encrypted, theKey, Form.myAlgorithm);
</cfscript>
<cfset encfile="#Form.myfile#" & "_enc">
<cfset decfile="#Form.myfile#" & "_dec">
<cffile file="#encfile#" action="write" output="#encrypted#">
<cffile file="#decfile#" action="write" output="#decrypted#">
<!--- Display the values used for encryption and decryption,
and the results. --->
<cfoutput>
<b>The algorithm:</b> #Form.myAlgorithm#<br>
<b>The key:</B> #theKey#<br>
<br>
<b>The InputFile:</b> #Form.myfile# <br>
<br>
<b>Encrypted:</b> #encfile#<br>
```

```
<br>
<b>Decrypted:</b> #decfile#<br>
</cfoutput>
</cfif>
<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
<b>Select the algorithm</b><br>
    <select size="1" name="myAlgorithm">
        <option selected>CFMX_COMPAT</option>
        <option>AES</option>
        <option>DES</option>
        <option>DESEDE</option>
</select><br>
<br>
<b>Input your key</b> (used for CFMX_COMPAT encryption only)<br>
<input type = "Text" name = "myKey" value = "MyKey"><br>
<br>
<b>Enter filename to encrypt</b><br>
<input type="text" name="myfile" value="Enter the path of the file to encrypt"><br>
<input type = "Submit" value = "Encrypt file ">
</form>
```

DeleteClientVariable

Description

Deletes a client variable. (To test for the existence of a variable, use `IsDefined`.)

Returns

True, if the variable is successfully deleted; false, otherwise.

Category

Other functions

Function syntax

```
DeleteClientVariable("name")
```

See also

`GetClientVariablesList`

History

ColdFusion MX: Changed behavior: if the variable is not present, this function now returns False. (In earlier releases, it threw an error.)

Parameters

Parameter	Description
`name`	Name of a client variable to delete, surrounded by double-quotation marks

Example

```
<!--- This view-only example shows DeleteClientVariable --->
<h3>DeleteClientVariable Example</h3>

<p>This view-only example deletes a client variable called "User_ID", if it
    exists in the list of client variables returned by GetClientVariablesList.
<p>This example requires the existence of an Application.cfm file and client
    management to be in effect.
<!---
<cfset client.somevar = "">
<cfset client.user_id = "">
<p>Client variable list:<cfoutput>#GetClientVariablesList()#</cfoutput>
<cfif ListFindNoCase(GetClientVariablesList(), "User_ID") is not 0>

        <cfset temp = DeleteClientVariable("User_ID")>
        <p>Was variable "User_ID" Deleted? <cfoutput>#temp#</cfoutput>
</cfif>
<p>Amended Client variable list:<cfoutput>#GetClientVariablesList()#
</cfoutput>
--->
```

DeserializeJSON

Description

Converts a JSON (JavaScript Object Notation) string data representation into CFML data, such as a CFML structure or array.

Returns

The data value in ColdFusion format: a structure, array, query, or simple value.

Category

Conversion functions

Syntax

```
DeserializeJSON(JSONVar[, strictMapping])
```

See also

IsJSON, SerializeJSON, cfajaxproxy, "Using Ajax Data and Development Features" on page 648 in the *ColdFusion Developer's Guide*, http://www.json.org

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
JSONVar	A string that contains a valid JSON construct, or variable that represents one.
strictMapping	A Boolean value that specifies whether to convert the JSON strictly, as follows: • true: (Default) Convert the JSON string to ColdFusion data types that correspond directly to the JSON data types. • false: Determine if the JSON string contains representations of ColdFusion queries, and if so, convert them to queries.

Usage

This function is useful any time a ColdFusion page receives data as JSON strings. It is useful in ColdFusion applications that use Ajax to represent data on the client browser, and lets you consume on the server JSON format data from the client-side Ajax JavaScript. You can also use it on pages that get data from services that supply data as JavaScript function calls with JSON parameters; the example shows this use case.

The DeserializeJSON function converts each JSON data type directly into the equivalent ColdFusion data type, as follows:

- If the strictMapping parameter is true (the default), all JSON objects become CFML structures.
- If the strictMapping parameter is false, ColdFusion determines if JSON objects represent queries and, if so, converts them to ColdFusion query object. All other JSON objects become ColdFusion structures. The DeserializeJSON function recognizes a JSON structure as a query and converts it properly if the structure uses either of the two query representation formats described in the SerializeJSON reference.
- JSON Arrays, Strings, and Numbers become ColdFusion arrays, strings, and numbers.
- The JSON null value becomes the string null.

• JSON string representations of a dates and times remain strings, but ColdFusion date/time handling code can recognize them as representing dates and times.

Example

This example displays weather information from a JSON-format data feed that is generated by the example for the SerializeJSON function. Similar code might consume data that is exported as a JavaScript page. The feed is in the form of a JavaScript function call where the parameter is a JSON string that contains the feed data. The example does the following operations:

1 Uses a cfhttp tag to get the feed (in the cfhttp.fileContent variable).

2 Strips the function call wrapper from the text.

3 Uses the IsJSON function to check whether the result of the previous step is a valid JSON format string. If it is not, it displays a message and stops processing.

4 If the string is valid JSON text, uses the DeserializeJSON function to convert the string to a ColdFusion variable; in this case, a structure that contains two arrays that represent a ColdFusion query. The first array has the query column names, the second has the query data.

5 Parses the object and displays the contents of its arrays.

To run this example, put this file and the example for the SerializeJSON function in an appropriate location under your ColdFusion web root, replace the URL with the correct URL for the serialization example, and run this page.

```
<!--- Get the JSON Feed --->
<cfhttp url="http://localhost:8500/My_Stuff/Ajax/Books/CreateJSON_NEW.cfm">

<!--- JSON data is sometimes distributed as a JavaScript function.
     The following REReplace functions strip the function wrapper. --->
<cfset theData=REReplace(cfhttp.FileContent,
        "^\s*[[:word:]]*\s*\(\s*","")>
<cfset theData=REReplace(theData, "\s*\)\s*$", "")>

<!--- Test to make sure you have JSON data. --->
<cfif !IsJSON(theData)>
    <h3>The URL you requested does not provide valid JSON</h3>
    <cfdump var="#theData#">

<!--- If the data is in JSON format, deserialize it. --->
<cfelse>
    <cfset cfData=DeserializeJSON(theData)>
    <!--- Parse the resulting array or structure and display the data.
             In this case, the data represents a ColdFusion query that has been
             serialized by the SerializeJSON function into a JSON structure with
             two arrays: an array column names, and an array of arrays,
             where the outer array rows correspond to the query rows, and the
             inner array entries correspond to the column fields in the row. --->
    <!--- First, find the positions of the columns in the data array. --->
    <cfset colList=ArrayToList(cfData.COLUMNS)>
    <cfset cityIdx=ListFind(colList, "City")>
    <cfset tempIdx=ListFind(colList, "Temp")>
    <cfset fcstIdx=ListFind(colList, "Forecasts")>
    <!--- Now iterate through the DATA array and display the data. --->
    <cfoutput>
        <cfloop index="i" from="1" to="#ArrayLen(cfData.DATA)#">
            <h3>#cfData.DATA[i][cityIdx]#</h3>
            Current Temperature: #cfData.DATA[i][tempIdx]#<br><br>
            <b>Forecasts</b><br><br>
            <cfloop index="j" from="1" to="#ArrayLen(cfData.DATA[i][fcstIdx])#">
```

```
                <b>Day #j#</b><br>
                Outlook: #cfData.DATA[i][fcstIdx][j].WEATHER#<br>
                High: #cfData.DATA[i][fcstIdx][j].HIGH#<br>
                Low: #cfData.DATA[i][fcstIdx][j].LOW#<br><br>
            </cfloop>
        </cfloop>
    </cfoutput>
</cfif>
```

DirectoryExists

Description

Determines whether a directory exists.

Returns

Yes, if the specified directory exists; No, otherwise.

Category

System functions

Function syntax

`DirectoryExists(absolute_path)`

See also

`FileExists`

Parameters

Parameter	Description
`absolute_path`	An absolute path

Example

```
<h3>DirectoryExists Example</h3>
<h3>Enter a directory to check for existence.</h2>
<form action = "directoryexists.cfm" method="post">
    <input type = "text" name = "yourDirectory">
    <br>
    <input type = "submit" name = "submit">
</form>

<cfif IsDefined("FORM.yourDirectory")>
    <cfif FORM.yourDirectory is not "">
    <cfset yourDirectory = FORM.yourDirectory>
        <cfif DirectoryExists(yourDirectory)>
            <cfoutput>
            <p>Your directory exists. Directory name: #yourDirectory#
            </cfoutput>
        <cfelse>
            <p>Your directory does not exist.</p>
        </cfif>
    </cfif>
</cfif>
```

DollarFormat

Description

Formats a string in U.S. format. (For other currencies, use LSCurrencyFormat or LSEuroCurrencyFormat .

Returns

A number as a string, formatted with two decimal places, thousands separator, and dollar sign. If *number* is negative, the return value is enclosed in parentheses. If *number* is an empty string, returns zero.

Category

Display and formatting functions

Function syntax

DollarFormat(*number*)

See also

DecimalFormat, NumberFormat

Parameters

Parameter	Description
number	Number to format

Example

```
<!--- This example shows the use of DollarFormat --->
...
<h3>DollarFormat Example</h3>
<cfloop from = 8 to = 50 index = counter>
    <cfset full = counter>
    <cfset quarter = counter + (1/4)>
    <cfset half = counter + (1/2)>
    <cfset threefourth = counter + (3/4)>
    <cfoutput>
    <pre>
    bill#DollarFormat(full)##DollarFormat(quarter)#
     #DollarFormat(half)# #DollarFormat(threefourth)#
    18% tip#DollarFormat(full * (18/100))#
     #DollarFormat(quarter * (18/100))#
     #DollarFormat(half * (18/100))#
     #DollarFormat(threefourth * (18/100))#
    </pre>
    </cfoutput>
</cfloop>
...
```

DotNetToCFType

Description

Explicitly converts a value returned by a .NET method to the corresponding ColdFusion data type.

Returns

A ColdFusion data value.

Category

Structure functions, System functions

Function syntax

`DotNetToCFType(variable_name)`

See also

`CreateObject: .NET object`, `cfobject: .NET object`, "Converting between .NET and ColdFusion data types" on page 960 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
variable_name	Name of the .NET variable to convert

Usage

For detailed information on when and why you use this function, see "Working with complex .NET data types" on page 964 in the *ColdFusion Developer's Guide*.

Example

The following example creates a .NET System.Data.DataTable object and converts it to a ColdFusion query.

```
<!---Create a SQL Command Object--->
<cfobject action="create" name="sqlCommandObject"
    class="System.Data.SqlClient.SqlCommand" type=".Net"
    assembly="#assemblyList#">

<cfset sqlCommandObject.init("SELECT [ID], [FriendlyName] FROM [Batch]",
        sqlConnectionObject)>

<cfset sqlDataReaderObject = sqlCommandObject.ExecuteReader()>

<cfset dataTable = createObject(".net", "System.Data.DataTable",
        assemblyList)>
<!--- populate the datatable --->
<cfset dataTable.load(sqlDataReaderObject)>

<!--- convert to cfquery --->
<cfset myquery=DotNetToCFType(dataTable)>
```

Duplicate

Description
Returns a clone, also known as a deep copy, of a variable. There is no reference to the original variable.

Returns
A clone of a variable.

Category
Structure functions, System functions

Function syntax
```
Duplicate(variable_name)
```

See also
`StructCopy`, other Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Changed behavior: this function can duplicate CFCs.

ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
`variable_name`	Name of a variable to duplicate

Usage
Use this function to duplicate complex structures, such as nested structures and queries.

When you duplicate a CFC instance, the entire CFC contents is copied, including the values of the variables in the this scope at the time you call the `Duplicate` function. Thereafter, the two CFC instances are independent, and changes to one copy, for example by calling one of its functions, have no effect on the other copy.

Note: *With this function, you cannot duplicate a COM, CORBA, or JAVA object returned from the `cfobject` tag or the `CreateObject` function. If an array element or structure field is a COM, CORBA, or JAVA object, you cannot duplicate the array or structure.*

Example
```
<h3>Duplicate Example</h3>
<cfset s1 = StructNew()>
<cfset s1.nested = StructNew()>
<cfset s1.nested.item = "original">
<cfset copy = StructCopy(s1)>
<cfset clone = Duplicate(s1)>
<!--- modify the original --->
<cfset s1.nested.item = "modified">
<cfoutput>
<p>The copy contains the modified value: #copy.nested.item#</p>
<p>The duplicate contains the original value: #clone.nested.item#</p>
</cfoutput>
```

Encrypt

Description

Encrypts a string using a specific algorithm and encoding method.

Returns

String; can be much longer than the original string.

Category

Security functions, String functions

Function syntax

`Encrypt(string, key [, algorithm, encoding, IVorSalt, iterations])`

See also

`Decrypt`, `EncryptBinary`, `DecryptBinary`

History

ColdFusion 8: Added support for encryption using the RSA BSafe Crypto-J library on Enterprise Edition.

ColdFusion MX 7.01 : Added the *`IVorSalt`* and *`iterations`* parameters.

ColdFusion MX 7: Added the *`algorithm`* and *`encoding`* parameters.

Parameters

Parameter	Description
`string`	String to encrypt.
`key`	String. Key or seed used to encrypt the string. • For the CFMX_COMPAT algorithm, any combination of any number of characters; used as a seed used to generate a 32-bit encryption key. • For all other algorithms, a key in the format used by the algorithm. For these algorithms, use the `GenerateSecretKey` function to generate the key.
`algorithm`	(Optional) The algorithm to use to encrypt the string. The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. It includes the following algorithms: • AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197. • DES: the Data Encryption Standard algorithm defined by NIST FIPS-46-3. • DES-EDE: the "Triple DES" algorithm defined by NIST FIPS-46-3. • DESX: The extended Data Encryption Standard symmetric encryption algorithm.

Parameter	Description
	• RC2: The RC2 block symmetric encryption algorithm defined by RFC 2268. • RC4: The RC4 symmetric encryption algorithm. • RC5: The RC5 encryption algorithm. • PBE: Password-based encryption algorithm defined in PKCS #5. • MD2: The MD2 hash algorithm defined by RFC 1319. • MD5: The defined by RFC 1321. • RIPEMD160: RACE Integrity Primitives Evaluation Message Digest 160-bit message digest algorithm and cryptographic hash function. • SHA-1: The 160-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • SHA-224: The 224-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • SHA-256: The 256-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • SHA-384: The 384-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • SHA-512: The 512-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • HMAC-MD5: The hash message authentication code calculated using the MD5 hash algorithm. • HMAC-RIPEMD160: The hash message authentication code calculated using the RACE Integrity Primitives Evaluation Message Digest 160-bit message digest algorithm and cryptographic hash function. • HMAC-SHA1: The hash message authentication code calculated using the 160-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • HMAC-SHA224: The hash message authentication code calculated using the 224-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • HMAC-SHA256: The hash message authentication code calculated using the 256-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • HMAC-SHA384: The hash message authentication code calculated using the 384-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • HMAC-SHA512: The hash message authentication code calculated using the 512-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198. • RSA: The RSA public key algorithm defined by PKCS#1 v1.5 and v2.0. • DSA: The digital signature algorithm defined by FIPS 186-2. • Diffie-Hellman: The Diffie-Hellman key exchange algorithmdefined by PKCS #3. In addition to these algorithms, you can use the algorithms provided in the Standard Edition of ColdFusion. The Standard Edition of ColdFusion installs a cryptography library with the following algorithms: • CFMX_COMPAT: the algorithm used in ColdFusion MX and prior releases. This algorithm is the least secure option (default). • AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197. • BLOWFISH: the Blowfish algorithm defined by Bruce Schneier. • DES: the Data Encryption Standard algorithm defined by NIST FIPS-46-3. • DESEDE: the "Triple DES" algorithm defined by NIST FIPS-46-3. If you install a security provider with additional cryptography algorithms, you can also specify any of its string encryption and decryption algorithms.

Parameter	Description
encoding	(Optional; if you specify this parameter, you must also specify the *algorithm* parameter). The binary encoding in which to represent the data as a string. • Base64: the Base64 algorithm, as specified by IETF RFC 2045. • Hex: the characters !-F0-9 represent the hexadecimal byte values. • UU: the UUEncode algorithm (default).
IVorSalt	(Optional) Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, you must also specify the *algorithm* parameter. • For Block Encryption algorithms: This is the binary Initialization Vector value to use with the algorithm. The algorithm must contain a Feedback Mode other than ECB. This must be a binary value that is exactly the same size as the algorithm block size. You must use the same value in the Decrypt function to successfully decrypt the data. • For Password Based Encryption algorithms: This is the binary Salt value to transform the password into a key. The same value must be used to decrypt the data.
iterations	(Optional) The number of iterations to transform the password into a binary key. Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, you must also specify the algorithm parameter with a Password Based Encryption (PBE) algorithm. Do not specify this parameter for Block Encryption algorithms. You must use the same value to encrypt and decrypt the data.

Usage

This function uses a symmetric key-based algorithm, in which the same key is used to encrypt and decrypt a string. The security of the encrypted string depends on maintaining the secrecy of the key.

The following are the FIPS-140 approved algorithms included in the RSA BSafe Crypto-J library:

- AES – ECB, CBC, CFB (128), OFB (128) – [128, 192, 256 bit key sizes]
- AES – CTR
- Diffie-Hellman Key Agreement
- DSA
- FIPS 186-2 General Purpose [(x-Change Notice); (SHA-1)]
- FIPS 186-2 [(x-Change Notice); (SHA-1)]
- HMAC-SHAx (where x is 1, 224, 256, 384, or 512)
- RSASSA-PSS (sign, verify) (SHA-1)
- RSASSA-PSS (sign, verify) (SHA-224, SHA-256, SHA-384, SHA-512)
- RSA PKCS#1 v1.5 (sign, verify) (SHA-1,SHA-224,SHA-256,SHA-384,SHA-512)
- Secure Hash Standard (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)
- Triple DES - ECB, CBC, CFB (64 bit), and OFB (64 bit)
- RSA X9.31 (keygen, sign, verify)

All algorithms included in the RSA BSafe Crypto-J library are available for use in the Enterprise Edition. In certain cases, you may want to restrict the available algorithms to FIPS-140 approved algorithms only. To do so, you specify the following in the JVM arguments on the Java and JVM page of the ColdFusion Administrator:

```
-Dcoldfusion.enablefipscrypto=true
```

FIPS-140 approved cryptography is not available if you are running ColdFusion on Websphere of JBoss.

To use the IBM/Lotus Sametime Instant Messaging Gateway in the Enterprise edition, you must disable the FIPS-140-only cryptography setting by specifying the following in the JVM arguments on the Java and JVM page of the ColdFusion Administrator:

```
-Dcoldfusion.disablejsafe=true
```

In Standard Edition, for all algorithms except the default algorithm, ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

The default algorithm, which is the same one used in ColdFusion 5 and ColdFusion MX, uses an XOR-based algorithm that uses a pseudo-random 32-bit key, based on a seed passed by the user as a function parameter. This algorithm is less secure than the other available algorithms.

Example

The following example encrypts and decrypts a text string. It lets you specify the encryption algorithm and encoding technique. It also has a field for a key seed to use with the CFMX_COMPAT algorithm. For all other algorithms, it generates a secret key.

```
<h3>Encrypt Example</h3>
<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>
    <cfscript>
        /* GenerateSecretKey does not generate key for the CFMX_COMPAT algorithm,
        so use the key from the form.
        */
        if (Form.myAlgorithm EQ "CFMX_COMPAT")
            theKey=Form.MyKey;
        // For all other encryption techniques, generate a secret key.
        else
            theKey=generateSecretKey(Form.myAlgorithm);
        //Encrypt the string
        encrypted=encrypt(Form.myString, theKey, Form.myAlgorithm,
            Form.myEncoding);
        //Decrypt it
        decrypted=decrypt(encrypted, theKey, Form.myAlgorithm, Form.myEncoding);
    </cfscript>

    <!--- Display the values used for encryption and decryption,
            and the results. --->
    <cfoutput>
        <b>The algorithm:</b> #Form.myAlgorithm#<br>
        <b>The key:</B> #theKey#<br>
        <br>
        <b>The string:</b> #Form.myString# <br>
        <br>
        <b>Encrypted:</b> #encrypted#<br>
        <br>
        <b>Decrypted:</b> #decrypted#<br>
    </cfoutput>
</cfif>

<!--- The input form.--->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select the encoding</b><br>
    <select size="1" name="myEncoding">
        <option selected>UU</option>
        <option>Base64</option>
```

```
        <option>Hex</option>
    </select><br>
    <br>
    <b>Select the algorithm</b><br>
    <select size="1" name="myAlgorithm">
        <option selected>CFMX_COMPAT</option>
        <option>AES</option>
        <option>DES</option>
        <option>DESEDE</option>
    </select><br>
    <br>
    <b>Input your key</b> (used for CFMX_COMPAT encryption only)<br>
    <input type = "Text" name = "myKey" value = "MyKey"><br>
    <br>
    <b>Enter string to encrypt</b><br>
    <textArea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">This string will be
encrypted (you can replace it with more typing).
    </textArea><br>
    <input type = "Submit" value = "Encrypt my String">
</form>
```

EncryptBinary

Description

Encrypts binary data using a specific algorithm and encoding method.

Returns

Binary data.

Category

Security functions, String functions

Function syntax

```
EncryptBinary(bytes, key [, algorithm, IVorSalt, iterations])
```

See also

Decrypt, DecryptBinary, Encrypt

History

ColdFusion 8: Added support for encryption using the RSA BSafe Crypto-J library on Enterprise Edition.

ColdFusion MX 7.01 : Added this function.

Parameters

Parameter	Description
bytes	Bindary data to encrypt.
key	String. Key or seed used to encrypt the string. • For the CFMX_COMPAT algorithm, any combination of any number of characters; used as a seed used to generate a 32-bit encryption key. • For all other algorithms, a key in the format used by the algorithm. For these algorithms, use the GenerateSecretKey function to generate the key.
algorithm	(Optional) The algorithm to use to decrypt the string. The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. For a list of algorithms, see the Encrypt function. The Standard Edition of ColdFusion installs a cryptography library with the following algorithms: • CFMX_COMPAT: the algorithm used in ColdFusion and prior releases. This algorithm is the least secure option (default). • AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197. • BLOWFISH: the Blowfish algorithm defined by Bruce Schneier. • DES: the Data Encryption Standard algorithm defined by NIST FIPS-46-3. • DESEDE: the "Triple DES" algorithm defined by NIST FIPS-46-3. If you install a security provider with additional cryptography algorithms, you can also specify any of its string encryption and decryption algorithms.

Parameter	Description
IVorSalt	(Optional) Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, you must also specify the *algorithm* parameter. • For Block Encryption algorithms: This is the binary Initialization Vector value to use with the algorithm. The algorithm must contain a Feedback Mode other than ECB. This must be a binary value that is exactly the same size as the algorithm block size. You must use the same value in the Decrypt function to successfully decrypt the data. • For Password Based Encryption algorithms: This is the binary Salt value to transform the password into a key. The same value must be used to decrypt the data.
iterations	(Optional) The number of iterations to transform the password into a binary key. Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, you must also specify the *algorithm* parameter with a Password Based Encryption (PBE) algorithm. Do not specify this parameter for Block Encryption algorithms. You must use the same value to encrypt and decrypt the data.

Usage

This function uses a symmetric key-based algorithm, in which the same key is used to encrypt and decrypt binary data. The security of the encrypted data depends on maintaining the secrecy of the key.

For all algorithms except the default algorithm, ColdFusion MX 7 uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

The default algorithm, which is the same as was used in ColdFusion 5 and ColdFusion MX, uses an XOR-based algorithm that uses a pseudo-random 32-bit key, based on a seed passed by the user as a function parameter. This algorithm is less secure than the other available algorithms.

Example

The following example encrypts and decrypts binary data. It encrypts the binary data contained in a file and then decrypts the encrypted file. It lets you specify the encryption algorithm and encoding technique. It also has a field for a key seed to use with the CFMX_COMPAT algorithm. For all other algorithms, it generates a secret key.

```
<h3>EncryptBinary Example</h3>
<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myfile")>

<cffile file="#Form.myfile#" action="readBinary" variable="myData">
<cfscript>
/* GenerateSecretKey does not generate key for the CFMX_COMPAT algorithm,
so use the key from the form.
*/
if (Form.myAlgorithm EQ "CFMX_COMPAT")
theKey=Form.MyKey;
// For all other encryption techniques, generate a secret key.
else
theKey=generateSecretKey(Form.myAlgorithm);
//Encrypt the string
encrypted=encryptBinary(myData, theKey, Form.myAlgorithm);
//Decrypt it
decrypted=decryptBinary(encrypted, theKey, Form.myAlgorithm);
</cfscript>
<cfset encfile="#Form.myfile#" & "_enc">
<cfset decfile="#Form.myfile#" & "_dec">
<cffile file="#encfile#" action="write" output="#encrypted#">
<cffile file="#decfile#" action="write" output="#decrypted#">

<!--- Display the values used for encryption and decryption,
```

```
and the results. --->
<cfoutput>
<b>The algorithm:</b> #Form.myAlgorithm#<br>
<b>The key:</B> #theKey#<br>
<br>
<b>The InputFile:</b> #Form.myfile# <br>
<br>
<b>Encrypted:</b> #encfile#<br>
<br>
<b>Decrypted:</b> #decfile#<br>
</cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
<b>Select the algorithm</b><br>
<select size="1" name="myAlgorithm">
<option selected>CFMX_COMPAT</option>
<option>AES</option>
<option>DES</option>
<option>DESEDE</option>
</select><br>
<br>
<b>Input your key</b> (used for CFMX_COMPAT encryption only)<br>
<input type = "Text" name = "myKey" value = "MyKey"><br>
<br>
<b>Enter filename to encrypt</b><br>
<input type="text" name="myfile" value="Enter the path of the file to encrypt"><br>
<input type = "Submit" value = "Encrypt file ">
</form>
```

Evaluate

Description

Evaluates one or more string expressions, dynamically, from left to right. (The results of an evaluation on the left can have meaning in an expression to the right.) Returns the result of evaluating the rightmost expression.

Returns

An object; the result of the evaluation(s).

Category

Dynamic evaluation functions

Function syntax

```
Evaluate(string_expression1 [, string_expression2 , … ])
```

See also

DE, IIf, PrecisionEvaluate, "Using Expressions and Number Signs" on page 50 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
string_expression1, string_expression2...	Expressions to evaluate

Usage

String expressions can be complex. If a string expression contains a single- or double-quotation mark, the mark must be escaped.

This function is useful for forming one variable from multiple variables. For example, to reference a column of the query qNames with a variable, var, using an index value to traverse rows, you could use the following code:

```
<cfset var=Evaluate("qNames.#colname#[#index#]")>
```

Example

```
<!--- This example shows the use of PrecisionEvaluate and DE functions.--->
<h3>Evaluate Example</h3>
<cfif IsDefined("FORM.myExpression")>
    <cftry>
        <!--- Evaluate the expression --->
        <cfset theExpression = Evaluate(Form.myExpression)>
        <cfoutput>
            <!--- The DE function prevents the Evaluate function from evaluating
                    the expression. --->
            The value of the expression #Evaluate(DE(FORM.MyExpression))#
            is #theExpression#.<br>
            <!--- The following line does not use the DE function. --->
            The value of the expression #FORM.MyExpression#
            is #theExpression#.<br>
        </cfoutput>

        <cfcatch type="application">
            <cfoutput>Could not evaluate the expression #Form.myExpression#.<br>
                    Make sure you enter a valid ColdFusion Expression.
            </cfoutput>
```

```
        </cfcatch>
    </cftry>
</cfif>

<cfform preservedata="yes">
    <h3>Enter a ColdFusion expression for evaluation</h3>
    <cfinput type="text" name="myExpression" size="60"><br />
    <br />
    <cfinput type="submit" name="submit">
</cfform>
```

Exp

Description

Calculates the exponent whose base is e that represents *number*. The constant e equals 2.71828182845904, the base of the natural logarithm. This function is the inverse of Log, the natural logarithm of *number*.

Returns

The constant e, raised to the power of *number*.

Category

Mathematical functions

Function syntax

Exp(*number*)

See also

Log, Log10

Parameters

Parameter	Description
number	Exponent to apply to the base e

Usage

To calculate powers of other bases, use the exponentiation operator (^).

Example

```
<h3>Exp Example</h3>
<cfif IsDefined("FORM.Submit")>
    <cfoutput>
    <p>Your number, #FORM.number#
    <br>#FORM.number# raised to the E power: #exp(FORM.number)#
    <cfif FORM.number LTE 0>
        <br>You must enter a positive real number to see its natural logarithm
        <cfelse><br>
        The natural logarithm of #FORM.number#: #log(FORM.number)#
    </cfif>
    <cfif FORM.number LTE 0><br>
        You must enter a positive real number to see its logarithm to base 10
        <cfelse><br>
            The logarithm of #FORM.number# to base 10: #log10(FORM.number)#
</cfif>
</cfoutput>
</cfif>
<cfform action = "exp.cfm">
Enter a number to see its value raised to the E power, its natural logarithm,
    and the logarithm of number to base 10.
<cfinput type = "Text" name = "number" message = "You must enter a number"
    validate = "float" required = "No">
<input type = "Submit" name = "Submit">
</cfform>
```

ExpandPath

Description

Creates an absolute, platform-appropriate path that is equivalent to the value of *relative_path*, appended to the base path. This function (despite its name) can accept an absolute or relative path in the *relative_path* parameter

The base path is the currently executing page's directory path. It is stored in pageContext.getServletContext().

Returns

A string. If the relative path contains a trailing forward slash or backward slash, the return value contains the same trailing character.

Category

System functions

Function syntax

ExpandPath(*relative_path*)

See also

FileExists, GetCurrentTemplatePath, GetFileFromPath

History

ColdFusion MX: Changed behavior for the *relative_path* parameter: this function can now accept an absolute or relative path in the *relative_path* parameter. To resolve a path, this function uses virtual mappings that are defined in the ColdFusion Administrator. This function does not reliably use virtual mappings that are defined in IIS, Apache, or other web servers.

Parameters

Parameter	Description
relative_path	Relative or absolute directory reference or filename, within the current directory, (.\ and ..\) to convert to an absolute path. Can include forward or backward slashes.

Usage

If the parameter or the returned path is invalid, the function throws an error.

These examples show the valid constructions of relative_path:

- ExpandPath("*.*")
- ExpandPath("/")
- ExpandPath("\")
- ExpandPath("/mycfpage.cfm")
- ExpandPath("mycfpage.cfm")
- ExpandPath("myDir/mycfpage.cfm")
- ExpandPath("/myDir/mycfpage.cfm")
- ExpandPath("../../mycfpage.cfm")

Example

```
<h3>ExpandPath Example - View Only</h3>
<!---
<cfset thisPath=ExpandPath("*.*")>
<cfset thisDirectory=GetDirectoryFromPath(thisPath)>
```

```
<cfoutput>
The current directory is: #GetDirectoryFromPath(thisPath)#

<cfif IsDefined("form.yourFile")>
<cfif form.yourFile is not "">
<cfset yourFile = form.yourFile>
    <cfif FileExists(ExpandPath(yourfile))>
    <p>Your file exists in this directory. You entered
    the correct filename, #GetFileFromPath("#thisPath#/#yourfile#")#
    <CFELSE>
    <p>Your file was not found in this directory:
    <br>Here is a list of the other files in this directory:
    <!--- use CFDIRECTORY to give the contents of the
    snippets directory, order by name and size --->
    <CFDIRECTORY DIRECTORY="#thisDirectory#"
    NAME="myDirectory"
    SORT="name ASC, size DESC">
    <!--- Output the contents of the CFDIRECTORY as a CFTABLE --->
    <CFTABLE QUERY="myDirectory">
    <CFCOL HEADER="NAME:"
            TEXT="#Name#">
    <CFCOL HEADER="SIZE:"
            TEXT="#Size#">
    </CFTABLE>
    </cfif>
</cfif>
<cfelse>
<h3>Please enter a filename</h3>
</CFIF>
</cfoutput>

<FORM action="expandpath.cfm" METHOD="post">
<h3>Enter the name of a file in this directory <I>
    <FONT SIZE="-1">(try expandpath.cfm)</FONT></I></h3>
<INPUT TYPE="Text" NAME="yourFile">
<INPUT TYPE="Submit" NAME="">
</form>
 --->
```

FileClose

Description

Closes a file that is open. When you use the FileOpen function, ColdFusion returns a handle to a file object.When you close the file, the handle is still available; however, it lists the file as closed.

Category

System functions

Function syntax

FileClose(*fileObj*)

See also

FileCopy, FileIsEOF, FileOpen, FileRead, FileReadLine, FileWrite

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
fileobj	The file to close.

Usage

You should always close a file after opening it. When you use the FileOpen function to open a file, the file stream from the disk is opened and contents are read from or written to it. The FileClose function closes the stream. If you do not close a file, the stream remains open; in that case, the operating system can lock the file, which results in the file not being usable until the server is restarted.

Example

The following example checks to see if a file is still open and closes it.

```
<cfscript>
myfile = FileOpen("c:\ColdFusionScorpio\wwwroot\test1.txt", "read");
while(NOT FileIsEOF(myfile))
{
x = FileReadLine(myfile);
WriteOutput("#x# <br>");
}
</cfscript>
<!--- Additional code goes here. --->
<cfif #myfile.status# IS "open">
    <cfoutput>The file #myfile.filepath# is #myfile.status#</cfoutput><br>
    <cfscript>
    FileClose(myfile);
    </cfscript>
</cfif>
```

FileCopy

Description

Copies the specified source file to the specified destination file.

Category

System functions

Function syntax

`FileCopy(source, destination)`

See also

`FileClose`, `FileIsEOF`, `FileOpen`, `FileRead`, `FileReadLine`, `FileWrite`, `cffile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`source`	Pathname of the file to copy. If not an absolute path (starting a with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`destination`	Pathname of a directory or file on the web server where the file is copied. If you specify a filename without a directory path, ColdFusion copies it relative to the source directory.

Example

The following example copies the test1.txt file from the c:\testingdir\ directory to the c:\productiondir\ directory in Windows and names the new copy of the file test2.txt:

```
<h3>FileCopy Example</h3>
<cfset sourcefile="c:\testingdir\test1.txt">
<cfset destinationfile="c:\productiondir\test2.txt">

<cfif FileExists(#sourcefile#)>
    <cfif FileExists(#destinationfile#)>
        <cfoutput>A copy of #destinationfile# already exists.</cfoutput>
    <cfelse>
        <cfscript>
        FileCopy(#sourcefile#, #destinationfile#);
        </cfscript>
        <cfoutput>Copied: #sourcefile# <br>
            To: #destinationfile#</cfoutput><br>
    </cfif>
<cfelse>
    <cfoutput>The source file does not exist.</cfoutput><br>
</cfif>
```

FileDelete

Description

Deletes the specified file on the server.

Category

System functions

Function syntax

`FileDelete(filepath)`

See also

`FileClose`, `FileIsEOF`, `FileOpen`, `FileRead`, `FileReadLine`, `FileWrite`, `cffile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`filepath`	Pathname of the file to delete. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.

Example

The following example deletes the file c:\productiondir\test1.txt before moving c:\testdir\test1.txt:

```
<h3>FileDelete Example</h3>

<cfset sourcefile="c:\testdir\test1.txt">
<cfset destinationfile="c:\productiondir\test1.txt">

<cfif FileExists(#sourcefile#)>
    <cfif FileExists(#destinationfile#)>
        <cfoutput>The destination file already exists.<br>
            Deleting previous copy of #destinationfile#.<br>
            Moving: #sourcefile# <br>
            To: <br> #destinationfile#.</cfoutput><br>
        <cfscript>
            FileDelete(#destinationfile#);
            FileMove(#sourcefile#, #destinationfile#);
        </cfscript>
    <cfelse>
        <cfscript>
            FileMove(#sourcefile#, #destinationfile#);
        </cfscript>
        <cfoutput>Moved: #sourcefile# <br>
            To: <br> #destinationfile#.</cfoutput><br>
    </cfif>
<cfelse>
    <cfoutput>The source file does not exist.</cfoutput><br>
</cfif>
```

FileExists

Description
Determines whether a file exists.

Returns
Yes, if the file specified in the parameter exists; No, otherwise.

Category
System functions, Decision functions

Function syntax
`FileExists(absolute_path)`

See also
`DirectoryExists`, `ExpandPath`, `GetTemplatePath`

Parameters

Parameter	Description
`absolute_path`	An absolute path

Usage
To access a file on a remote system, the account (for Windows) or user (for UNIX and Linux) that is running ColdFusion must have permission to access the file, directory, and remote system. For example, if you run ColdFusion in the Server Configuration as a Windows service, by default it runs under the local system account, which does not have sufficient privileges to access remote systems. You can change this, however, on the Log On page of the Services Properties dialog box.

Example
```
<h3>FileExists Example</h3>

<cfset thisPath = ExpandPath("*.*")>
<cfset thisDirectory = GetDirectoryFromPath(thisPath)>
<cfoutput>
The current directory is: #GetDirectoryFromPath(thisPath)#
<cfif IsDefined("FORM.yourFile")>
<cfif FORM.yourFile is not "">
<cfset yourFile = FORM.yourFile>
    <cfif FileExists(ExpandPath(yourfile))>
    <p>Your file exists in this directory. You entered
    the correct filename, #GetFileFromPath("#thisPath#/#yourfile#")#</p>
    <cfelse>
```

FileIsEOF

Description

Determines whether ColdFusion has reached the end of the file while reading it.

Returns

Yes, if the end of the file has been reached; No, otherwise.

Category

System functions, Decision functions

Function syntax

`FileIsEOF(fileObj)`

See also

`FileClose`, `FileOpen`, `FileRead`, `FileReadLine`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`fileObj`	The file object.

Example

The following example reads a file until it reaches the end of the file:

```
<h3>FileIsEOF Example</h3>

<cfscript>
myfile = FileOpen("c:\ColdFusionScorpio\wwwroot\test1.txt", "read");
while(NOT FileIsEOF(myfile))
{
x = FileReadLine(myfile);
WriteOutput("#x# <br>");
}
FileClose(myfile);
</cfscript>
```

FileMove

Description

Moves a file from one location to another on the server.

Category

System functions

History

ColdFusion 8: Added this function.

Function syntax

`FileMove(source, destination)`

See also

`FileClose`, `FileCopy`, `FileOpen`, `FileRead`, `FileReadLine`, `FileWrite`, `cffile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`source`	Pathname of the file to move. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
`destination`	Pathname of the destination directory or file. If not an absolute path, it is relative to the source directory.

Example

The following example moves the test1.txt file from the c:\testingdir\ directory to the c:\productiondir\ directory in Windows and renames the file test2.txt:

```
<h3>FileMove Example</h3>
<cfset sourcefile="c:\testingdir\test1.txt">
<cfset destinationfile="c:\productiondir\test2.txt">

<cfif FileExists(#sourcefile#)>
    <cfif FileExists(#destinationfile#)>
        <cfoutput>The destination file already exists.</cfoutput>
    <cfelse>
        <cfscript>
        FileMove(#sourcefile#, #destinationfile#);
        </cfscript>
        <cfoutput>Moved: #sourcefile# <br>
            To: <br> #destinationfile#.</cfoutput><br>
    </cfif>
<cfelse>
    <cfoutput>The source file does not exist.</cfoutput><br>
</cfif>
```

FileOpen

Description

Opens a file to read, write, or append. Use this function with the `FileRead` function to read large files.

Returns

The filename, the absolute filepath, when the file was most recently modified, the mode for which you opened the file, the file size in bytes, and whether the file is open or closed.

Category

System functions

Function syntax

`FileOpen(filepath, [mode, charset])`

See also

`FileClose`, `FileCopy`, `FileReadBinary`, `FileRead`, `FileReadLine`, `FileWrite`, `cffile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`filepath`	An absolute path of a file on the server.
`mode`	Action to perform on the file, including the following: • `read` • `readBinary` • `write` • `append` If you do not specify the mode, ColdFusion opens the file in read mode.
`charset`	The character set of the file.

Usage

The file object is a handle to a file. The handle includes the following information:

- filename Name of the file you opened
- filepath Absolute path and filename
- lastmodified The time when the file was most recently modified
- mode The action for which the file was opened
- size The file size in bytes
- status Whether the file object is open or closed

You refer to these as elements of a structure, for example fileobj.filename. The following opens a file, and then displays the absolute path and filename of that file:

```
<cfscript>
myfile = FileOpen("c:\temp\test1.txt", "read");
</cfscript>
```

```
myfile refers to:
<cfdump var="#myfile.filepath#">
```

You should always close a file after opening it. When you use the `FileOpen` function to open a file, the file stream from the disk is opened and contents are read from or written to it. The `FileClose` function closes the stream. If you do not close a file, the stream remains open; in that case, the operating system can lock the file, which results in the file not being usable until the server is restarted.

Example

The following example opens a file, reads and outputs each line of the file, then closes the file.

```
<h3>FileOpen Example</h3>

<cfscript>
myfile = FileOpen("c:\temp\test1.txt", "read");
while(NOT FileIsEOF(myfile))
{
x = FileReadLine(myfile);
WriteOutput("#x# <br>"); }
FileClose(myfile);
</cfscript>
```

FileRead

Description

Reads a text file or a file object created with the FileOpen function. You use this function either as an alternative to the cffile tag with the action="read" attribute or to read very large file by reading the file object created by the FileOpen function to improve performance, because FileRead does not read the entire file into memory.

Returns

If you specify a filepath, the full text content of the file.

If you specify a file object, the character or byte buffer of the specified size.

If the file was opened in read mode, FileRead returns the character data (a string), otherwise it returns binary data.

Category

System functions

Function syntax

```
FileRead(filepath [, charset])

OR

FileRead(fileobj [, buffersize])
```

See also

FileClose, FileIsEOF, FileReadBinary, FileReadLine, FileWrite

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
filepath	An absolute path to a text file on the server.
charset	The character encoding in which the file contents is encoded. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 If the file starts with a byte order mark and you set this attribute to a conflicting character encoding, ColdFusion generates an error.
fileobj	The file object from which to read.
buffersize	The number of characters to read.

Usage

You can read a text file or a file object with the `FileRead` function. When you specify an absolute path of a text file, ColdFusion reads the entire contents of the file. When you specify a file object, which you created using the `FileOpen` function, ColdFusion reads the number of characters specified in `buffersize`.

Example

```
<h3>FileRead Example - Reading a file</h3>

<!--- This reads and outputs the entire file contents. --->
<cfscript>
myfile = FileRead("c:\temp\myfile.txt");
    WriteOutput("#myfile#");
</cfscript>

<!--- This reads and outputs the first 100 characters --->
<!--- from the same file. --->
<cfscript>
myfile = FileOpen("c:\temp\test1.txt", "read");
x = FileRead(myfile, 100);
WriteOutput("#x#");
FileClose(myfile);
</cfscript>
```

FileReadBinary

Description

Reads a binary file (such as an executable or image file) on the server, into a binary object parameter that you can use in the page. To send it through a web protocol (such as HTTP or SMTP) or store it in a database, first convert it to Base64 by using the `ToBase64` function.

Note: *This action reads the file into a variable in the local Variables scope. It is not intended for use with large files, such as logs, because they can bring down the server.*

Returns

The entire contents of a binary file.

Category

System functions

Function syntax

`FileReadBinary(`*`filepath`*`)`

See also

`FileClose`, `FileIsEOF`, `FileRead`, `FileReadLine`, `FileWrite`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`filepath`	An absolute path to a binary file on the server

Usage

You convert the binary file to Base64 to transfer it to another site. ColdFusion 8 supports reading an image file as a binary and passing the result to a `cfimage`.

Example

The following example reads a binary file.

```
<h3>FileReadBinary Example</h3>
<cfscript>
myfile = FileReadBinary("c:\testingdir\test3.jpg");
</cfscript>
```

FileReadLine

Description
Reads a line from the file.

Returns
The line of the file.

Category
System functions

Function syntax
`FileReadLine(fileObj)`

See also
`FileClose`, `FileIsEOF`, `FileRead`, `FileWrite`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`fileObj`	The file object

Example
The following example opens a file, reads each line, outputs each line, and then closes the file.

```
<h3>FileReadLIne Example</h3>

<cfscript>
myfile = FileOpen("c:\ColdFusionScorpio\wwwroot\test1.txt", "read");
while(NOT FileisEOF(myfile))
{
x = FileReadLine(myfile); // read line
WriteOutput("#x#");
}
FileClose(myfile);
</cfscript
```

FileSetAccessMode

Description

Sets the attributes of a file on UNIX or Linux.

Category

System functions

Function syntax

`FileSetAccessMode(filepath, mode)`

See also

`FileCopy`, `FileDelete`, `FileExists`, `FileMove`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`filepath`	An absolute path to the file on the server.
`mode`	A three-digit value, in which each digit specifies the file access for individuals and groups: • The first digit represents the owner. • The second digit represents a group. • The third digit represents anyone. Each digit of this code sets permissions for the appropriate individual or group: • 4 specifies read permission. • 2 specifies write permission. • 1 specifies execute permission. You use the sums of these numbers to indicate combinations of the permissions: • 3 specifies write and execute permission. • 5 specifies read and execute permission. • 6 indicates read and write permission. • 7 indicates read, write, and execute permission. For example, 400 specifies that only the owner can read the file; 004 specifies that anyone can read the file.

Example

The following example sets the access mode of a file so that only the owner can read the file.

```
<h3>FileSetAccessMode Example</h3>

<cfscript>
    FileSetAccessMode("test1.txt", "004");
</cfscript>
```

FileSetAttribute

Description
Sets the attributes of a file in Windows.

Category
System functions

Function syntax
`FileSetAttribute(filepath, attribute)`

See also
`FileCopy`, `FileDelete`, `FileExists`, `FileMove`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`filepath`	An absolute path to a file on the server.
`attribute`	One of the following: • `readOnly` • `hidden` • `normal` Set the attribute to `normal` to make a file not read-only and not hidden.

Example
The following example sets the access mode of a file to be read-only.

```
<h3>FileSetAttribute Example</h3>

<cfscript>
    FileSetAttribute("c:\temp\test1.txt", "readOnly");
</cfscript>
```

FileSetLastModified

Description

Sets the date when the file was most recently modified.

Category

System functions

Function syntax

`FileSetLastModified(filepath, date)`

See also

`FileCopy`, `FileDelete`, `FileExists`, `FileMove`, `FileSetAccessMode`, `FileSetAttribute`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`filepath`	An absolute path to a file on the server.
`date`	A valid ColdFusiondate or datetime.

Example

```
<cfscript>
    FileSetLastModified("c:\temp\test1.txt", "#Now()#");
    WriteOutput(#GetFileInfo("c:\temp\test1.txt").lastmodified#);
</cfscript>
```

FileWrite

Description

If you specify a file path, writes the entire content to the file. If you specify a file object, writes text or binary data to the file object.

Category

System functions

Function syntax

```
FileWrite(filepath, data [, charset])

OR

FileWrite(fileobj, data)
```

See also

FileCopy, FileDelete, FileExists, FileMove, cffile

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
charset	The character encoding in which the file contents is encoded. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 If the file starts with a byte order mark and you set this attribute to a conflicting character encoding, ColdFusion generates an error.
data	Content of the file or file object to create.
fileobj	Name of the file object to write.
filepath	Pathname of the file to write. If not an absolute path (starting a with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the GetTempDirectory function.

Example

```
<h3>FileWrite Example</h3>
<!--- This example gets the email addresses of employees, --->
<!--- creates a file object that contains the e-mail addresses, --->
<!--- read the file object, and then creates a text file with a --->
```

```
<!--- list of e-mail addresses. --->

<cfquery name="getemployees" datasource="cfdocexamples">
SELECT EMAIL
FROM Employees
</cfquery>

<cfset companymail = "">

<cfloop query = "getemployees">
 <cfset companymail = companymail & #EMAIL# & ";" & " ">
</cfloop>

<cfscript>
FileWrite("mail_list", "#companymail#");
mlist = FileRead("mail_list");
FileWrite("c:\temp\mail_list.txt", "#mlist#");
</cfscript>
```

FileWriteLine

Description

Appends the specified text to the file object.

Category

System functions

Function syntax

`FileWriteLine(fileobj, text)`

See also

`FileCopy`, `FileDelete`, `FileExists`, `FileMove`, `FileWrite`, `cffile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`text`	Content to add to the file object.
`fileobj`	Pathname of the file to write. If not an absolute path (starting a with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.

Example

```
<h3>FileWriteLine Example</h3>

<cfscript>
    myfile = FileOpen("c:\temp\test1.txt", "read");
    FileWriteLine(myfile,"this line is new");
    FileWrite("c:\temp\mail_list.txt", "#myfile#");
    FileClose(myfile);
</cfscript>
```

Find

Description

Finds the first occurrence of a *substring* in a *string,* from a specified start position. The search is case-sensitive.

Returns

A number; the position of *substring* in *string; or* 0, if *substring* is not in *string*.

Category

String functions

Function syntax

`Find(substring, string [, start ])`

See also

`FindNoCase`, `Compare`, `FindOneOf`, `REFind`, `Replace`

Parameters

Parameter	Description
`substring`	A string or a variable that contains one. String for which to search.
`string`	A string or a variable that contains one. String in which to search.
`start`	Start position of search.

Example

```
<cfoutput>
    <cfset stringToSearch = "The quick brown fox jumped over the lazy dog.">
    #find("the",stringToSearch)#<br>
    #find("the",stringToSearch,35)#<br>
    #find("no such substring",stringToSearch)#<br>
    <br>
    #findnocase("the",stringToSearch)#<br>
    #findnocase("the",stringToSearch,5)#<br>
    #findnocase("no such substring",stringToSearch)#<br>
    <br>
    #findoneof("aeiou",stringToSearch)#<br>
    #findoneof("aeiou",stringToSearch,4)#<br>
    #findoneof("@%^*()",stringToSearch)#<br>
</cfoutput>
```

FindNoCase

Description

Finds the first occurrence of a *substring* in a *string,* from a specified start position. If *substring* is not in *string*, returns zero. The search is case-insensitive.

Returns

The position of *substring* in *string; or* 0, if *substring* is not in *string*.

Category

String functions

Function syntax

`FindNoCase(substring, string [, start ])`

See also

`Find`, `CompareNoCase`, `FindOneOf`, `REFind`, `Replace`

Parameters

Parameter	Description
`substring`	A string or a variable that contains one. String for which to search.
`string`	A string or a variable that contains one. String in which to search.
`start`	Start position of search.

Example

In the following example, the `Find` function returns 33 as the first position found because "the" is lowercase. The `FindNoCase` function returns 1 as the first postion because the case is ignored.

```
<cfset stringToSearch = "The quick brown fox jumped over the lazy dog.">

stringToSearch = <cfoutput>#stringToSearch#</cfoutput><br>
<p>
Find Function:<br>
Find("the",stringToSearch) returns <cfoutput>#find("the",stringToSearch)#</cfoutput><br>
<p>
FindNoCase Function:<br>
FindNoCase("the",stringToSearch) returns
<cfoutput>#FindNoCase("the",stringToSearch)#</cfoutput>
```

FindOneOf

Description

Finds the first occurrence of *any one of a set of characters* in a *string,* from a specified start position. The search is case-sensitive.

Returns

The position of the first member of *set* found in *string; or* 0, if no member of *set* is found in *string*.

Category

String functions

Function syntax

`FindOneOf(set, string [, start ])`

See also

`Find`, `Compare`, `REFind`

Parameters

Parameter	Description
`set`	A string or a variable that contains one. String that contains one or more characters to search for.
`string`	A string or a variable that contains one. String in which to search.
`start`	Start position of search.

Example

```
<cfset stringToSearch = "The quick brown fox jumped over the lazy dog.">
#find("the",stringToSearch)#<br>
#find("the",stringToSearch,35)#<br>
#find("no such substring",stringToSearch)#<br>
<br>
#findnocase("the",stringToSearch)#<br>
#findnocase("the",stringToSearch,5)#<br>
#findnocase("no such substring",stringToSearch)#<br>
<br>
#findoneof("aeiou",stringToSearch)#<br>
#findoneof("aeiou",stringToSearch,4)#<br>
#findoneof("@%^*()",stringToSearch)#<br>
```

FirstDayOfMonth

Description

Determines the ordinal (day number, in the year) of the first day of the month in which a given date falls.

Returns

A number that corresponds to a day-number in a year.

Category

Date and time functions

Function syntax

`FirstDayOfMonth(date)`

See also

`Day`, `DayOfWeek`, `DayOfWeekAsString`, `DayOfYear`, `DaysInMonth`, `DaysInYear`

Parameters

Parameter	Description
`date`	Date/time object, in the range 100 AD–9999 AD.

Usage

When passing a date/time value as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Example

```
<h3>FirstDayOfMonth Example</h3>

<cfoutput>
The first day of #MonthAsString(Month(Now()))#, #Year(Now())# was
day #FirstDayOfMonth(Now())# of the year.
</cfoutput>
```

Fix

Description

Converts a real number to an integer.

Returns

If *number* is greater than or equal to 0, the closest integer less than *number*.

If *number* is less than 0, the closest integer greater than *number*.

Category

Mathematical functions

Function syntax

`Fix(number)`

See also

`Ceiling`, `Int`, `Round`

Parameters

Parameter	Description
`number`	A number

Example

```
<h3>Fix Example</h3>
<p>Fix returns the closest integer less than the number if the number is
    greater than or equal to 0. Fix returns the closest integer greater than
    the number if number is less than 0.</p>
<cfoutput>
<p>The fix of 3.4 is #Fix(3.4)#</p>
<p>The fix of 3 is #Fix(3)#</p>
<p>The fix of 3.8 is #Fix(3.8)#</p>
<p>The fix of -4.2 is #Fix(-4.2)#</p>
</cfoutput>
```

FormatBaseN

Description

Converts *number* to a string, in the base specified by *radix*.

Returns

String that represents *number,* in the base *radix.*

Category

Display and formatting functions, Mathematical functions, String functions

Function syntax

`FormatBaseN(number, radix)`

See also

`InputBaseN`

Parameters

Parameter	Description
`number`	Number to convert
`radix`	Base of the result

Example

```
<h3>FormatBaseN Example</h3>
<p>Converts a number to a string in the base specified by Radix.
<p>
<cfoutput>
<br>FormatBaseN(10,2): #FormatBaseN(10,2)#
<br>FormatBaseN(1024,16): #FormatBaseN(1024,16)#
<br>FormatBaseN(125,10): #FormatBaseN(125,10)#
<br>FormatBaseN(10.75,2): #FormatBaseN(10.75,2)#
</cfoutput>
<h3>InputBaseN Example</h3>
<p>InputBaseN returns the number obtained by converting a string,
    using base specified by Radix (an integer from 2 to 36).</p>

<cfoutput>
<br>InputBaseN("1010",2): #InputBaseN("1010",2)#
<br>InputBaseN("3ff",16): #InputBaseN("3ff",16)#
<br>InputBaseN("125",10): #InputBaseN("125",10)#
<br>InputBaseN(1010,2): #InputBaseN(1010,2)#
</cfoutput>
```

GenerateSecretKey

Description

Gets a secure key value for use in the `Encrypt` function.

Returns

A string that contains the encryption key.

Category

Security functions, String functions

Function syntax

`GenerateSecretKey(algorithm [,keysize])`

See also

`Decrypt`, `Encrypt`

History

ColdFusion 8: Added the `keysize` attribute.

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`algorithm`	The encryption algorithm for which to generate the key. ColdFusion installs a cryptography library with the following algorithms: • AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197. • BLOWFISH: the Blowfish algorithm defined by Bruce Schneier. • DES: the Data Encryption Standard algorithm defined by NIST FIPS-46-3. DESEDE: the "Triple DES" algorithm defined by NIST FIPS-46-3.
`keysize`	Number of bits requested in the key for the specified algorithm. You can use this to request longer keys when allowed by the JDK. For example, the AES algorithm keys are limited to 128 bits unless the Java Unlimited Strength Jurisdiction Policy Files are installed. For more information, see http://java.sun.com/products/jce/index-14.html.

Usage

You cannot use the `GenerateSecretKey` function to generate a key for the ColdFusion default encryption algorithm (CFMX_COMPAT) of the `Encrypt` and `Decrypt` functions.

ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

Example

The following example encrypts and decrypts a text string. It lets you specify the encryption algorithm and encoding technique. It also has a field for a key seed to use with the CFMX_COMPAT algorithm. For all other algorithms, it uses the `GenerateSecretKey` function to generate a secret key.

```
<h3>Decrypt Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>
    <cfscript>
        /* GenerateSecretKey does not generate keys for the CFMX_COMPAT algorithm,
           so we use a key from the form.
        */
        if (Form.myAlgorithm EQ "CFMX_COMPAT")
            theKey=Form.MyKey;
        // For all other encryption techniques, generate a secret key.
        else
            theKey=generateSecretKey(Form.myAlgorithm);
        //Encrypt the string.
        encrypted=encrypt(Form.myString, theKey, Form.myAlgorithm,
                Form.myEncoding);
        //Decrypt it.
        decrypted=decrypt(encrypted, theKey, Form.myAlgorithm, Form.myEncoding);
    </cfscript>

    <!--- Display the values used for encryption and decryption,
            and the results. --->
    <cfoutput>
        <b>The algorithm:</b> #Form.myAlgorithm#<br>
        <b>The key:</B> #theKey#<br>
        <br>
        <b>The string:</b> #Form.myString# <br>
        <br>
        <b>Encrypted:</b> #encrypted#<br>
        <br>
        <b>Decrypted:</b> #decrypted#<br>
    </cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select the encoding</b><br>
    <select size="1" name="myEncoding" >
        <option selected>UU</option>
        <option>Base64</option>
        <option>Hex</option>
    </select><br>
    <br>
    <b>Select the algorithm</b><br>
    <select size="1" name="myAlgorithm" >
        <option selected>CFMX_COMPAT</option>
        <option>AES</option>
        <option>DES</option>
        <option>DESEDE</option>
    </select><br>
    <br>
    <b>Input your key</b> (used for CFMX_COMPAT encryption only)<br>
    <input type = "Text" name = "myKey" value = "foobar"><br>
    <br>
    <b>Enter string to encrypt</b><br>
    <textArea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">This string will be
encrypted (you can replace it with more typing).
    </textArea><br>
    <input type = "Submit" value = "Encrypt my String">
</form>
```

GetAuthUser

Description
Gets the name of an authenticated user.

Returns
The name of an authenticated user.

Category
Security functions

Function syntax
```
GetAuthUser()
```

See also
`cflogin`, `cfloginuser`, `cflogout`, `GetUserRoles`, `IsUserInAnyRole`, `IsUserInRole`, `IsUserLoggedIn`, "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Added this function.

Usage
This function works with `cflogin` authentication or web server authentication. It checks for a logged-in user as follows:

1. It checks for a login made with `cfloginuser`.
2. If no user was logged in with `cfloginuser`, it checks for a web server login (cgi.remote_user).

Example
```
<H3>GetAuthUser Example</H3>

<P>Authenticated User: <cfoutput>#GetAuthUser()#</cfoutput>
```

GetBaseTagData

Description

Used within a custom tag. Finds calling (ancestor) tag by name and accesses its data.

Returns

An object that contains data (variables, scopes, and so on) from an ancestor tag. If there is no ancestor by the specified name, or if the ancestor does not expose data (for example, `cfif`), an exception is thrown.

Category

Other functions

Function syntax

`GetBaseTagData(tagname [, instancenumber ])`

See also

`GetBaseTagList`; "High-level data exchange" on page 202 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`tagname`	(Required) Ancestor tag name for which to return data
`instancenumber`	(Optional) Number of ancestor levels to jump before returning data. The default value is 1 (closest ancestor).

Example

```
<!--- This example shows the use of GetBaseTagData
    function. Typically used in custom tags.--->
...
<cfif trim(inCustomTag) neq "">
        <cfoutput>
            Running in the context of a custom
            tag named #inCustomTag#.<p>
        </cfoutput>
        <!--- Get the tag instance data --->
        <cfset tagData = GetBaseTagData(inCustomTag)>
        <!--- Find the tag's execution mode --->
        Located inside the
        <cfif tagData.thisTag.executionMode neq 'inactive'>
            template
        <cfelse>
            BODY
        </cfif>
```

GetBaseTagList

Description

Gets ancestor tag names, starting with the parent tag.

Returns

A comma-delimited list of uppercase ancestor tag names, as a string. The first list element is the current tag. If the current tag is nested, the next element is the parent tag. If the function is called for a top-level tag, it returns an empty string. If an ancestor does not expose data (see `GetBaseTagData`), its name might not be returned.

Category

Other functions

Function syntax

```
GetBaseTagList()
```

See also

`GetBaseTagData`; "High-level data exchange" on page 202 in the *ColdFusion Developer's Guide*

Usage

This function does not display the following tags or end tags in the ancestor tag list:

- `cfif`, `cfelseif`, `cfelse`
- `cfswitch`, `cfcase`, `cfdefaultcase`
- `cftry`, `cfcatch`

This function displays the following tags only under the following conditions:

- `cfloop`: if it uses a `query` attribute
- `cfoutput`: if at least one of its children is a complex expression
- `cfprocessingdirective`: if it has at least one other attribute besides `pageencoding`

Example

```
<!--- This example shows the use of GetBaseTagList function.
Typically used in custom tags. --->
<cfif thisTag.executionMode is "start">
    <!--- Get the tag context stack
    The list will look something like "CFIF,MYTAGNAME..." --->
    <cfset ancestorList = GetBaseTagList()>
<br><br>Dump of GetBaseTagList output:<br>
    <cfdump var="#ancestorList#"><br><br>
    <!--- Output current tag name --->
    <cfoutput>This is custom tag#ListGetAt(ancestorList,1)#</cfoutput><br>
    <!--- Determine whether this is nested inside a loop --->
    <cfset inLoop = ListFindNoCase(ancestorList, "cfloop")>
    <cfif inLoop>
        Running in the context of a cfloop tag.<br>
    </cfif>
</cfif>
```

GetBaseTemplatePath

Description

Gets the absolute path of an application's base page.

Returns

The absolute path of the application base page, as a string.

Category

Other functions, System functions

Function syntax

```
GetBaseTemplatePath()
```

See also

GetCurrentTemplatePath, FileExists, ExpandPath

Example

```
<h3>GetBaseTemplatePath Example</h3>

<p>The template path of the current page is:
<cfoutput>#GetBaseTemplatePath()#</cfoutput>
```

GetClientVariablesList

Description

Finds the client variables to which a page has write access.

Returns

Comma-delimited list of non-read-only client variables, as a string.

Category

List functions, Other functions

Function syntax

```
GetClientVariablesList()
```

See also

DeleteClientVariable

Usage

The list of variables returned by this function is compatible with ColdFusion list functions.

Example

```
<!--- This example is view-only. --->

<h3>GetClientVariablesList Example</h3>

<p>This view-only example deletes a client variable called "User_ID", if it exists
    in the list of client variables returned by GetClientVariablesList().</p>
<p>This example requires the existence of an Application.cfm file and that client
    management be in effect.</p>
<!---
<cfset client.somevar = "">
<cfset client.user_id = "">
<p>Client variable list:<cfoutput>#GetClientVariablesList()#</cfoutput></p>

<cfif ListFindNoCase(GetClientVariablesList(), "User_ID") is not 0>
<!--- Delete that variable.
    <cfset temp = DeleteClientVariable("User_ID")>
    <p>Was variable "User_ID" Deleted? <cfoutput>#temp#</cfoutput>
</cfif>

<p>Amended Client variable list:<cfoutput>#GetClientVariablesList()#
 </cfoutput>
--->
```

GetComponentMetaData

Description

Gets metadata (such as the functions and implemented interfaces of a component) for a CFC or ColdFusion interface.

Returns

A structure containing the metadata for the CFC or interface. For information on the structure contents, see the component entry in the table in the `GetMetaData` Usage section.

Category

Extensibility functions

Function syntax

`GetComponentMetaData(path)`

See also

`GetMetaData`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`path`	The dot-delimited path of the interface or CFC definition. The path can be relative to the current directory or the web root. For example, if a cfm page that calls this function is in *web_root*/my_apps/interfaces, the interface file is in *web_root*/my_apps/interfaces/definitions, and you want to get the metadata for the interface defined in I2.cfc, specify either of the following values in this parameter: • definitions.I2 • my_apps.interface.definitions.I2.cfc

Usage

This function and the `getMetaData` function return the same data. This function, however, takes a path to the CFC or Interface definition file, and does not use or create an object instance. Also, this function can get data about CFCs and interfaces only, and you cannot specify an interface in the `getMetaData` function.

GetContextRoot

Description

Returns path to the J2EE server context root for the current request.

Returns

The path from the web root to the context root for the current page. The path starts with a forward slash character (/) but does not end with a forward slash character (/). For applications in the default (root) context, returns the empty string.

Category

System functions

History

ColdFusion MX 7: Added this function.

Function syntax

```
GetContextRoot()
```

See also

GetPageContext

Usage

This function is equivalent to calling `GetPageContext().GetRequest().GetContextPath()`. On J2EE configurations, it returns the path from the Web root to the J2EE context root of the ColdFusion J2EE application. On server configurations, it returns the empty string, because the context root is at the web root.

This function is useful in applications that might be installed at varying J2EE context roots.

Example

The ColdFusion Administrator uses the following line to get the location of the administrator directory:

```
<cfset request.thisURL = "#getContextRoot()#/CFIDE/administrator/">
```

The Administrator uses the returned value in places where it uses a URL to access Administrator resources, such as images, as in the following line:

```
<a href="index.cfm"><img src="#request.thisURL#images/back.gif" width="16"
            height="16" border="0" alt=" "></a>
```

GetCurrentTemplatePath

Description

Gets the path of the page that calls this function.

Returns

The absolute path of the page that contains the call to this function, as a string.

Category

System functions

Function syntax

```
GetCurrentTemplatePath()
```

See also

GetBaseTemplatePath, FileExists, ExpandPath

Usage

If the function call is made from a page included with a `cfinclude` tag, this function returns the page path of an included page. Contrast this with the `GetBaseTemplatePath` function, which returns the path of the top-level page, even if it is called from an included page.

Example

```
<!--- This example uses GetCurrentTemplatePath to show the
    template path of the current page --->
<h3>GetCurrentTemplatePath Example</h3>

<p>The template path of the current page is:
<cfoutput>#GetCurrentTemplatePath()#</cfoutput>
```

GetDirectoryFromPath

Description

Extracts a directory from an absolute path.

Returns

Absolute path, without the filename. The last character is a forward or backward slash, depending on the operating system.

Category

System functions

Function syntax

`GetDirectoryFromPath(path)`

See also

`ExpandPath`, `GetFileFromPath`

Parameters

Parameter	Description
`path`	Absolute path (drive, directory, filename, and extension)

Example

```
<h3>GetDirectoryFromPath Example</h3>
<cfset thisPath = ExpandPath("*.*")>
<cfset thisDirectory = GetDirectoryFromPath(thisPath)>
<cfoutput>
The current directory is: #GetDirectoryFromPath(thisPath)#
<cfif IsDefined("FORM.yourFile")>
    <cfif FORM.yourFile is not "">
        <cfset yourFile = FORM.yourFile>
            <cfif FileExists(ExpandPath(yourfile))>
            <p>Your file exists in this directory. You entered the correct filename,
        #GetFileFromPath("#thisPath#/#yourfile#")#
    <cfelse>
        <p>Your file was not found in this directory:
        <br>Here is a list of the other files in this directory:
        <!--- use cfdirectory show directory, order by name & size --->
        <cfdirectory directory = "#thisDirectory#"
            name = "myDirectory" SORT = "name ASC, size DESC">
        <!--- Output the contents of the cfdirectory as a CFTABLE --->
        <cftable query = "myDirectory">
            <cfcol header = "NAME:" text = "#Name#">
        <cfcol header = "SIZE:" text = "#Size#">
        </cftable>
        </cfif>
    </cfif>
<cfelse>
    <H3>Please enter a filename</H3>
    </cfif>
    </cfoutput>
    <form action="getdirectoryfrompath.cfm" METHOD="post">
        <H3>Enter the name of a file in this directory <I><FONT SIZE="-1">
    (try expandpath.cfm)</FONT></I></H3>
        <input type="Text" NAME="yourFile">
```

```
    <input type="Submit" NAME="">
</form> --->
```

GetEncoding

Description

Returns the encoding (character set) of the Form or URL scope.

Returns

String; the character encoding of the specified scope.

Category

International functions, System functions

Function syntax

`GetEncoding(scope_name)`

See also

`SetEncoding`, `cfcontent`, `cfprocessingdirective`, `URLDecode`, `URLEncodedFormat`

History

ColdFusion MX: Added this function.

Parameters

Parameter	Description
`scope_name`	• Form • URL.

Usage

Use this function to determine the character encoding of the URL query string or the fields of a form that was submitted to the current page. The default encoding, if none has been explicitly set, is UTF-8.

For more information, see www.iana.org/assignments/character-sets.

Example

```
<!--- This example sends the contents of two fields and interprets them as
        big5 encoded text. Note that the form fields are received as URL variables because
the form uses the GET method.--->
<cfcontent type="text/html; charset=big5">
<form action='#cgi.script_name#' method='get'>
<input name='xxx' type='text'>
<input name='yyy' type='text'>
<input type="Submit" value="Submit">
</form>

<cfif IsDefined("URL.xxx")>
<cfscript>
    SetEncoding("url", "big5");
    WriteOutput("URL.XXX is " & URL.xxx & "<br>");
    WriteOutput("URL.YYY is " & URL.yyy & "<br>");
theEncoding = GetEncoding("URL");
     WriteOutput("The URL variables were decoded using '" & theEncoding & "' encoding.");

WriteOutput("The encoding is " & theEncoding);
</cfscript>
</cfif>
```

GetException

Description

Used with the `cftry` and `cfcatch` tags. Retrieves a Java exception object from a Java object.

Returns

Any Java exception object raised by a previous method call on the Java object.

Category

System functions

Syntax

`GetException(object)`

Parameters

Parameter	Description
`object`	A Java object.

Usage

ColdFusion stores a Java exception object for each method call on a Java object. Subsequent method calls reset the exception object. To get the current exception object, you must call `GetException` on the Java object before other methods are invoked on it.

Example

```
<!--- Create the Java object reference --->
<cfobject action = create type = java class = primativetype name = myObj>
<!--- Calls the object's constructor --->
<cfset void = myObj.init()>
<cftry>
<cfset void = myObj.DoException() >
<Cfcatch type = "Any">
    <cfset exception = GetException(myObj)>
<!--- User can call any valid method on the exception object.--->
    <cfset message = exception.toString()>
    <cfoutput>
        Error<br>
        I got exception <br>
        <br> The exception message is: #message# <br>
    </cfoutput>
</cfcatch>
</cftry>
```

GetFileInfo

Description
Retrieves information about a file.

Returns
The filename, path, parent directory, type, size, when the file was most recently modified, whether the file has read permission, write permission, and is hidden.

Category
System functions

Function syntax
`GetFileInfo(path)`

See also
`FileOpen`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`path`	Absolute path (drive, directory, filename, and extension)

Usage
The function returns a structure that includes the following keys:

- Name: name of the file
- Path: absolute path of the file
- Parent: path to the file's parent directory
- Type: either "directory" or "file"
- Size: file size in bytes
- Lastmodified: datetime when it was the file was most recently modified
- canRead: whether the file can be read
- canWrite: whether the file has write permission
- isHidden: whether the file is a hidden

Example
```
<cfscript>
    FileSetLastModified("c:\temp\test1.txt", "#Now()#");
    WriteOutput(#GetFileInfo("c:\temp\test1.txt").lastmodified#);
</cfscript>
```

GetFileFromPath

Description

Extracts a filename from an absolute path.

Returns

Filename, as a string.

Category

System functions

Function syntax

`GetFileFromPath(path)`

See also

`ExpandPath`, `GetCurrentTemplatePath`

Parameters

Parameter	Description
`path`	Absolute path (drive, directory, filename, and extension)

Example

```
<h3>GetFileFromPath Example</h3>
<cfset thisPath = ExpandPath("*.*")>
<cfset thisDirectory = GetDirectoryFromPath(thisPath)>
<cfoutput>
The current directory is: #GetDirectoryFromPath(thisPath)#
<cfif IsDefined("FORM.yourFile")>
<cfif FORM.yourFile is not "">
<cfset yourFile = FORM.yourFile>
<cfif FileExists(ExpandPath(yourfile))>
        <p>Your file exists in this directory. You entered the correct file
    name, #GetFileFromPath("#thisPath#/#yourfile#")#
    <cfelse>
        <p>Your file was not found in this directory:
        <br>Here is a list of the other files in this directory:
        <!--- use CFDIRECTORY to give the contents of the snippets
    directory, order by name and size --->
        <CFDIRECTORY
            DIRECTORY = "#thisDirectory#"
            name = "myDirectory"
            SORT = "name ASC, size DESC">
        <!--- Output the contents of the CFDIRECTORY as a CFTABLE --->
        <CFTABLE QUERY = "myDirectory">
        <CFCOL HEADER = "NAME:" TEXT = "#Name#">
    <CFCOL HEADER = "SIZE:" TEXT = "#Size#">
...
```

GetFunctionList

Description

Displays a list of the functions that are available in ColdFusion.

Returns

A structure of functions.

Category

System functions

Function syntax

GetFunctionList()

Example

```
<!----- This example shows the use of GetFunctionList. ---->
<cfset fList = GetFunctionList()>
<cfoutput>#StructCount(fList)# functions<br><br>
    </cfoutput>
<cfloop COLLECTION = "#fList#" ITEM = "key">
    <cfoutput>#key#<br>
    </cfoutput>
</cfloop>
```

GetGatewayHelper

Description
Gets a Java GatewayHelper object that provides methods and properties for use with a ColdFusion event gateway.

Returns
A Java GatewayHelper object.

Category
Extensibility functions

Function syntax
`GetGatewayHelper(gatewayID)`

See also
`SendGatewayMessage`

History
ColdFusion MX 7: Added the function.

Parameters

Parameter	Description
`gatewayID`	Identifier of the gateway that provides the GatewayHelper object. Must be the Gateway ID of one of the ColdFusion event gateway instances configured on the ColdFusion Administrator Event Gateways section's Gateways page.

Usage
The ColdFusion `GetGatewayHelper` function returns a Java GatewayHelper object that provides event gateway-specific helper methods and properties. To use this function, the event gateway must provide access to a class that implements the GatewayHelper class. For example, an instant messaging event gateway might make buddy-list management functions available in a GatewayHelper object.

An event gateway listener CFC can get the `gatewayID` value from the CFEvent structure of the incoming message.

You access the GatewayHelper object's methods and properties using standard ColdFusion Java object access techniques. For more information, see "The role of the GatewayHelper object" on page 1069 in the *ColdFusion Developer's Guide*.

Example
If an event gateway's helper class includes an `addBuddy` method that takes a single String parameter, you could use the following code to get the GatewayHelper object and add a buddy to the buddies list:

```
<h3>GetGatewayHelper Example</h3>
<cfscript>
    myHelper = getGatewayHelper(myGatewayID);
    status = myHelper.addBuddy("jsmith");
</cfscript>
```

GetHttpRequestData

Description

Makes HTTP request headers and body available to CFML pages. Useful for capturing SOAP request data, which can be delivered in an HTTP header.

Returns

A ColdFusion structure.

Category

System functions

Function syntax

```
GetHttpRequestData()
```

Returns

The function returns a structure that contains the following entries:

Structure element	Description
content	Raw content from form submitted by client, in string or binary format. For content to be considered string data, the FORM request header "CONTENT_TYPE" must start with "text/" or be special case "application/x-www-form-urlencoded". Other types are stored as a binary object.
headers	Structure that contains HTTP request headers as value pairs. Includes custom headers, such as SOAP requests.
method	String that contains the CGI variable Request_Method.
protocol	String that contains the Server_Protocol CGI variable.

Usage

To determine whether data is binary, use `IsBinary(x.content)`. To convert data to a string value, if it can be displayed as a string, use `ToString(x.content)`.

Example

The following example shows how this function can return HTTP header information.

```
<cfset x = GetHttpRequestData()>
<cfoutput>
<table cellpadding = "2" cellspacing = "2">
<tr>
     <td><b>HTTP Request item</b></td>
     <td><b>Value</b></td> </tr>
<cfloop collection = #x.headers# item = "http_item">
    <tr>
        <td>#http_item#</td>
        <td>#StructFind(x.headers, http_item)#</td></tr>
</cfloop>
<tr>
    <td>request_method</td>
    <td>#x.method#</td></tr>
<tr>
    <td>server_protocol</td>
    <td>#x.protocol#</td></tr>
</table>
<b>http_content --- #x.content#</b>
</cfoutput>
```

GetHttpTimeString

Description

Gets the current time, in the Universal Time code (UTC).

Returns

The time, as a string, according to the HTTP standard described in RFC 1123 and its underlying RFC, 822. This format is commonly used in Internet protocols, including HTTP.

Category

Date and time functions, International functions

Function syntax

`GetHttpTimeString(date_time_object)`

See also

`GetLocale`, `GetTimeZoneInfo`, `SetLocale`

Parameters

Parameter	Description
`date_time_object`	A ColdFusion date-time object string or Java Date object

Usage

The time in the returned string is UTC, consistent with the HTTP standard.

Example

```
<cfoutput>
    #GetHttpTimeString(now())#
</cfoutput>
```

GetK2ServerDocCount

Description

This function is deprecated.

Returns

The number of collection metadata items stored in Verity collections.

Category

Full-text search functions, Query functions

Function syntax

```
GetK2ServerDocCount()
```

See also

GetK2ServerDocCountLimit

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.

ColdFusion MX: Added this function.

Example

```
<cfoutput>GetK2ServerDocCount =
    $*#GetK2ServerDocCount()#*$</cfoutput>
```

GetK2ServerDocCountLimit

Description

This function is deprecated.

Returns

Number of collection metadata items that the K2 server permits, as an integer

Category

Full-text search functions, Query functions

Function syntax

```
GetK2ServerDocCountLimit()
```

See also

GetK2ServerDocCount

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.

ColdFusion MX: Added this function.

Usage

If a search generates a larger number of documents than the limit, ColdFusion puts a warning message in the Administrator and in the log file.

Example

```
<cfoutput>GetK2ServerDocCountLimit =
    $*#GetK2ServerDocCountLimit()#*$</cfoutput>
```

GetLocale

Description

Gets the current ColdFusion geographic/language locale value.

To set the default display format of date, time, number, and currency values in a ColdFusion application session, you use the `SetLocale` function.

Returns

The current locale value, as an English string. If a locale has a Java name and a name that ColdFusion used prior to the ColdFusion MX 7 release (for example, en_US and English (US)), ColdFusion returns the ColdFusion name (for example, English (US)).

Category

Display and formatting functions, International functions, System functions

Function syntax

```
GetLocale()
```

See also

`GetLocaleDisplayName`, `SetLocale`

History

ColdFusion MX 7: Added support for all Java locales and locale names.

ColdFusion MX: Changed behavior to that described in usage.

Usage

This function returns the locale name as it is represented in ColdFusion; for example, Portuguese (Brazilian), or ca_ES. To get a locale name in the language of the locale, use the `GetLocaleDisplayName` function, which returns português (Brasil) and(Espanya).

This function determines whether a locale value is set for ColdFusion. (The value is set with the `SetLocale` function.)

- If the ColdFusion locale value is present, the function returns it.
- If the ColdFusion locale has not been explicitly set, ColdFusion now determines whether the default locale of the ColdFusion server computer operating system is among the locale values it supports. (The default locale is stored in the user environment variables user.language and user.region.)

  If the default locale value is not supported, the function returns English (US). ColdFusion sets the locale in the JVM to this value; it persists until the server is restarted or it is reset with the `SetLocale` function.

This function does not access a web browser's Accept-Language HTTP header setting.

Note: *When ColdFusion is started, it stores the supported locale values in the variable Server.ColdFusion.SupportedLocales. ColdFusion supports the locales supported by its Java runtime environment. The SupportedLocales value lists the Java names for the supported locales and the corresponding names that ColdFusion used prior to the ColdFusion MX 7 release.*

For more information, see "Locales" on page 341 in the *ColdFusion Developer's Guide*.

Example

```
<h3>Example: Using SetLocale and GetLocale</h3>
<cfoutput>
    <!--- For each new request, the locale gets reset to the JVM locale --->
    Initial locale's ColdFusion name: #GetLocale()#<br>
    <br>
    <!--- Do this only if the form was submitted. --->
    <cfif IsDefined("form.mylocale")>
        <b>Changing locale to #form.mylocale#</b><br>
        <br>
        <!--- Set the locale to the submitted value and save the old ColdFusion locale name.-
-->
        <cfset oldlocale=SetLocale("#form.mylocale#")>
        <!--- Get the current locale. It should have changed. --->
        New locale: #GetLocale()#<br>
    </cfif>

    <!--- Self-submitting form to select the new locale. --->
    <cfform>
        <h3>Please select the new locale:</h3>
        <cfselect name="mylocale">
            <!--- The server.coldfusion.supportedlocales variable is a
                    list of all supported locale names. Use a list cfloop tag
                    to create an HTML option tag for each name in the list. --->
            <cfloop index="i" list="#server.coldfusion.supportedlocales#">
                <option value="#i#">#i#</option>
            </cfloop>
        </cfselect><br>
        <br>
        <cfinput type="submit" name="submitit" value="Change Locale">
    </cfform>
</cfoutput>
```

GetLocaleDisplayName

Description

Gets a locale value and displays the name in a manner that is appropriate to a specific locale. By default, gets the current locale in the current locale's language.

Returns

The localized display name of the locale, in the language of the specified locale.

Category

Display and formatting functions, International functions, System functions

Function syntax

```
GetLocaleDisplayName([locale, inLocale])
```

See also

GetLocale, SetLocale

History

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
locale	The locale whose name you want. The default is the current ColdFusion locale, or if no ColdFusion locale has been set, the JVM locale.
inlocale	The locale in which to return the name. The default is the current ColdFusion locale, or if no ColdFusion locale has been set, the JVM locale.

Example

The following example expands on the GetLocale example to show the use of the GetLocaleDisplayName function to display locale names in the current locale and in other locales. It lets you select a locale from all supported locales, changes the ColdFusion locale to the selected locales, and displays the old and new locale names.

```
<html>
<head>
    <title>Displaying locales</title>
</head>

<body>
<h3>Example: Changing and Displaying Locales</h3>
<cfoutput>
    <!--- For each new request, the locale gets reset to the JVM locale --->
    Initial locale's ColdFusion name: #GetLocale()#<br>
    Initial locale's display name: #GetLocaleDisplayName()#<br>
    <br>
    <!--- Do this only if the form was submitted. --->
    <cfif IsDefined("form.mylocale")>
        <b>Changing locale to #form.mylocale#</b><br>
        <br>
        <!--- Set the locale to the submitted value.
            SetLocale returns the old ColdFusion locale name.--->
        <cfset oldlocale=SetLocale("#form.mylocale#")>
        <!--- Get the current locale's ColdFusion name.
            It should have changed. --->
```

```
        <cfset newlocale=GetLocale()>
        New locale's ColdFusion name: #newlocale#<br>
        New locale's display name in current locale: #GetLocaleDisplayName()#<br>
        New locale's display name in old locale:
            #GetLocaleDisplayName(newlocale, oldlocale)#<br>
        New locale's display name in en_US:
            #GetLocaleDisplayName(newlocale, "en_US")#<br>
        <br>
        Old locale's display name in current locale:
            #GetLocaleDisplayName(oldlocale)#<br>
        Old locale's display name in en_US:
            #GetLocaleDisplayName(oldlocale, "en_US")#<br>
    </cfif>

    <!--- Self-submitting form to select the new locale. --->
    <cfform>
        <h3>Please select the new locale:</h3>
        <cfselect name="mylocale">
            <!--- The server.coldfusion.supportedlocales variable is a
                    list of all supported locale names. Use a list cfloop tag
                    to create an HTML option tag for each name in the list. --->
            <cfloop index="i" list="#server.coldfusion.supportedlocales#">
            <!--- In the select box, we use the US English display names for
                    the locales. You can change en_US to your prefered locale. --->
                <option value="#i#">#GetLocaleDisplayName(i, "en_US")#</option>
            </cfloop>
        </cfselect><br>
        <br>
        <cfinput type="submit" name="submitit" value="Change Locale">
    </cfform>
</cfoutput>

</body>
</html>
```

GetLocalHostIP

Description

Returns the localhost IP address, which is `127.0.0.1` for IPv4 and `::1` for IPv6 addresses.

Returns

The localhost IP address.

Category

Decision functions

Function syntax

```
GetLocalHostIP()
```

See also

`IsLocalHost`

History

ColdFusion MX 7.01 : Added this function.

Example

```
<h3>GetLocalHostIP Example</h3>

<cfoutput>The localhost IP address is #GetLocalHostIP()#.</cfoutput>
```

GetMetaData

Description

Gets metadata (such as the methods, properties, and parameters of a component) associated with an object that is deployed on the ColdFusion server.

Returns

Structured metadata information: for ColdFusion components (CFCs) and user defined functions (UDFs), a structure; for query objects, an array of structures.

Category

System functions

Function syntax

`GetMetaData(object)`

See also

`CreateObject`, `GetComponentMetaData`, `QueryAddColumn`, `QueryNew`

History

ColdFusion MX 7: Added support for getting query object metadata.

ColdFusion MX: Added this function.

Parameters

Parameter	Description
`object`	A ColdFusion component, user-defined function, or query object. Within a CFC, the parameter can also specify the This scope.

Usage

This function provides information about application data, and lets applications dynamically determine the structure of an object and how to use it. This function is useful for CFCs and query objects. The metadata for a CFC includes information on the component and its functions, arguments, and properties. The `getMetaData` function also returns metadata for user-defined functions that are not part of CFCs. Use the `GetComponentMetaData` function to get information about ColdFusion interfaces, or about CFC definitions that you have not instantiated.

The following table lists the data returned by the function for supported object types:

Object	Field	Description
Component		A structure containing the following fields:
	displayname	Value of the cfcomponent tag displayname attribute, if any.
	extends	Metadata for the component's ancestor component. Components that do not explicitly extend another component extend the WEB-INF.cftags.component.
	fullname	The dot delimited path from the *cf_webroot* of the component.
	functions	Array of metadata structures for the component's functions.
	hint	Value of the cfcomponent tag displayname attribute, if any.
	implements	A structure containing the metadata of the interfaces that the component implements. The key of the structure is the interface name, and the value is a structure containing the interface metadata.
	name	Component name, including the period-delimited path from a component search root such as the web root or a directory specified in the administrator Custom Tag Paths page.
	output	Value of the cfcomponent tag output attribute, if any
	path	Absolute path to the component.
	properties	Array of structures containing metadata specified by the component's cfproperty tags, if any.
	type	Always, component.
	userMetadata	User-specified attributes in the cfcomponent tag
Function		A structure containing the following fields.
	access	Value of the cffunction tag access attribute, if any.
	displayname	Value of the cffunction tag displayname attribute, if any.
	hint	Value of the cffunction tag hint attribute, if any.
	name	Function name.
	output	Value of the cffunction tag output attribute, if any.
	parameters	Array of structures containing metadata for the function parameters.
	returntype	Value of the cffunction tag returntype attribute, if any.
	roles	Value of the cffunction tag output attribute, if any.
	userMetadata	User-specified attributes in the cffunction tag.
Parameter or Property		A structure containing the following fields:
	default	Value of the cfargument or cfproperty tag default attribute, if any.
	displayname	Value of the cfargument or cfproperty tag displayname attribute, if any.
	hint	Value of the cfargument or cfproperty tag hint attribute, if any.
	name	Function parameter or CFC property name.
	required	Value of the cfargument or cfproperty tag required attribute, if any.
	type	Value of the cfargument or cfproperty tag type attribute, if any.
	userMetadata	User-specified attributes in the argument tag.

Object	Field	Description
Query		An array of structures containing the following elements:
	IsCaseSensitive	Boolean value indicating whether character data must be case correct.
	Name	The column name.
	TypeName	The SQL data type (Omitted if the query object is created with QueryNew without specifying types.)

Note: *Use the This scope to access component metadata inside the CFC. The This scope is available at runtime in the component body and in the CFC methods. It is used to read and write variables that are present during the life of the component.*

For more information, see "Using introspection to get information about components" on page 186 in the *ColdFusion Developer's Guide*.

Example

The following example uses the `cfdump` tag to display metadata for the utilities CFC that is used by the ColdFusion component browser. It also displays the names and data types of the fields in the cfdocexamples database Employees table.

```
<!--- Create an instance of the Component Explorer utilities CFC.
        and get its metadata --->
<cfscript>
componentutils = createObject("component", "cfide.componentutils.utils");
utilmetadata = getMetaData(componentutils);
</cfscript>

<h4>Metadata for the CFC component utilities</h4>
<cfdump var="#utilmetadata#">

<!--- use GetMetadata to get the names and data types of the fields in the
           cfdocexamples Employees table --->
<cfquery name="getemployees" datasource="cfdocexamples">
SELECT *
FROM Employees
</cfquery>
<cfset employeemeta=getMetaData(getemployees)>

<h4>The Employees table has the following columns</h4>
<cfloop index="i" from="1" to="#arrayLen(employeemeta)#">
    <cfoutput>
        #employeemeta[i].name# #employeemeta[i].TypeName#
#employeemeta[i].isCaseSensitive#<br>
    </cfoutput>
</cfloop>
```

GetMetricData

Description
Gets server performance metrics.

Returns
ColdFusion structure that contains metric data, depending on the `mode` value.

Category
System functions

Function syntax
`GetMetricData(mode)`

History
ColdFusion MX: Deprecated the `cachepops` parameter. It might not work, and it might cause an error, in later releases.

Parameters

Parameter	Option	Description
`mode`	`perf_monitor`	Returns internal data, in a structure. To receive data, you must enable PerfMonitor in ColdFusion Administrator before executing the function. On Windows, this data is otherwise displayed in the Windows PerfMonitor.
	`simple_load`	Returns an integer value that is computed from the state of the server's internal queues. Indicates the overall server load.
	`prev_req_time`	Returns the time, in milliseconds, that it took the server to process the previous request.
	`avg_req_time`	Returns the average time, in milliseconds, that it takes the server to process a request. Changing the setting to 0 prevents the server from calculating the average and removes overhead associated with gathering data. The default value is 120 seconds.

Usage
If `mode = "perf_monitor"`, the function returns a structure with these data fields:

Field	Description
`InstanceName`	The name of the ColdFusion server. The default value is cfserver.
`PageHits`	Number of HTTP requests received since ColdFusion MX was started.
`ReqQueued`	Number of HTTP requests in the staging queue, waiting for processing.
`DBHits`	Number of database requests since the server was started.
`ReqRunning`	Number of HTTP requests currently running. In the ColdFusion Administrator, you can set the maximum number of requests that run concurrently.
`ReqTimedOut`	Number of HTTP requests that timed out while in the staging queue or during processing.
`BytesIn`	Number of bytes in HTTP requests to ColdFusion MX.

Field	Description
BytesOut	Number of bytes in HTTP responses from ColdFusion MX.
AvgQueueTime	For the last two HTTP requests (current and previous), the average length of time the request waited in the staging queue.
AvgReqTime	For the last two HTTP requests (current and previous), the average length of time the server required to process the request
AvgDBTime	For the last two HTTP requests (current and previous), the average length of time the server took to process CFQueries in the request.
cachepops	This parameter is deprecated. ColdFusion automatically sets its value to -1.

Example

```
<!--- This example gets and displays metric data from Windows NT PerfMonitor --->
<cfset pmData = GetMetricData( "PERF_MONITOR" ) >
<cfoutput>
    Current PerfMonitor data is: <p>
    InstanceName:#pmData.InstanceName# <p>
    PageHits:#pmData.PageHits# <p>
    ReqQueued: #pmData.ReqQueued# <p>
    DBHits: #pmData.DBHits# <p>
    ReqRunning: #pmData.ReqRunning# <p>
    ReqTimedOut: #pmData.ReqTimedOut# <p>
    BytesIn: #pmData.BytesIn# <p>
    BytesOut: #pmData.BytesOut# <p>
    AvgQueueTime: #pmData.AvgQueueTime# <p>
    AvgReqTime: #pmData.AvgReqTime# <p>
    AvgDBTime: #pmData.AvgDBTime# <p>
</cfoutput>
```

GetPageContext

Description

Gets the current ColdFusion PageContext object that provides access to page attributes and configuration, request, and response objects.

Returns

The current ColdFusion Java PageContext Java object.

Category

System functions

Function syntax

```
GetPageContext()
```

History

ColdFusion MX: Added this function.

Usage

The ColdFusion PageContext class is a wrapper class for the Java PageContext object that can resolve scopes and perform case-insensitive variable lookups.

The PageContext object exposes fields and methods that can be useful in J2EE integration. It includes the `include` and `forward` methods that provide the equivalent of the corresponding standard JSP tags. You use these methods to call JSP pages and servlets. For example, you use the following code in CFScript to include the JSP page hello.jsp and pass it a `name` parameter:

```
GetPageContext().include("hello.jsp?name=Bobby"); ===
```

When you use `GetPageContext` to include a JSP page in a CFML page on WebLogic, you may need to flush the output of the CFML page with `cfflush` before calling the JSP page. Otherwise, the ColdFusion output appears after the JSP output.

The methods supported on the returned PageContext are only those mandated by the JSP specification. To look up scopes by name, use the StructGet function, for example:

```
<cfset myscope = "server">
<cfset myserver = StructGet(myscope)>
```

For more information, see your Java Server Pages (JSP) documentation.

Note: *On Weblogic, you may need to flush the output of the CFML page (using `cfflush`) before calling a JSP page. If you do not, the ColdFusion output appears after the JSP output.*

Example

```
<!--- This example shows using the page context to set a page
    variable and access the language of the current locale. --->
<cfset pc = GetPageContext()>

<cfset pc.setAttribute("name","John Doe")>
<cfoutput>name: #variables.name#<br></cfoutput>

<cfoutput>Language of the current locale is
    #pc.getRequest().getLocale().getDisplayLanguage()#</cfoutput>.
```

GetPrinterInfo

Description
Determines which print attributes are supported by a selected printer.

Returns
A structure that contains the attributes supported by the printer. If the printer is an empty string, the structure contains attributes supported by the default printer, if one exists.

Category
System functions

Function syntax
`GetPrinterInfo("printer")`

See also
`cfpdf`, `cfprint`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`printer`	The name of a printer. An example in Windows is "\\s1001prn02\NTN-2W-HP_BW02". The default is the default printer for the account where the ColdFusion server is running. Printer names are case-sensitive and must be entered exactly as they appear in the System Information page of the ColdFusion Administrator. If you specify an empty string, for example `GetPrinterInfo("")`, ColdFusion generates and error. Use the following code to retrieve information on the default printer: `GetPrinterInfo()`.

Usage
Use this function in conjunction with the `cfprint` tag when processing large print jobs. Not all printers support the complete list of print attributes allowed by the `cfprint` tag. If the selected printer does not support a print attribute, the printer ignores the attribute.

For more information, see "attributeStruct" on page 483.

Example
```
<!--- The following code returns information on the default printer. --->
<cfdump var="#GetPrinterInfo()#">

<!--- The following code returns information on the specified printer. --->
<cfdump var="#GetPrinterInfo('\\S1001prn02\NTN-2W-SHARP01')#">
```

GetProfileSections

Description

Gets all the sections of an initialization file.

An initialization file assigns values to configuration variables, also known as entries, that are set when the system boots, the operating system comes up, or an application starts. An initialization file has the suffix INI; for example, boot.ini, Win32.ini.

Returns

An initialization file, as a structure whose format is as follows:

- Each initialization file section name is a key in the structure
- Each list of entries in a section of an initialization file is a value in the structure

If there is no value, returns an empty string.

Category

System functions

Function syntax

`GetProfileSections(iniFile)`

See also

`GetProfileString`, `SetProfileString`

History

ColdFusion MX: Added this function.

Parameters

Parameter	Description
`iniFile`	Absolute path (drive, directory, filename, extension) of initialization file; for example, C:\boot.ini

GetProfileString

Description

Gets an initialization file entry.

An initialization file assigns values to configuration variables, also known as entries, that are set when the system boots, the operating system comes up, or an application starts. An initialization file has the suffix INI; for example, boot.ini, Win32.ini.

Returns

An entry in an initialization file, as a string. If there is no value, returns an empty string.

Category

System functions

Function syntax

```
GetProfileString(iniPath, section, entry)
```

See also

GetProfileSections, GetProfileString, SetProfileString

Parameters

Parameter	Description
iniPath	Absolute path (drive, directory, filename, extension) of initialization file; for example, C:\boot.ini
section	Section of initialization file from which to extract information
entry	Name of value to get

Example

```
<h3>GetProfileString Example</h3>
Uses GetProfileString to get the value of timeout in an initialization file. Enter
    the full path of your initialization file, and submit the form.
<!--- If the form was submitted, it gets the initialization path and timeout value specified
in the form --->
<cfif Isdefined("Form.Submit")>
    <cfset IniPath = FORM.iniPath>
    <cfset Section = "boot loader">
    <cfset timeout = GetProfileString(IniPath, Section, "timeout")>
    <h4>Boot Loader</h4>
    <!--- If no entry in an initialization file, nothing displays --->
    <p>Timeout is set to: <cfoutput>#timeout#</cfoutput>.</p>
</cfif>
<form action = "getprofilestring.cfm">
<table cellspacing = "2" cellpadding = "2" border = "0">
    <tr>
         <td>Full Path of Init File</td>
         <td><input type = "Text" name = "IniPath" value = "C:\myboot.ini">
        </td>
    </tr>
    <tr>
     <td><input type = "Submit" name = "Submit" value = "Submit"></td>
     <td></td>
    </tr></table>
</form>
```

GetReadableImageFormats

Description
Returns a list of image formats that ColdFusion can read on the operating system where ColdFusion is deployed.

Returns
A list of image file formats.

Category
System functions

History
ColdFusion 8: Added this function.

Function syntax
```
GetReadableImageFormats()
```

See also
`GetWriteableImageFormats`"Supported image file formats" on page 304

Usage
Use this function to determine image file compatibility on the ColdFusion server.

Example
```
<cfoutput>#GetReadableImageFormats()#</cfoutput>
```

GetSOAPRequest

Description
Returns an XML object that contains the entire SOAP request. Usually called from within a web service CFC.

Returns
An XML object that contains the entire SOAP request.

Category
XML functions

History
ColdFusion MX 7: Added this function.

Function syntax
`GetSOAPRequest()`

See also
`AddSOAPRequestHeader`, `AddSOAPResponseHeader`, `GetSOAPRequestHeader`, `GetSOAPResponse`, `GetSOAPResponseHeader`, `IsSOAPRequest`; "Basic web service concepts" on page 903 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`webservice`	Optional. A `webservice` object as returned from the `cfobject` tag or the `CreateObject` function. Required if the function is called from the client.

Usage
Call this function to obtain a web service request object after the web service has been invoked. If you call this function from outside a web service CFC without the `webservice` parameter, it throws the following expression error:

```
Unable to use getSOAPRequest: not processing a web service request.
```

If you call this function from within a web service CFC, you can omit the `webservice` argument. The function executes against the request that it is currently processing.

You can use CFML XML functions to examine the returned XML object.

Example
This example makes a request to execute the `echo_me` function of the headerservice.cfc web service. For information on implementing the headerservice.cfc web service and also to see the `echo_me` function and the content of the web service CFC, see the example for either the `AddSOAPResponseHeader` function or the `GetSOAPRequestHeader` function.

```
<!--- Note that you might need to modify the URL in the CreateObject function
to match your server and the location of the headerservice.cfc file if it is
different than shown here.
Note, too, that getSOAPRequest is called from the client here, whereas often it
would be called from within the web service CFC. --->

<cfscript>
    ws = CreateObject("webservice",
"http://localhost/soapheaders/headerservice.cfc?WSDL");
```

```
    ws.echo_me("hello world");
    req = getSOAPRequest(ws);
</cfscript>
<cfdump var="#req#">
```

GetSOAPRequestHeader

Description

Obtains a SOAP request header. Call only from within a CFC web service function that is processing a request as a SOAP web service.

Returns

A SOAP request header.

Category

XML functions

History

ColdFusion MX 7: Added this function.

Function syntax

`GetSOAPRequestHeader(namespace, name [, asXML])`

See also

`AddSOAPRequestHeader`, `AddSOAPResponseHeader`, `GetSOAPRequest`, `GetSOAPResponse`, `GetSOAPResponseHeader`, `IsSOAPRequest`; "Basic web service concepts" on page 903 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`namespace`	A String that is the namespace for the header.
`name`	A String that is the name of the header.
`asXML`	If true, the header is returned as a CFML XML object; if false (default), the header is returned as a Java object.

Usage

If you specify `false` for the `asXML` parameter, ColdFusion first attempts to retrieve the header using the data type specified in the header's `xsi:type` attribute. If the `xsi:type` attribute is not available, ColdFusion attempts to retrieve the header as a string. If you specify `true` for the `asXML` parameter, ColdFusion retrieves the header as raw XML.

This function throws an error if it is invoked in a context that is not a web service request. Use the `IsSOAPRequest` function to determine whether the CFC is running as a web service.

Example

This example creates a CFC web service that illustrates the operation of the `GetSOAPRequestHeader` function and also provides a web service that illustrates the operation of other ColdFusion SOAP functions.

Save the following code as headerservice.cfc in a folder called soapheaders under your web root. Test its operation, and specifically the operation of the `GetSOAPRequestHeader` function, by executing the examples that invoke this web service. For example, see the example for `AddSOAPRequestHeader`.

```
<h3>GetSOAPRequestHeader Example</h3>
<cfcomponent displayName="tester" hint="Test for SOAP headers">

<cffunction name="echo_me"
    access="remote"
    output="false"
```

```
    returntype="string"
    displayname="Echo Test" hint="Header test">

<cfargument name="in_here" required="true" type="string">

<cfset isSOAP = isSOAPRequest()>
<cfif isSOAP>

    <!--- Get the first header as a string and as XML --->
    <cfset username = getSOAPRequestHeader("http://mynamespace/", "username")>
    <cfset return = "The service saw username: " & username>
    <cfset xmlusername = getSOAPRequestHeader("http://mynamespace/", "username", "TRUE")>
    <cfset return = return & "<br> as XML: " & xmlusername>

    <!--- Get the second header as a string and as XML --->
    <cfset password = getSOAPRequestHeader("http://mynamespace/", "password")>
    <cfset return = return & "The service saw password: " & password>
    <cfset xmlpassword = getSOAPRequestHeader("http://mynamespace/", "password", "TRUE")>
    <cfset return = return & "<br> as XML: " & xmlpassword>

    <!--- Add a header as a string --->
<cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED
VALUE", false)>

    <!--- Add a second header using a CFML XML value --->
    <cfset doc = XmlNew()>
    <cfset x = XmlElemNew(doc, "http://www.tomj.org/myns", "returnheader2")>
    <cfset x.XmlText = "hey man, here I am in XML">
    <cfsetx.XmlAttributes["xsi:type"] = "xsd:string">
    <cfset tmp = addSOAPResponseHeader("ignoredNameSpace", "ignoredName", x)>

<cfelse>
    <!--- Add a header as a string - Must generate error!
<cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED
VALUE", false)>
    --->
<cfset return = "Not invoked as a web service">
</cfif>

<cfreturn return>

</cffunction>

</cfcomponent>
```

GetSOAPResponse

Description
Returns an XML object that contains the entire SOAP response after invoking a web service.

Returns
An XML object that contains the entire SOAP response.

Category
XML functions

History
ColdFusion MX 7: Added this function.

Function syntax
`GetSOAPResponse(webservice)`

See also
`AddSOAPRequestHeader`, `AddSOAPResponseHeader`, `GetSOAPRequest`, `GetSOAPRequestHeader`, `GetSOAPResponseHeader`, `IsSOAPRequest`; "Basic web service concepts" on page 903 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`webservice`	A webservice object as returned from the `cfobject` tag or the `CreateObject` function.

Usage
You must first invoke the web service before attempting to get the response. You can use CFML XML functions to examine the XML response.

Example
This example makes a request to execute the `echo_me` function of the headerservice.cfc web service. Following the request, the example calls the `GetSOAPResponse` function to get the SOAP response, and then calls `cfdump` to display its content.

for information on implementing the headerservice.cfc web service and also to see the `echo_me` function and the content of the web service CFC, see the example for either the `AddSOAPResponseHeader` function or the `GetSOAPRequestHeader` function.

```
<!--- Note that you might need to modify the URL in the CreateObject function
to match your server and the location of the headerservice.cfc file if it is
different than shown here. --->

<cfscript>
    ws = CreateObject("webservice",
"http://localhost/soapheaders/headerservice.cfc?WSDL");
    ws.echo_me("hello world");
    resp = getSOAPResponse(ws);
</cfscript>
<cfdump var="#resp#">
```

GetSOAPResponseHeader

Description

Returns a SOAP response header. Call this function from within code that is invoking a web service *after* making a web service request.

Returns

A SOAP response header.

Category

XML functions

History

ColdFusion MX 7: Added this function.

Function syntax

`GetSOAPResponseHeader(webservice, namespace, name [, asXML])`

See also

`AddSOAPRequestHeader`, `AddSOAPResponseHeader`, `GetSOAPRequest`, `GetSOAPRequestHeader`, `GetSOAPResponse`, `IsSOAPRequest`; "Basic web service concepts" on page 903 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`webservice`	A webservice object as returned from the `cfobject` tag or the `CreateObject` function.
`namespace`	A String that is the namespace for the header.
`name`	A String that is the name of the SOAP header.
`asXML`	If True, the header is returned as a CFML XML object. If False (default), the header is returned as a Java object.

Usage

If you specify `false` for the `asXML` parameter, ColdFusion first attempts to retrieve the header using the data type specified in the header's `xsi:type` attribute. If the `xsi:type` attribute is not available, ColdFusion attempts to retrieve the header as a string. If you specify `true` for the `asXML` parameter, ColdFusion retrieves the header as raw XML.

Used within CFML code by a consumer of a web service *after* it calls the web service with `cfinvoke`.

Example

There are two parts to this example. The first part is the web service CFC that this function (as well as the other ColdFusion SOAP functions) uses to demonstrate its interaction with a web service. To implement the web service for this function, see the example for either the `AddSOAPResponseHeader` function or the `GetSOAPRequestHeader` function.

Execute the following example to see how the `GetSOAPResponseHeader` function operates:

```
<!--- Note that you might need to modify the URL in the CreateObject function
to match your server and the location of the headerservice.cfc file if it is
different than shown here. Likewise for the cfinvoke tag at the end --->

<h3>GetSOAPResponseHeader Example</h3>
<cfscript>
```

```
 // Create the web service object
 ws = CreateObject("webservice", "http://localhost/soapheaders/headerservice.cfc?WSDL");

 // Set the username header as a string
 addSOAPRequestHeader(ws, "http://mynamespace/", "username", "tom", false);

 // Set the password header as a CFML XML object
 doc = XmlNew();
 doc.password = XmlElemNew(doc, "http://mynamespace/", "password");
 doc.password.XmlText = "My Voice is my Password";
 doc.password.XmlAttributes["xsi:type"] = "xsd:string";
 addSOAPRequestHeader(ws, "ignoredNameSpace", "ignoredName", doc);

 // Invoke the web service operation
 ret = ws.echo_me("argument");

 // Get the first header as an object (string) and as XML
 header = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader");
 XMLheader = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader", true);

 // Get the second header as an object (string) and as XML
 header2 =getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader2");
 XMLheader2 = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader2", true);
</cfscript>
<hr>
<cfoutput>
Soap Header value: #HTMLCodeFormat(header)#<br>
Soap Header XML value: #HTMLCodeFormat(XMLheader)#<br>
Soap Header 2 value: #HTMLCodeFormat(header2)#<br>
Soap Header 2 XML value: #HTMLCodeFormat(XMLheader2)#<br>
Return value: #HTMLCodeFormat(ret)#<br>
</cfoutput>
<hr>

<cfinvoke component="soapheaders.headerservice" method="echo_me" returnvariable="ret"
in_here="hi">
</cfinvoke>
<cfoutput>Cfinvoke returned: #ret#</cfoutput>
```

GetTempDirectory

Description

Gets the path of the directory that ColdFusion uses for temporary files. The directory depends on the account under which ColdFusion is running and other factors. Before using this function in an application, test to determine the directory it returns under your account.

Returns

The absolute pathname of a directory, including a trailing slash, as a string.

Category

System functions

Function syntax

```
GetTempDirectory()
```

See also

GetTempFile

History

ColdFusion MX: Changed behavior: on Windows, this function now returns the temporary directory of the embedded Java application server. On other platforms, it returns the temporary directory of the operating system.

Example

```
<h3>GetTempDirectory Example</h3>

<p>The temporary directory for this ColdFusion server is
    <cfoutput>#GetTempDirectory()#</cfoutput>.</p>
<p>We have created a temporary file called:
<cfoutput>#GetTempFile(GetTempDirectory(),"testFile")#</cfoutput></p>
```

GetTempFile

Description

Creates a temporary file in a directory whose name starts with (at most) the first three characters of *prefix*.

Returns

Name of a temporary file, as a string.

Category

System functions

Function syntax

`GetTempFile(dir, prefix)`

See also

`GetTempDirectory`

Parameters

Parameter	Description
`dir`	Directory name
`prefix`	Prefix of a temporary file to create in the `dir` directory

Example

```
<h3>GetTempFile Example</h3>
<p>The temporary directory for this ColdFusion Server is
    <cfoutput>#GetTempDirectory()#</cfoutput>.</p>
<p>We have created a temporary file called:
<cfoutput>#GetTempFile(GetTempDirectory(),"testFile")#</cfoutput></p>
```

GetTemplatePath

Description

This function is deprecated. Use the GetBaseTemplatePath function instead.

Gets the absolute path of an application's base page.

History

ColdFusion MX: Deprecated this function. It might not work, and it might cause an error, in later releases.

GetTickCount

Description

Returns the current value of an internal millisecond timer.

Returns

A string representation of the system time, in milliseconds.

Category

Date and time functions, System functions

Function syntax

```
GetTickCount()
```

Usage

This function is useful for timing CFML code segments or other page processing elements. The value of the counter has no meaning. To generate useful timing values, take the difference between the results of two `GetTickCount` calls.

Example

```
<!--- Setup timing test --->
<cfset iterationCount = 1000>
<!--- Time an empty loop with this many iterations --->
<cfset tickBegin = GetTickCount()>
<cfloop Index = i From = 1 To = #iterationCount#></cfloop>
<cfset tickEnd = GetTickCount()>
<cfset loopTime = tickEnd - tickBegin>

<!--- Report --->
<cfoutput>Loop time (#iterationCount# iterations) was: #loopTime#
     milliseconds</cfoutput>
```

GetTimeZoneInfo

Description

Gets local time zone information for the computer on which it is called, relative to Universal Time Coordinated (UTC). UTC is the mean solar time of the meridian of Greenwich, England, used as the basis for calculating standard time throughout the world.

ColdFusion stores date and time values as date-time objects: real numbers on a universal time line. It converts an object to a time zone when it formats an object; it converts a date/time value from a time zone to a real number when it parses a value.

Returns

Structure that contains these elements and keys:

- utcTotalOffset: offset of local time, in seconds, from UTC
  - A plus sign indicates a time zone west of UTC (such as a zone in North America)
  - A minus sign indicates a time zone east of UTC (such as a zone in Germany)
- utcHourOffset: offset, in hours of local time, from UTC
- utcMinuteOffset: offset, in minutes, beyond the hours offset. For North America, this is 0. For countries that are not exactly on the hour offset, the number is between 0 and 60. For example, standard time in Adelaide, Australia is offset 9 hours and 30 minutes from UTC.
- isDSTOn: True, if Daylight Savings Time (DST) is on in the host; False, otherwise

Category

Date and time functions, International functions

Function syntax

```
GetTimeZoneInfo()
```

See also

DateConvert, CreateDateTime, DatePart

Example

```
<h3>GetTimeZoneInfo Example</h3>
<!--- This example shows the use of GetTimeZoneInfo --->
<cfoutput>
The local date and time are #now()#.
</cfoutput>

<cfset info = GetTimeZoneInfo()>
<cfoutput>
<p>Total offset in seconds is #info.utcTotalOffset#.</p>
<p>Offset in hours is #info.utcHourOffset#.</p>
<p>Offset in minutes minus the offset in hours is
    #info.utcMinuteOffset#.</p>
<p>Is Daylight Savings Time in effect? #info.isDSTOn#.</p>
</cfoutput>
```

GetToken

Description

Determines whether a token of the list in the `delimiters` parameter is present in a string.

Returns

The token found at position *index* of the string, as a string. If *index* is greater than the number of tokens in the string, returns an empty string.

Category

String functions

Function syntax

`GetToken(string, index [, delimiters ])`

See also

`Left`, `Right`, `Mid`, `SpanExcluding`, `SpanIncluding`

Parameters

Parameter	Description
`string`	A string or a variable that contains one. String in which to search.
`index`	Positive integer or a variable that contains one. The position of a token.
`delimiters`	A string or a variable that contains one. A delimited list of delimiters. Elements may consist of multiple characters.
	Default list of delimiters: space character, tab character, newline character; or their codes: "chr(32)", "chr(9)", chr(10).
	Default list delimiter: comma character.

Usage

The following examples show how this function works.

Example 1

In the following example, the function call requests element number 2 from the string, using the delimiter ":;".

```
GetToken("red,blue:;red,black,tan:;red,pink,brown:;red,three", 2, ":;")
```

The output is as follows:

```
   red,black,tan
```

Example 2

```
<cfset mystring = "four,"
    & #chr(32)# & #chr(9)# & #chr(10)#
    & ",five, nine,zero:;"
    & #chr(10)#
    & "nine,ten:, eleven:;twelve:;thirteen,"
    & #chr(32)# & #chr(9)# & #chr(10)#
    & ",four">
<cfoutput>
    #mystring#<br><br>
</cfoutput>
```

The output is as follows:

```
four,
```

```
,five, nine,zero:;
nine,ten:, eleven:;twelve:;thirteen,
,four
```

The GetToken function recognizes explicit spaces, tabs, or newline characters as the parameter delimiters (To specify a space character, the code is chr(32); a tab character, chr(9); and a newline character, chr(10).)

In the example string mystring, there is:

- A forced space between the substrings "four," and ",five"
- A literal space between "five," and "nine"
- A literal space between "ten:," and "eleven,"
- A forced space between "thirteen," and ",four"

In the following call against mystring, no spaces are specified in delimiters (it is omitted), so the function uses the space character as the string delimiter:

```
<br>
<cfoutput>
    GetToken(mystring, 3) is : #GetToken(mystring, 3)#
</cfoutput><br>
```

The output of this code is as follows:

```
GetToken(mystring, 3) is : nine,zero:;
```

The function finds the third delimiter, and returns the substring just before it that is between the second and third delimiter. This substring is "nine,zero:;".

Example 3

```
<cfset mystring2 = "four,"
    &#chr(9)# & #chr(10)#
    & ",five,nine,zero:;"
    & #chr(10)#
    & "nine,ten:,eleven:;twelve:;thirteen,"
    & #chr(9)# & #chr(10)# & ",four">
<cfoutput>
    #mystring2#<br>
</cfoutput>
```

The output is as follows:

```
four,
,five,nine,zero:;
nine,ten:,eleven:;twelve:;thirteen,
,four
```

The following is a call against mystring2:

```
<cfoutput>
    GetToken(mystring2, 2) is : #GetToken(mystring2, 2)#
</cfoutput>
```

The output is as follows:

```
GetToken(mystring2, 2) is : ,five,nine,zero:;
```

The function finds the second delimiter, and returns the substring just before it that is between the first and second delimiter. This substring is ",five,nine,zero:;".

Example

```
<h3>GetToken Example</h3>
<cfif IsDefined("FORM.yourString")>
<!--- set delimiter --->
<cfif FORM.yourDelimiter is not "">
        <cfset yourDelimiter = FORM.yourDelimiter>
    <cfelse>
        <cfset yourDelimiter = " ">
</cfif>
<!--- check whether number of elements in list is greater than or
    equal to the element sought to return --->
<cfif ListLen(FORM.yourString, yourDelimiter) GTE FORM.returnElement>
    <cfoutput>
    <p>Element #FORM.ReturnElement# in #FORM.yourString#,
    delimited by "#yourDelimiter#"
    <br>is:#GetToken(FORM.yourString, FORM.returnElement, yourDelimiter)#
    </cfoutput>
...
```

GetUserRoles

Description

Retrieves the list of roles for the current user. This function returns only ColdFusion roles, not roles set for the servlet API.

Returns

The list of roles for the current user.

Category

Security functions

Function syntax

```
GetUserRoles()
```

See also

cflogin, cfloginuser, cflogout, GetAuthUser, IsUserInAnyRole, IsUserInRole, IsUserLoggedIn,
"Securing Applications" on page 312 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function.

Example

```
<cfloginuser name = "#cflogin.name#" password = "#cflogin.password#"
 roles = "#GetUserRoles()#" />
```

GetWriteableImageFormats

Description
Returns a list of image formats that ColdFusion can write on the operating system where ColdFusion is deployed.

Returns
A list of image file formats.

Category
System functions

History
ColdFusion 8: Added this function.

Function syntax
```
GetWriteableImageFormats()
```

See also
`GetReadableImageFormats`"Supported image file formats" on page 304

Usage
Use this function to determine image file compatibility on the ColdFusion server.

Example
```
<cfoutput>#GetWriteableImageFormats()#</cfoutput>
```

Hash

Description

Converts a variable-length string to a fixed-length string that can act as a "fingerprint" or unique identifier for the original string. It is not possible to convert the hash result back to the source string.

Returns

A string.

Category

Conversion functions, Security functions, String functions

Function syntax

```
Hash(string [, algorithm [, encoding ]])
```

History

ColdFusion MX 7: Added the *algorithm* and *encoding* parameters.

Parameters

Parameter	Description
string	String to hash.
algorithm	(Optional) The algorithm to use to hash the string. ColdFusion installs a cryptography library with the following algorithms: • CFMX_COMPAT: Generates a hash string identical to that generated by ColdFusion MX and ColdFusion MX 6.1 (default). • MD5: (default) Generates a 32-character, hexadecimal string, using the MD5 algorithm (The algorithm used in ColdFusion MX and prior releases). • SHA: Generates a 28-character string using the Secure Hash Standard SHA-1 algorithm specified by Nation Institute of Standards and Technology (NIST) FIPS-180-2. • SHA-256: Generates a 44-character string using the SHA-256 algorithm specified by FIPS-180-2. • SHA-384: Generates a 64-character string using the SHA-384 algorithm specified by FIPS-180-2. • SHA-512: Generates an 88-character string using the SHA-1 algorithm specified by FIPS-180-2. If you install a security provider with additional cryptography algorithms, you can also specify any of its hashing algorithms.
encoding	(Optional; to use this attribute you must also specify the *algorithm* parameter) A string specifying the encoding to use when converting the string to byte data used by the hash algorithm. Must be a character encoding name recognized by the Java runtime. The default value is the value specified by the defaultCharset entry in the neo-runtime.xml file, which is normally UTF-8. Ignored when using the CFMX_COMPAT algorithm.

Usage

The result of this function is useful for comparison and validation. For example, you can store the hash of a password in a database without exposing the password. You can check the validity of the password by hashing the entered password and comparing the result with the hashed password in the database.

ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

The `encoding` attribute is normally not required. It provides a mechanism for generating identical hash values on systems with different default encodings. ColdFusion uses a default encoding of UTF-8 unless you modify the defaultCharset entry in the neo-runtime.xml file.

Example

The following example lets you enter a password and compares the hashed password with a hash value saved in the SecureData table of the cfdocexamples database. This table has the following entries:

User ID	Password
blaw	blaw
dknob	dknob

```
<h3>Hash Example</h3>

<!--- Do the following if the form is submitted. --->
<cfif IsDefined("Form.UserID")>

    <!--- query the data base. --->
    <cfquery name = "CheckPerson" datasource = "cfdocexamples">
        SELECT PasswordHash
        FROM SecureData
        WHERE UserID = <cfqueryparam value = "#Form.userID#"
            cfsqltype = 'CF_SQL_VARCHAR'>
    </cfquery>

    <!--- Compare query PasswordHash field and the hashed form password
            and display the results. --->
    <cfoutput>
        <cfif Hash(Form.password, "SHA") is not checkperson.passwordHash>
            User ID #Form.userID# or password is not valid. Try again.
        <cfelse>
            Password is valid for User ID #Form.userID#.
        </cfif>
    </cfoutput>
</cfif>

<!--- Form for entering ID and password. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>User ID: </b>
    <input type = "text" name="UserID" ><br>
    <b>Password: </b>
    <input type = "text" name="password" ><br><br>
    <input type = "Submit" value = "Encrypt my String">
</form>
```

Hour

Description
Gets the current hour of the day.

Returns
Ordinal value of the hour, in the range 0–23.

Category
Date and time functions

Function syntax
```
Hour(date)
```

See also
DatePart, Minute, Second

Parameters

Parameter	Description
date	Date/time object, in the range 100 AD–9999 AD.

Usage
When passing a date/time value as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Example
```
<!--- This example shows the use of Hour, Minute, and Second --->
<h3>Hour Example</h3>
<cfoutput>
The time is currently #TimeFormat(Now())#.
We are in hour #Hour(Now())#, Minute #Minute(Now())#
and Second #Second(Now())# of the day.
</cfoutput>
```

HTMLCodeFormat

Description

Replaces special characters in a string with their HTML-escaped equivalents and inserts <pre> and </pre> tags at the beginning and end of the string.

Returns

HTML-escaped string *string*, enclosed in <pre> and </pre> tags. Return characters are removed; line feed characters are preserved. Characters with special meanings in HTML are converted to HTML character entities such as >.

Category

Display and formatting functions

Function syntax

```
HTMLCodeFormat(string [, version ])
```

See also

HTMLEditFormat

Parameters

Parameter	Description
string	A string or a variable that contains one.
version	HTML version to use; currently ignored. • -1: The latest implementation of HTML • 2.0: HTML 2.0 (default) • 3.2: HTML 3.2

Usage

This function converts the following characters to HTML character entities:

Text character	Encoding
<	<
>	>
&	&
"	"

This function typically increases the length of a string. This can cause unpredictable results when performing certain string functions (Left, Right, and Mid, for example) against the expanded string.

The only difference between this function and HTMLEditFormat is that HTMLEditFormat does not surround the text in an HTML pre tag.

Example

```
<!--- This example shows the effects of HTMLCodeFormat and
HTMLEditFormat. View it in your browser; then View it
using your browser's the View Source command. --->
<cfset testString="This is a test
```

```
& this is another
<This text is in angle brackets>

Previous line was blank!!!">

<cfoutput>
<h3>The text without processing</h3>
#testString#<br>
<h3>Using HTMLCodeFormat</h3>
#HTMLCodeFormat(testString)#
<h3>Using HTMLEditFormat</h3>
#HTMLEditFormat(testString)#
</cfoutput>
```

HTMLEditFormat

Description

Replaces special characters in a string with their HTML-escaped equivalents.

Returns

HTML-escaped string *string*. Return characters are removed; line feed characters are preserved. Characters with special meanings in HTML are converted to HTML character entities such as >.

Category

Display and formatting functions

Function syntax

```
HTMLEditFormat(string [, version ])
```

See also

HTMLCodeFormat, cfapplication

Parameters

Parameter	Description
string	A string or a variable that contains one.
version	HTML version to use; currently ignored. • -1: The latest implementation of HTML • 2.0: HTML 2.0 (default) • 3.2: HTML 3.2

Usage

This function converts the following characters to HTML character entities:

Text character	Encoding
<	<
>	>
&	&
"	"

This function can used to help protect ColdFusion pages that return user-provided data to the client browser from cross-site scripting attacks. However, the scriptprotect attribute of the cfapplication tag or the equivalent This.scriptProtect variable setting in Application.cfc can be preferable in most instances, because you only need to specify it once for an application.

This function typically increases the length of a string. This can cause unpredictable results when performing certain string functions (Left, Right, and Mid, for example) against the expanded string.

The only difference between this function and HTMLCodeFormat is that HTMLCodeFormat surrounds the text in an HTML pre tag.

Example

```
<!--- This example shows the effects of HTMLCodeFormat and
```

```
HTMLEditFormat. View it in your browser, then View it
using your browser's the View Source command. --->
<cfset testString="This is a test
& this is another
<This text is in angle brackets>

Previous line was blank!!!">

<cfoutput>
<h3>The text without processing</h3>
#testString#<br>
<h3>Using HTMLCodeFormat</h3>
#HTMLCodeFormat(testString)#
<h3>Using HTMLEditFormat</h3>
#HTMLEditFormat(testString)#
</cfoutput>
```

IIf

Description

Evaluates a Boolean conditional dynamic expression. Depending on whether the expression is yes or no, dynamically evaluates one of two string expressions and returns the result. This function is convenient for incorporating a `cfif` tag in-line in HTML.

For general conditional processing, see `cfif`. For error handling, see `cftry`. For more information, see the *ColdFusion Developer's Guide*.

Returns

If result is yes, returns the value of `Evaluate(string_expression1)`; otherwise, returns the value of `Evaluate(string_expression2)`.

Category

Decision functions, Dynamic evaluation functions

Function syntax

`IIf(condition, string_expression1, string_expression2)`

See also

`DE`, `Evaluate`

Parameters

Parameter	Description
`condition`	An expression that can be evaluated as a Boolean.
`string_expression1`	A string or a variable that contains one. Expression to evaluate and return if condition is yes.
`string_expression2`	A string or a variable that contains one. Expression to evaluate and return if condition is no.

Usage

The `IIf` function is a shortcut for the following construct:

```
<cfif condition>
    <cfset result = Evaluate(string_expression1)>
<cfelse>
    <cfset result = Evaluate(string_expression2)>
</cfif>
```

The expressions *string_expression1* and *string_expression2* must be string expressions, so that they are not evaluated immediately as the parameters of `IIf`. For example:

```
IIf(y is 0, DE("Error"), x/y)
```

If `y = 0`, this generates an error, because the third expression is the value of x/0 (invalid expression).

ColdFusion evaluates *string_expression1* and *string_expression2*. To return the string itself, use the `DE` function.

Note: *If you use number signs (#) in `string_expression1` or `string_expression2`, ColdFusion evaluates the part of the expression in number signs first. If you misuse the number signs, you can cause unexpected results from the IIf function. For example, if you use number signs around the whole expression in `string_expression1`, and if there is an undefined variable in `string_expression1`, the function might fail, with the error "Error Resolving Parameter."*

If a variable is undefined, ColdFusion throws an error when it processes this function. The following example shows this problem:

```
#IIf(IsDefined("Form.Deliver"), DE(Form.Deliver), DE("no"))#
```

This returns "Error resolving parameter FORM.DELIVER".

To avoid this problem, use the `DE` and `Evaluate` functions in code such as the following:

```
#IIf(IsDefined("Form.Deliver"), Evaluate(DE("Form.Deliver")), DE("no"))#
```

This returns "no"; ColdFusion does not throw an error.

In the following example, `LocalVar` is undefined; however, if you omit number signs around `LocalVar`, the code works properly:

```
<cfoutput>
    #IIf(IsDefined("LocalVar"), "LocalVar",
        DE("The variable is not defined."))#
</cfoutput>
```

The output is:

```
    The variable is not defined.
```

The number signs around `LocalVar` in the following code cause it to fail with the error message 'Error Resolving Parameter', because ColdFusion never evaluates the original condition `IsDefined("LocalVar")`.

Here is another example:

```
<cfoutput>
#IIf(IsDefined("LocalVar"), DE("#LocalVar#"), DE("The variable is not defined."))#
</cfoutput>
```

The error message would be as follows:

```
    Error resolving parameter LOCALVAR
```

The `DE` function has no effect on the evaluation of `LocalVar`, because the number signs cause it to be evaluated immediately.

Example

```
<h3>IIf Function Example</h3>
<p>IIf evaluates a condition, and does an Evaluate on string
    expression 1 or string expression 2 depending on the Boolean
    outcome <I>(yes: run expression 1; no: run expression 2)</I>.</p>
<p>The result of the expression
IIf( Hour(Now()) GTE 12,
    DE("It is afternoon or evening"),
    DE("It is morning"))
is:<br><b>
<cfoutput>
    #IIf( Hour(Now()) GTE 12,
    DE("It is afternoon or evening"),
    DE("It is morning"))#
</cfoutput>
</b>
```

ImageAddBorder

Description
Adds a rectangular border around the outside edge of a ColdFusion image.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageAddBorder(name, thickness [, color, borderType])
```

See also
cfimage, ImageDrawRect, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
thickness	Required. Thickness of the border in pixels. The default value is 1. The border is added to the outside edge of the image; the image area is increased accordingly.
color	Optional. Border color. The default border color is black. See Usage. Only valid if the borderType is not specified or if borderType = "constant".
borderType	Optional. The type of border: • zero: Sets the border color to black. • constant: Sets the border to the specified color (default). • copy: Sets sample values to copies of the nearest valid pixel. For example, pixels to the left of the valid rectangle assume the value of the valid edge pixel in the same row. Pixels both above and to the left of the valid rectangle assume the value of the upper-left pixel. • reflect: Mirrors the edges of the source image. For example, if the left edge of the valid rectangle is located at x = 10, pixel (9, y) is a copy of pixel (10, y) and pixel (6, y) is a copy of pixel (13, y). • wrap: Tiles the source image in the plane.

Usage
The thickness of the border is specified in pixels by the thickness parameter. The thickness cannot be less than 0.

For the color value, specify a hexadecimal value or supported named color; see the name list in "Valid HTML named colors" on page 305. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxx", where x = 0–9 or A–F; use two number signs or none.

Example
Example 1

```
<!--- This example shows how to create a 10-pixel-wide red border around an image with a 5-pixel-wide green border around the red border.--->
```

```
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Draw a red border around the outside edge of the image. --->
<cfset ImageAddBorder(myImage,10,"red")>
<!--- Draw a green border around the outside edge of the red border. --->
<cfset ImageAddBorder(myImage,5,"green")>
<!--- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpeg"/>
```

Example 2

```
<!--- This example shows how to create a border from the tiled image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/lori05.jpg" name="myImage">
<!--- Add a 50-pixel-wide border to the outside edge of the image that is a tiled version
of the image itself. --->
<cfset ImageAddBorder(myImage,50,"","wrap")>
<!--- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/lori05.jpg"/>
<img src="test_myImage.jpeg"/>
```

Example 3

```
<!--- This example shows how to create a 100-pixel-wide border that is a mirror of the source
image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" name="myImage">
<!--- Create the border. --->
<cfset ImageAddBorder(myImage,100,"","reflect")>
<!--- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/maxwell01.jpg"/>
<img src="test_myImage.jpeg"/>
```

Example 4

```
<!--- This example shows how to copy 100 pixels from the outer edge of the image and create
a border from it. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<cfset ImageAddBorder(myImage,100,"","copy")>
<!--- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpeg"/>
```

ImageBlur

Description
Smooths (blurs) the ColdFusion image.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageBlur(name [, blurRadius])
```

See also
ImageSharpen, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
blurRadius	Optional. The size of the blur radius.
	Value must be greater than or equal to 3 and less than or equal to 10. The default value is 3.

Usage
The blurRadius operation affects performance: as the blurRadius value increases, performance decreases.

Example
```
<!--- This example shows how to blur an image by a radius of 10. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Use the maximum blur radius to blur the image. --->
<cfset ImageBlur(myImage,10)>
<!--- Save the modified ColdFusion image to a JPEG file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpeg"/>
```

ImageClearRect

Description
Clears the specified rectangle by filling it with the background color of the current drawing surface.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageClearRect(name, x, y, width, height)`

See also
`ImageSetBackgroundColor`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`x`	Required. The x coordinate of the rectangle to clear.
`y`	Required. The y coordinate of the rectangle to clear.
`width`	Required. The width of the rectangle to clear.
`height`	Required. The height of the rectangle to clear.

Usage
Use this function in conjunction with the `ImageSetBackgroundColor` function.

Example
```
<!--- This example shows how to set the background color to green and draws four rectangles
in that color on the image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Set the background color to green. --->
<cfset ImageSetBackgroundColor(myImage,"green")>
<!--- Draw four rectangles in the background color. --->
<cfset ImageClearRect(myImage,10,25,50,50)>
<cfset ImageClearRect(myImage,100,25,50,50)>
<cfset ImageClearRect(myImage,10,100,50,50)>
<cfset ImageClearRect(myImage,100,100,50,50)>
<!--- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpeg"/>
```

ImageCopy

Description
Copies a rectangular area of an image.

Returns
A ColdFusion image for the copied area.

Category
Image functions

Function syntax
```
ImageCopy(name, x, y, width, height [, dx, dy])
```

See also
ImageNew, ImagePaste, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
x	Required. The x coordinate of the source rectangle.
y	Required. The y coordinate of the source rectangle.
width	Required. The width of the source rectangle.
height	Required. The height of the source rectangle.
dx	Optional. The x coordinate of the destination rectangle.
dy	Optional. The y coordinate of the destination rectangle.

Usage
The rectangle is specified by (*x*,*y*,*width*,*height*). The area is copied to the rectangle with the upper-left corner specified by (*dx*, *dy*).

Example
Example 1

```
<!--- This example shows how to copy a rectangular area of an image to a new image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/lori05.jpg" name="myImage">
<!--- Copy the rectangular area specified by the coordinates (25,25,50,50) in the image to
the rectangle beginning at (75,75), and return this copied rectangle as a new ColdFusion
image. --->
<cfset dupArea = ImageCopy(myImage,25,25,50,50,75,75)>
<!--- Write the result to a PNG file. --->
<cfimage source=#myImage# action="write" destination="test_myImage.png" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/lori05.jpg">
<img src="test_myImage.png">
```

Example 2

```
<!--- This example shows how to copy a rectangular area from one image and paste it over
another image. --->
<!--- Create a ColdFusion image named "myImage1" from an existing JPEG file.--->
<cfimage source="../cfdocs/images/artgallery/lori05.jpg" name="myImage1">
<!--- Create the ColdFusion image "myImage2" from a different JPEG file.
    --->
<cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" name="myImage2">
<!--- Copy a rectangular region of "myImage1" as a new image. --->
<cfset resImage = ImageCopy(myImage1,1,1,55,55)>
<!--- Paste the rectangular area on the second image. --->
<cfset ImagePaste(myImage2,resImage,50,75)>
<!--- Write the second ColdFusion image to a file. --->
<cfimage action="write" source="#myImage2#" destination="test_myImage.jpg" overwrite="yes">
<!--- Display the two source files and the new file. --->
<img src="../cfdocs/images/artgallery/lori05.jpg"/>
<img src="../cfdocs/images/artgallery/maxwell01.jpg"/>
<img src="test_myImage.jpg"/>
```

ImageCrop

Description
Crops a ColdFusion image to a specified rectangular area.

Returns
Nothing.

Category
Image functions

Function syntax
ImageCrop(name, x, y, width, height)

See also
ImageFlip, ImageResize, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
x	Required. The x origin of the crop area.
y	Required. The y origin of the crop area.
width	Required. The width of the crop area.
height	Required. The height of the crop area.

Usage
The rectangular area cannot be empty, and must be fully contained within the source image bounds.

Example
```
<!--- This example shows how to crop an image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/lori05.jpg" name="myImage">
<!--- Crop myImage to 100x100 pixels starting at the coordinates (10,10).
--->
<cfset ImageCrop(myImage,10,10,100,100)>
<!--- Write the result to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/lori05.jpg"/>
<img src="test_myImage.jpg"/>
```

ImageDrawArc

Description
Draws a circular or elliptical arc.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageDrawArc(name, x, y, width, height, startAngle, arcAngle [, filled])
```

See also
ImageDrawCubicCurve, ImageDrawOval, ImageDrawQuadraticCurve, ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
x	Required. The x coordinate of the upper-left corner of the arc.
y	Required. The y coordinate of the upper-left corner of the arc.
width	Required. The width of the arc.
height	Required. The height of the arc.
startAngle	Required. The beginning angle in degrees.
arcAngle	Required. The angular extent of the arc, relative to the start angle.
filled	Optional. Specify whether the arc is filled: • yes: The arc is filled with the specified drawing color. • no: Only the outline of the arc is drawn (default).

Usage
The resulting arc begins at startAngle and extends for arcAngle degrees. Degrees start at 0 in the three o'clock position. A positive value indicates a counter-clockwise rotation; a negative value indicates a clockwise rotation.

The center of the arc is the center of the rectangle whose origin is (x,y) and whose size is specified by the width and height parameters.

The angles are specified relative to the non-square extents of the bounding rectangle so that 45 degrees always falls on the line from the center of the ellipse to the upper-right corner of the bounding rectangle. As a result, if the bounding rectangle is noticeably longer on one axis than the other, the angles to the start and end of the arc segment are skewed farther along the longer axis of the bounds.

If the filled parameter is set to yes, the area inside the oval is filled with the current drawing color.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to specify the color and line attributes of the arc. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example

```
<!--- This example shows how to use the ImageNew function to create a blank
ColdFusion image that is 250 pixels wide and 180 pixels high. --->
<cfset myImage=ImageNew("",250,320)>
<!--- Set the drawing color for the arc to orange. --->
<cfset ImageSetDrawingColor(myImage,"orange")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Draw an enclosed orange arc starting at the coordinate (5,5). --->
<cfset ImageDrawArc(myImage,5,5,200,300,100,100,"yes")>
<!--- Display the image in a browser. --->
<cfimage action="writeToBrowser" source="#myImage#">
```

ImageDrawBeveledRect

Description
Draws a rectangle with beveled edges.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageDrawBeveledRect(name, x, y, width, height, raised [, filled])
```

See also
ImageClearRect, ImageDrawLine, ImageDrawLines, ImageDrawRect, ImageDrawRoundRect, ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke

History

ColdFusion 8: Added this function.**Parameters**

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
x	Required. The x coordinate of the rectangle.
y	Required. The y coordinate of the rectangle.
width	Required. The width of the rectangle.
height	Required. The height of the rectangle.
raised	Required. Specify whether the rectangle appears raised above the surface or sunk into the surface: • yes: The rectangle is raised. • no: The rectangle is sunk (default).
filled	Optional. Specify whether the rectangle is filled: • yes: The rectangle is filled with the specified drawing color. • no: Only the outline of the rectangle is drawn (default).

Usage
The edges of the rectangle are highlighted so that they appear beveled and lit from the upper-left corner. The colors used for the highlighting effect are determined by the current drawing color.

If the filled parameter is set to yes, the cuboid area is filled with the current drawing color.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to specify the color and line attributes of the rectangle. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to create a bevel-edged rectangle. --->
<!--- Use the ImageNew function to create a 200x200-pixel image. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Turn on antialiasing to improve image quality. --->
```

```
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the drawing color for the image to light gray. --->
<cfset ImageSetDrawingColor(myImage,"lightgray")>
<!--- Draw a 3D gray, filled, raised rectangle. --->
<cfset ImageDrawBeveledRect(myImage,50,50,100,75,"yes","yes")>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageDrawCubicCurve

Description
Draws a cubic curve.

Returns
Nothing.

Category,
Image functions

Function syntax
```
ImageDrawCubicCurve(name, ctrlx1, ctrly1, ctrlx2, ctrly2, x1, y1, x2, y2)
```

See also
ImageDrawQuadraticCurve, ImageDrawRect, ImageDrawRoundRect, ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
ctrlx1	Required. The x coordinate of the first control point of the cubic curve segment.
ctrly1	Required. The y coordinate of the first control point of the cubic curve segment.
ctrlx2	Required. The x coordinate of the second control point of the cubic curve segment.
ctrly2	Required. The y coordinate of the second control point of the cubic curve segment.
x1	Required. The x coordinate of the start point of the cubic curve segment.
y1	Required. The y coordinate of the start point of the cubic curve segment.
x2	Required. The x coordinate of the end point of the cubic curve segment.
y2	Required. The y coordinate of the end point of the cubic curve segment.

Usage
Coordinates can be integers or real numbers.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to specify the color and line attributes of the cubic curve. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to draw a curved line that looks like a stylized 7. --->
<!--- Use the ImageNew function to create a blank ColdFusion image that is 200 pixels wide
and 380 pixels high. --->
<cfset myImage=ImageNew("",200,380)>
<!--- Set the drawing color to magenta. --->
<cfset ImageSetDrawingColor(myImage,"magenta")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
```

```
<!--- Draw a curved line that starts at (380,28) and ends at (32,56) with its first control
point at (120,380) and its second control point at (5,15). --->
<cfset ImageDrawCubicCurve(myImage,120,380,5,15,380,28,32,56)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageDrawLine

Description
Draws a single line defined by two sets of x and y coordinates on a ColdFusion image.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageDrawLine(name, x1, y1, x2, y2)
```

See also
ImageDrawLines, ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
x1	Required. The x coordinate for the start point of the line.
y1	Required. The y coordinate for the start point of a line.
x2	Required. The x coordinate for the end point of the line.
y2	Required. The y coordinate for the end point of the line.

Usage
Each pair of coordinates, (x1,y1) and (x2,y2), defines a point. The start and end points cannot be the same.

This function is affected by the attributes defined in the ImageSetDrawingStroke and ImageSetDrawingColor functions. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to draw a square bisected by
    a diagonal line. --->
<!--- Use the ImageNew function to create a 100x100-pixel image. --->
<cfset myImage=ImageNew("",100,100)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Draw a diagonal line that bisects the square. --->
<cfset ImageDrawLine(myImage,0,0,100,100)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageDrawLines

Description
Draws a sequence of connected lines defined by arrays of x and y coordinates.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageDrawLines(name, xcoords, ycoords [, isPolygon, filled])
```

See also
ImageDrawBeveledRect, ImageDrawCubicCurve, ImageDrawLine, ImageDrawRect, ImageDrawRoundRect, ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
xcoords	Required. A CFML array of x coordinates.
ycoords	Required. A CFML array of y coordinates.
isPolygon	Optional. Specify whether the lines form a polygon: • yes: The lines are connected to form a polygon. • no: The lines do not form a polygon (default).
filled	Optional. Specify whether the polygon is filled: • yes: The polygon is filled with the specified drawing color. • no: Only the outline of the polygon is drawn (default).

Usage
Each pair of (x,y) coordinates defines a point.

To draw a polygon, set isPolygon to yes. The start point cannot be the same value as the end point. If isPolygon is yes, a line joining start point and the end point is drawn to complete a polygon. If isPolygon is no, line completing the polygon is not drawn.

Set the isPolygon and filled parameters to yes to draw a polygon filled with the current drawing color.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to control the color and line attributes. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to draw a zigzag line. --->
<!--- Use the ImageNew function to create a 250x250-pixel image. --->
<cfset myImage=ImageNew("",250,250)>
```

```
<!--- Set the drawing color to cyan. --->
<cfset ImageSetDrawingColor(myImage,"cyan")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Create arrays for the x and y coordinates. --->
<cfset x = ArrayNew(1)>
<cfset y = ArrayNew(1)>
<!--- Set the values for the x and y coordinates for three connected line segments: the first
segment begins at (100,50) and ends at (50,100). The second segment begins at (50, 100) and
ends at (200,100). The third segment begins at (200,100) and ends at (100,200). --->
    <cfset x[1] = "100">
    <cfset x[2] = "50">
    <cfset x[3] = "200">
    <cfset x[4] = "100">
    <cfset y[1] = "50">
    <cfset y[2] = "100">
    <cfset y[3] = "100">
    <cfset y[4] = "200">
<!--- Draw the lines on the image. --->
<cfset ImageDrawLines(myImage,x,y)>
<!--- Display the image in a browser. --->
<cfimage source=#myImage# action="writeToBrowser">
```

ImageDrawOval

Description
Draws an oval.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageDrawOval(name, x, y, width, height [, filled])
```

See also
ImageDrawArc, ImageDrawCubicCurve, ImageDrawQuadraticCurve, ImageDrawRoundRect, ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
x	Required. The x coordinate of the upper left corner of the oval to draw.
y	Required. The y coordinate of the upper left corner of the oval to draw.
width	Required. The width of the oval to draw.
height	Required. The height of the oval to draw.
filled	Optional. Specify whether the oval is filled: • yes: The oval is filled with the specified drawing color. • no: Only the outline of the oval is drawn (default).

Usage
The result is a circle or ellipse that fits within the rectangle specified by the x, y, width, and height arguments.

If the filled parameter is set to yes, the area inside the oval is filled with the current drawing color.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to specify the color and line attributes of the oval. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
Example 1

```
<!--- This example shows how to draw a green filled oval. --->
<!--- Use the ImageNew function to create a 200x110-pixel image. --->
<cfset myImage=ImageNew("",200,110)>
<!--- Set the drawing color to green. --->
<cfset ImageSetDrawingColor(myImage,"green")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
```

```
<!--- Draw a filled green oval on the image. --->
<cfset ImageDrawOval(myImage,5,5,190,100,"yes")>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```
<!--- This example shows how to draw a red circle with
    a line through it. --->
<!--- Use the ImageNew function to create a 201x201-pixel image. --->
<cfset myImage=ImageNew("",201,201)>
<!--- Set the drawing color to red. --->
<cfset ImageSetDrawingColor(myImage,"red")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the line width to 10 pixels. --->
<cfset attr=StructNew()>
<cfset attr.width = 10>
<cfset ImageSetDrawingStroke(myImage,attr)>
<!--- Draw a diagonal line starting at (40,40) and ending at (165,165) on myImage. --->
<cfset ImageDrawLine(myImage,40,40,165,165)>
<!--- Draw a circle starting at (5,5) and is 190 pixels high and 190 pixels wide. --->
<cfset ImageDrawOval(myImage,5,5,190,190)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageDrawPoint

Description
Draws a point at the specified (x,y) coordinate.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageDrawPoint(name, x, y)`

See also
`ImageDrawLine`, `ImageDrawLines`, `ImageSetAntialiasing`, `ImageSetDrawingColor`, `ImageSetDrawingStroke`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`x`	Required. The x coordinate of the point.
`y`	Required. The y coordinate of the point.

Usage
Use the `ImageSetDrawingStroke` and `ImageSetDrawingColor` functions to control the appearance of the drawing point. For example, set the `width` attribute of the `ImageSetDrawingStroke` function to 10 pixels to draw a 20-pixel-square centered at (x,y). Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to draw a large square in the middle of an image. --->
<!--- Use the ImageNew function to create a 200x200-pixel image. --->
<cfset myImage=ImageNew("",200,200)>
<!---Set the drawing color to orange. --->
<cfset ImageSetDrawingColor(myImage,"orange")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the drawing area to 10 pixels. --->
<cfset attr = StructNew()>
<cfset attr.width = 10>
<cfset ImageSetDrawingStroke(myImage,attr)>
<!--- Draw the point at (100,100). --->
<cfset ImageDrawPoint(myImage,100,100)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageDrawQuadraticCurve

Description
Draws a curved line. The curve is controlled by a single point.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageDrawQuadraticCurve(name, ctrlx1, ctrly1, ctrlx2, ctrly2, x1, y1, x2, y2)
```

See also
ImageDrawArc, ImageDrawOval, ImageDrawRoundRect,ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
ctrlx1	Required. The x coordinate of the first control point of the quadratic curve segment.
ctrly1	Required. The y coordinate of the first control point of the quadratic curve segment.
ctrlx2	Required. The x coordinate of the second control point of the quadratic curve segment.
ctrly2	Required. The y coordinate of the second control point of the quadratic curve segment.
x1	Required. The x coordinate of the start point of the quadratic curve segment.
y1	Required. The y coordinate of the start point of the quadratic curve segment.
x2	Required. The x coordinate of the end point of the quadratic curve segment.
y2	Required. The y coordinate of the end point of the quadratic curve segment.

Usage
A quadratic curve is a curve controlled by a single control point. The curve is drawn from the last point in the shape to the target x and y coordinates. Coordinates can be integers or real numbers.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to specify the color and lines of the quadratic curve. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to draw a curved line. --->
<!--- Use the ImageNew function to create a 400x400-pixel image. --->
<cfset myImage=ImageNew("",400,400)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the drawing color to green. --->
<cfset ImageSetDrawingColor(myImage,"green")>
<!--- Draw a curved line on the image. --->
```

```
<cfset ImageDrawQuadraticCurve(myImage,120,320,5,15,380,280)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageDrawRect

Description
Draws a rectangle.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageDrawRect(name, x, y, width, height [, filled])
```

See also
ImageDrawBeveledRect, ImageDrawCubicCurve, ImageDrawLine, ImageDrawLines, ImageDrawOval, ImageDrawQuadraticCurve, ImageDrawRoundRect, ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke, ImageDrawText, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
x	Required. The x coordinate of the rectangle.
y	Required. The y coordinate of the rectangle.
width	Required. The width of the rectangle.
height	Required. The height of the rectangle.
filled	Optional. Specify whether the rectangle is filled: • yes: The rectangle is filled with the specified drawing color. • no: Only the outline of the rectangle is drawn (default).

Usage
The left and right edges of the rectangle are at x and x plus width, respectively. The upper and lower edges are at y and y plus height, respectively.

Set the filled parameter to yes to fill the rectangle with the current drawing color.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to format the color and lines of the rectangle. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to draw a rectangle. --->
<!--- Use the ImageNew function to create a ColdFusion image that is 150 pixels wide and 200
pixels high. --->
<cfset myImage=ImageNew("",150,200)>
<!--- Set the drawing color for the image to yellow. --->
<cfset ImageSetDrawingColor(myImage,"yellow")>
```

```
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Draw a filled yellow rectangle on the image. --->
<cfset ImageDrawRect(myImage,25,45,100,20,"yes")>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageDrawRoundRect

Description
Draws a rectangle with rounded corners.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageDrawRoundRect(name, x, y, width, height, arcWidth, arcHeight [, filled])
```

See also
ImageDrawBeveledRect, ImageDrawCubicCurve, ImageDrawLine, ImageDrawLines, ImageDrawOval, ImageDrawQuadraticCurve, ImageDrawRect, ImageSetAntialiasing, ImageSetDrawingColor, ImageSetDrawingStroke, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
x	Required. The x coordinate of the rectangle.
y	Required. The y coordinate of the rectangle.
width	Required. The width of the rectangle.
height	Required. The height of the rectangle.
arcWidth	Required. The horizontal diameter of the arc at the four corners.
arcHeight	Required. The vertical diameter of the arc at the four corners.
filled	Optional. Specify whether the rectangle is filled: • yes: The rectangle is filled with the specified drawing color. • no: Only the outline of the rectangle is drawn (default).

Usage
The left and right edges of the rectangle are at x and x plus width, respectively. The upper and lower edges are at y and y plus height, respectively.

Set the filled parameter to yes to fill the rectangle with the current drawing color.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to control the color and line attributes of the rectangle. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
Example 1

```
<!--- This example shows how to draw a square with rounded corners. --->
```

```
<!--- Create a 200x200-pixel image. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Set the drawing color for the image to blue. --->
<cfset ImageSetDrawingColor(myImage,"blue")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Draw a blue filled square with round corners of arcWidth=10 and arcHeight=2. --->
<cfset ImageDrawRoundRect(myImage,5,5,190,190,"yes",10,2)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```
<!--- Create an image. --->
<cfset myImage = imageNew("",100,100,"argb")>
<!--- Create a text attribute collection. --->
<cfset textStruct = structNew()>
<cfset textStruct.size=16>
<cfset textStruct.style="bold">
<cfset textStruct.font="Trebuchet MS">

<cfoutput>
<cfloop from="1" to="20" index="i">
    <!--- Turn on antialiasing to improve the quality of the rendered image. --->
    <cfset ImageSetAntialiasing(myImage,"on")>
    <!--- Set the background color. --->
    <cfset ImageSetBackgroundColor(myImage,"cyan") />
    <cfset ImageClearRect(myImage,0,0,myImage.getwidth(),myImage.getheight())>
    <!--- Set the drawing color. --->
    <cfset ImageSetDrawingColor(myImage,"black") />
    <!--- Draw a rectangle with rounded corners. --->
    <cfset ImageDrawRoundRect(myImage,10,10,myImage.width-20, myImage.height-20,i,i,"yes")>
    <!--- Set the text arc value. --->
    <cfset ImageSetDrawingColor(myImage,"##cccccc")>
    <cfset ImageDrawText(myImage, "#i#",30,30,textStruct)>
    <!--- Write the image to a file. --->
    <cfset imageWrite(myImage,"#expandPath("#i#.png")#")>
    <!--- Display the image. --->
    <img src="#i#.png">
</cfloop>
</cfoutput>
```

ImageDrawText

Description
Draws a text string on a ColdFusion image with the baseline of the first character positioned at (x,y) in the image.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageDrawText(name, str, x, y [, attributeCollection])`

See also
ImageDrawArc, ImageDrawBeveledRect, ImageDrawCubicCurve, ImageDrawLine, ImageDrawLines, ImageDrawOval, ImageDrawQuadraticCurve, ImageDrawRect, ImageDrawRoundRect, ImageSetAntialiasing, ImageSetDrawingColor, ImageTranslateDrawingAxis, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
str	Required. The text string to draw.
x	Required. The x coordinate for the start point of the string.
y	Required. The y coordinate for the start point of the string.
attributeCollection	Optional. The structure used to specify the text characteristics. See the Usage section.

Usage
You must specify all the optional key-value pairs in an attributeCollection structure. To specify the text color, use the ImageSetDrawingColor function.

attributeCollection

Element	Description
font	The name of the font used to draw the text string. If you do not specify the font attribute, the text is drawn in the default system font.
size	The font size for the text string. The default value is 10 points.
style	The style to apply to the font: • bold • italic • boldItalic • plain (default)

Element	Description
strikethrough	Specify whether strikethrough is applied to the text image: • yes: For vertical text, strikethrough is applied to each character individually. • no (default)
underline	Specify whether underline is applied to the text image: • yes: For vertical text, underline is applied to each character individually. • no (default)

Example

Example 1

```
<!--- This example shows how to create a text string image. --->
<!--- Use the ImageNew function to create a 200x100-pixel image. --->
<cfset myImage=ImageNew("",200,100)>
<!--- Set the drawing color to green. --->
<cfset ImageSetDrawingColor(myImage,"green")>
<!--- Specify the text string and the start point for the text. --->
<cfset ImageDrawText(myImage,"It's not easy being green.",10,50)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```
<!--- This example shows how to draw three text strings with different text attributes. --->
<!--- Use the ImageNew function to create a 400x400-pixel image. --->
<cfset myImage=ImageNew("",400,400)>
<!--- Set the text attributes. --->
<cfset attr = StructNew()>
<cfset attr.underline = "yes">
<cfset attr.size = 25>
<cfset attr.style = "bold">
<cfset ImageSetDrawingColor(myImage,"yellow")>
<!--- Draw the text string "ColdFusion Rocks!" starting at (100,150). --->
<cfset ImageDrawText(myImage,"ColdFusion Rocks!",100,150,attr)>
<!--- Set new text attributes. --->
<cfset attr=StructNew()>
<cfset attr.size = 18>
<cfset attr.strikethrough = "yes">
<cfset attr.style = "bolditalic">
<cfset ImageSetDrawingColor(myImage,"red")>
<!--- Draw the text string "Powered by ColdFusion" starting at (100,200).
    --->
<cfset ImageDrawText(myImage,"Powered by ColdFusion",110,200,attr)>
<!--- Set new text attributes. --->
<cfset attr = StructNew()>
<cfset attr.font="Arial">
<cfset attr.style="italic">
<cfset attr.size=15>
<cfset ImageSetDrawingColor(myImage,"white")>
<!--- Draw the text string "Coming in 2007" starting at (150,250). --->
<cfset ImageDrawText(myImage,"We've arrived",150,250,attr)>
<!--- Display the text image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageFlip

Description
Flips an image across an axis.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageFlip(name [, transpose])`

See also
ImageBlur, ImageClearRect, ImageCrop, ImageNegative, ImageNew, ImageOverlay, ImagePaste, ImageResize, ImageRotate, ImageScaleToFit, ImageSetAntialiasing, ImageSharpen, ImageShear, ImageTranslate, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
transpose	Optional. Transpose the image: • vertical: Flip an image across an imaginary horizontal line that runs through the center of the image (default). • horizontal: Flip an image across an imaginary vertical line that runs through the center of the image. • diagonal: Flip an image across its main diagonal that runs from the upper-left to the lower-right corner. • antidiagonal: Flip an image across its main diagonal that runs from the upper-right to the lower-left corner. • ("90|180|270"): Rotate an image clockwise by 90, 180, or 270 degrees.

Usage
If you do not specify the `transpose` parameter for the `ImageFlip` function, the image is transposed on a vertical axis, creating an image that is an upside-down version of the source. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example
Example 1

```
<!--- This example shows how to rotate an image by 270 degrees. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Rotate the image by 270 degrees. --->
<cfset ImageFlip(myImage,"270")>
<!--- Display the modified image in a browser. --->
```

```
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```
<!--- This example shows how to flip an image on a vertical axis. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Flip the image so that it is upside down. --->
<cfset ImageFlip(myImage,"vertical")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 3

```
<!--- This example shows how to flip an image on a horizontal axis. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Flip the image so that it is a mirror image of the source. --->
<cfset ImageFlip(myImage,"horizontal")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 4

```
<!--- This example shows how to flip an image on a diagonal axis. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Flip the image on a diagonal axis. --->
<cfset ImageFlip(myImage,"diagonal")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 5

```
<!--- This example shows how to flip an image on an antidiagonal axis. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Flip the image on a antidiagonal axis. --->
<cfset ImageFlip(myImage,"antidiagonal")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageGetBlob

Description

Retrieves the bytes of the underlying image. The bytes are in the same image format as the source image.

Returns

The bytes of the underlying image of a BLOB.

Category

Image functions

Function syntax

`ImageGetBlob(source)`

See also

cfimage, ImageGetBufferedImage, ImageGetEXIFTag, ImageGetHeight, ImageGetIPTCTag, ImageGetWidth, ImageInfo, IsImage

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
source	Required. The ColdFusion image on which this operation is performed.

Usage

Use this function to insert ColdFusion images into BLOB columns of databases.

If you do not specify a source image, an "unknown source image format" error is generated.**Example**

Example 1

```
<!--- This example shows how to add a ColdFusion image to a database. --->
<!--- Create ColdFusion image from a an existing JPEG file. --->
<cfimage source="aiden01.jpg" name="myImage">
<!--- Use the cfquery tag to insert the ColdFusion image as a BLOB in the database. --->
<cfquery name="InsertBlobImage" datasource="myBlobData" >
INSERT into EMPLOYEES (FirstName,LastName,Photo)
VALUES ('Aiden','Quinn',<cfqueryparam value="#ImageGetBlob(myImage)#"
cfsqltype='cf_sql_blob'>)
</cfquery>
```

Example 2

The following example shows how to use the `ImageNew` function to generate thumbnail images in JPEG format from BLOB data retrieved from a database:

```
<!--- Use the cfquery tag to retrieve all employee photos and employee IDs from a database.
--->
<cfquery name= "GetBLOBs" datasource= "myBlobData">
SELECT EMLPOYEEID, PHOTO FROM Employees
</cfquery>
<!--- Use the ImageNew function to create a ColdFusion images from the BLOB data that was
retrieved from the database. --->
<cfset myImage = ImageNew(#GetBLOBs.PHOTO#)>
```

```
<!--- Create thumbnail versions of the images by resizing them to a 100x100-pixel image,
while maintaining the aspect ratio of the
    source image. --->
<cfset ImageScaleToFit(myImage,100,"")>
<!--- Convert the images to JPEG format and save them to files in the thumbnails subdirectory,
using the employee ID as the filename. --->
<cfimage source="#myImage#" action-"write"
destination="images/thumbnails/#GetBLOBs.EMPLOYEID#.jpg">
```

ImageGetBufferedImage

Description

Returns the `java.awt.BufferedImage` object underlying the current ColdFusion image.

Returns

The `java.awt.BufferedImage` object.

Category

Image functions

Function syntax

`ImageGetBufferedImage(`*name*`)`

See also

`cfimage`, `ImageGetBlob`, `ImageGetEXIFTag`, `ImageGetHeight`, `ImageGetIPTCTag`, `ImageGetWidth`, `ImageInfo`, `IsImage`, `IsImageFile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.

Usage

Use this function to return an image object that can be used with other Java Abstract Windowing Toolkit (AWT) objects embedded in the page.

Example

```
<!--- This example shows how to create a ColdFusion image, modify it, and retrieve the width
of the buffered image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul05.jpg" name="myImage">
<!--- Blur the image by an order of 10. --->
<cfset ImageBlur(myImage,10)>
<!--- Get the blurred image from the buffer and set it to variable x. --->
<cfset x = ImageGetBufferedImage(myImage)>
<!--- Return the width of the buffered image. --->
<cfoutput>#x.getWidth()#
</cfoutput>
```

ImageGetEXIFMetadata

Description
Retrieves the Exchangeable Image File Format (EXIF) headers in an image as a CFML structure.

Returns
A structure with the EXIF header values.

Category
Image functions

Function syntax
`ImageGetEXIFMetadata(name)`

See also
`cfimage`, `ImageGetEXIFTag`, `ImageGetBlob`, `ImageGetBufferedImage`, `ImageGetHeight`, `ImageGetIPTCTag`, `ImageGetWidth`, `ImageInfo`, `IsImage`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.

Usage
The EXIF is a standard for storing interchange information in image files, especially those using JPEG compression. Most digital cameras use the EXIF format.

EXIF metadata includes information pertaining to the creation of the image, such as the creation date, the software used to create the image, the aperture, the make and model, and the resolution of the image.

The result of the ImageGetEXIFMetadata function is cached in the ColdFusion image to optimize performance.

The `ImageGetEXIFMetadata` function applies only to JPEG images. If you try to retrieve metadata for Base64, BLOB, or other types of images, ColdFusion generates errors.

Example
```
<!--- This example shows how to retrieve the EXIF header information from a
JPEG file. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="images\paul05.jpg" name="myImage">
<!--- Retrieve the metadata associated with the image. --->
<cfset data =ImageGetEXIFMetadata(myImage)>
<!--- Display the ColdFusion image parameters. --->
<cfdump var="#myImage#">
<!--- Display the EXIF header information associated with the image
(creation date, software, and so on). --->
<cfdump var="#data#">
```

ImageGetEXIFTag

Description
Retrieves the specified EXIF tag in an image.

Returns
The value of the specified EXIF tag.

Category
Image functions

Function syntax
`ImageGetEXIFTag(name, tagName)`

See also
cfimage, ImageGetBlob, ImageGetBufferedImage, ImageGetHeight, ImageGetIPTCTag, ImageGetWidth, ImageInfo, IsImage, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
tagName	Required. The EXIF tag name to be returned.

Usage
The ImageGetEXIFTag function applies only to JPEG images. If you try to retrieve metadata for Base64, BLOB, or other types of images, ColdFusion generates errors.

Example
```
<!--- This example shows how to retrieve one element from the EXIF information associated
with an image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul05.jpg" name="myImage">
<!--- Retrieve the name of the software application used to create the original image. --->
<cfset data = ImageGetEXIFTag(myImage,"software")>
<!--- Display the name of the software. --->
<cfdump var="#data#">
```

ImageGetHeight

Description
Retrieves the height of the ColdFusion image in pixels.

Returns
The height of the specified ColdFusion image in pixels.

Category
Image functions

Function syntax
`ImageGetHeight(name)`

See also
cfimage, ImageGetBlob, ImageGetBufferedImage, ImageGetEXIFTag, ImageGetIPTCTag, ImageGetWidth, ImageInfo, IsImage, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.

Usage
Use this function to retrieve the height of a ColdFusion image.

Example
```
<!--- This example shows how to retrieve the height of an image. --->
<!--- Create a ColdFusion image from a JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Retrieve the height of the image. --->
<cfset height=#ImageGetHeight(myImage)#>
<!--- Display the height of the image. --->
<cfdump var="#height#">
```

ImageGetIPTCMetadata

Description

Retrieves the International Press Telecommunications Council (IPTC)headers in a ColdFusion image as a structure. The IPTC metadata contains text that describes the image that is stored with it. IPTC metadata includes, but is not limited to, caption, keywords, credit, copyright, object name, created date, byline, headline, and source.

Returns

A structure containing IPTC header values.

Category

Image functions

Function syntax

`ImageGetIPTCMetadata(`*name*`)`

See also

`cfimage`, `ImageGetBlob`, `ImageGetBufferedImage`, `ImageGetEXIFMetadata`, `ImageGetEXIFTag`, `ImageGetHeight`, z, `ImageGetWidth`, `ImageInfo`, `IsImage`, `IsImageFile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.

Usage

The IPTC metadata contains text that describes the image that is stored with it. IPTC metadata includes, but is not limited to, caption, keywords, credit, copyright, object name, created date, byline, headline, and source.

The result of the `ImageGetIPTCMetadata` function is cached in the ColdFusion image to optimize performance.

The `ImageGetIPTCMetada` function applies only to JPEG images. If you try to retrieve metadata for Base64, BLOB, or other types of images, ColdFusion generates errors.

Example

```
<!--- This example shows how to retrieve the IPTC header information for a
JPEG file. --->
<!--- Create a ColdFusion image from a JPEG file. --->
<cfimage source="images\aiden01.jpg" name="myImage">
<!--- Retrieve the IPTC header information saved with the image, such as
copyright, caption, and headline. --->
<cfset data = ImageGetIPTCMetadata(myImage)>
<!--- Display the parameter values for the ColdFusion image. --->
<cfdump var="#myImage#">
<!--- Display the IPTC header information. --->
<cfdump var=#data#>
```

ImageGetIPTCTag

Description
Retrieves the value of the IPTC tag for a ColdFusion image.

Returns
The value of the IPTC tag.

Category
Image functions

Function syntax
`ImageGetIPTCTag(name, tagName)`

See also
cfimage, ImageGetBlob, ImageGetBufferedImage, ImageGetEXIFMetadata, ImageGetEXIFTag, ImageGetHeight, ImageGetIPTCMetadata, ImageGetWidth, ImageInfo, IsImage, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
tagName	Required. The IPTC tag name whose value is returned.

Usage
The ImageGetIPTCTag function applies only to JPEG images. If you try to retrieve metadata for Base64, BLOB, or other image types, ColdFusion generates errors.

Example
```
<!--- This example shows how to retrieve the caption for a JPEG file. --->
<!--- Create a ColdFusion image from a JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul05.jpg" name="myImage" action="read">
<!--- Retrieve the camera make used to take the original picture. --->
<cfset cameraMake=ImageGetIPTCTag(myImage,"make")>
<cfdump var="#cameraMake#>"
```

ImageGetWidth

Description

Retrieves the width of the specified ColdFusion image.

Returns

An integer that represents the width of the ColdFusion image in pixels.

Category

Image functions

Function syntax

`ImageGetWidth(name)`

See also

`cfimage`, `ImageGetBlob`, `ImageGetBufferedImage`, `ImageGetEXIFTag`, `ImageGetHeight`, `ImageGetIPTCTag`, `ImageInfo`, `IsImage`, `IsImageFile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.

Example

```
<!--- This example shows how to retrieve the width of an image. --->
<!--- Create a ColdFusion image from an existing JPEG file.--->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Get the width of the image. --->
<cfset width=#ImageGetWidth(myImage)#>
<!--- Display the width of the image in pixels. --->
<cfdump var=#width#>
```

ImageGrayscale

Description
Converts a ColdFusion image to grayscale.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageGrayscale(name)`

See also
`ImageBlur`, `ImageNegative`, `ImageSetAntialiasing`, `ImageSharpen`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.

Usage
Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to change a color image to grayscale. --->
<!--- Create a ColdFusion image from an existing color image. --->
<cfimage source="../cfdocs/images/artgallery/jeff04.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Change the image to grayscale. --->
<cfset ImageGrayscale(myImage)>
<!--- Save the grayscale image to a JPEG file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpg" overwrite="yes">
<!--- Display the source image and the grayscale image. --->
<img src="../cfdocs/images/artgallery/jeff04.jpg"/>
<img src="test_myImage.jpg"/>
```

ImageInfo

Description
Returns a structure that contains information about the image, such as height, width, color model, size, and filename.

Returns
A structure that contains information for image parameters.

Category
Image functions

Function syntax
`ImageInfo(name)`

See also
`cfimage`, `ImageGetHeight`, `ImageGetWidth`, `IsImage`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.

Usage
Use this function to determine whether images are compatible. For example, to use the `ImageOverlay` function to overlay two images, both images must have the same color model.

Example
```
<!--- This example shows how to retrieve information associated with the image. --->
<!--- Create a ColdFusion image from a JPEG file.--->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Retrieve the information associated with the image. --->
<cfset info=ImageInfo(myImage)>
<cfdump var="#info#"></cfdump>
<p>height = <cfoutput>#info.height#</cfoutput>
<p>width = <cfoutput>#info.width#</cfoutput>
<p>source = <cfoutput>#info.source#"</cfoutput>
<p>pixel size = <cfoutput>#info.colormodel.pixel_size#</cfoutput>
<p>transparency = <cfoutput>#info.colormodel.transparency#</cfoutput>
```

ImageNegative

Description
Inverts the pixel values of a ColdFusion image.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageNegative(name)`

See also
`ImageBlur`, `ImageGrayscale`, `ImageSetAntialiasing`, `ImageSharpen`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.

Usage
The resulting image has the same dimensions of the source image, but not necessarily the same number bytes. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to create a negative version of an image. --->
<!--- Create a ColdFusion image from and existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Create a negative version of the image. --->
<cfset ImageNegative(myImage)>
<!--- Save the modified image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpg" overwrite="yes">
<!--- Display the source image and the negative image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpg"/>
```

ImageNew

Description
Creates a ColdFusion image.

Returns
A ColdFusion image.

Category
Image functions

Function syntax
`ImageNew([source, width, height, imageType, canvasColor])`

See also
`cfimage`, `ImageCopy`, `ImageRead`, `ImageReadBase64`, `ImageSetDrawingColor`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`source`	Optional. The source image pathname, URL, a ColdFusion variable that is read into the new ColdFusion image, or a Java buffered image.
`width`	Optional. The width of the new ColdFusion image. Valid when the height is specified and the source is not.
`height`	Optional. The height of the new ColdFusion image. Valid when the width is specified and the source is not.
`imageType`	Optional. The type of the ColdFusion image to create: • `rgb` • `argb` • `grayscale` Valid only when width and height are specified.
`canvasColor`	Optional. Color of the image canvas: • Hexadecimal value of RGB color. For example, specify the color white as `##FFFFFF` or `FFFFFF`. • String value of color (for example, `"black"`, `"red"`, `"green"`). The default value of the drawing color is `"black"`. • List of three numbers for (R,G,B) values. Each value must be in the range 0—255.

Usage
You can pass the `ImageNew` function any of the following parameters:

- Absolute or relative pathname: The image file located at the specified pathname on a disk is read and returned as a ColdFusion image.
- URL: The image from the specified URL is read and returned as a ColdFusion image.
- Width and height (`imageType` is optional): A blank ColdFusion image with the specified attributes is returned.
- ColdFusion image variable: An image variable in memory; for example, `#myImage#`.
- A BLOB from a database that contains image data.

- A byte array that contains Base64 image data.
- A Java buffered image.

ColdFusion generates an error when the passed attributes cannot create a valid ColdFusion image.

The `ImageNew` function and the `cfimage read` action support the SQL Server Image Column data type.

To read Base64 images, use the `ImageReadBase64` function.

If the color value is a string, specify a supported named color; see the name list in "Valid HTML named colors" on page 305. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxx", where x = 0–9 or A–F; use two number signs or none.

Note: *If you specify the ARGB image type, the image is white; however, if you specify RGB or grayscale, the image is black. To create blank images consistently, use the* `canvasColor` *parameter.*

Example

```
Example 1
<!--- Use the ImageNew function to create a 200x200-pixel image in ARGB format. --->
<cfset myImage = ImageNew("",200,200,"argb")>
<cfimage action="writeTobrowser" source="#myImage#">
```

Example 2

```
<!--- This example shows how to create a ColdFusion image from a BLOB in a database. --->
<cfquery
name="GetBLOBs" datasource="myblobdata">
SELECT LastName,Photo
FROM Employees
</cfquery>
<cfset i = 0>
<table border=1>
    <cfoutput query="GetBLOBS">
        <tr>
            <td>
            #LastName#
            </td>
            <td>
            <cfset i = i+1>
            <cfset myImage=ImageNew("#GetBLOBS.Photo#")>
            <cfset ImageWrite(myImage,"photo#i#.png")>
            </td>
        </tr>
    </cfoutput>
</table>
```

Example 3

```
<!--- This example shows how to create a ColdFusion image from a URL. --->
<cfset myImage = ImageNew("http://www.google.com/images/logo_sm.gif")>
<cfset ImageWrite(myImage,"google_via_imagenew.png")>
<img src="google_via_imagenew.png">
```

Example 4

```
<!--- This example shows how to use the cffile tag to convert an image file to binary format
and pass it as a variable to the ImageNew function. --->
<!---Use the cffile tag to read an image file, convert it to binary format, and write the
result to a variable. --->
<cffile action = "readBinary" file = apple.jpg"
    variable = "aBinaryObj">
```

```
<!--- Use the ImageNew function to create a ColdFusion image from the variable. --->
<cfset myImage = ImageNew(aBinaryObj)>
```

Example 5

```
<!--- This example shows how to use the cffile tag to write a ColdFusion image to a file. --->
<!--- Use the ImageNew function to create a ColdFusion image from a JPEG file. --->
<cfset myImage = ImageNew("../cfdocs/images/artgallery/aiden01.jpg")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Resize the image. --->
<cfset ImageResize(myImage,"50%","")>
<!--- Pass the image object to the cffile tag and write the result to a file on the local
drive. --->
<cffile file="#myImage#" action="write" output="c:\test_myImage.jpg">
<cfimage action="writeToBrowser" source="#myImage#">
```

Example 6

```
<!--- This example uses cfscript to pass a Java buffered image to the ImageNew function. --->
<cfscript>
    bufferedImage = createObject("java", "java.awt.image.BufferedImage");
    bufferedImage.init(JavaCast("int", 100), JavaCast("int", 100),
BufferedImage.TYPE_4BYTE_ABGR);
    myImage = imageNew(bufferedImage);
</cfscript>
```

ImageOverlay

Description
Reads two source ColdFusion images and overlays the second source image on the first source image.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageOverlay(source1, source2)`

See also
`ImageCopy`, `ImagePaste`, `ImageSetAntialiasing`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
source1	Required. The ColdFusion image that is the bottom layer in the ColdFusion image.
source2	Required. The ColdFusion image that is the top layer (overlaid on the source1 image) in the ColdFusion image.

Usage
The destination image always has the same bounding rectangle as the first source image and the same image type as the two sources. If the two source images do not intersect, the destination image is the same as the first source image.

The two source images must have the same color models. For example, you can overlay an RGB image over another RGB image, but you cannot overlay an RGB image on a grayscale image. To verify the color model of an image, use the `ImageInfo` function.

Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to overlay a smaller image on a
larger image. --->
<!--- Create a ColdFusion image from an existing JPEG file and enlarge it by 150%. This image
is displayed in the background. --->
<cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" name="myImage" action="resize"
width="150%" height="150%">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Create a ColdFusion image from an existing JPEG file. This image is overlaid on the
background image. --->
<cfimage source="../cfdocs/images/artgallery/viata05.jpg" name="topImage">
<!--- Overlay the top image on the background image. --->
<cfset ImageOverlay(myImage,topImage)>
<!--- Display the combined image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImagePaste

Description

Takes two images and an (x,y) coordinate and draws the second image over the first image with the upper-left corner at coordinate (x,y).

Returns

A ColdFusion image.

Category

Image functions

Function syntax

`ImagePaste(image1, image2, x, y)`

See also

`ImageCopy`, `ImageOverlay`, `ImageSetAntialiasing`, `IsImageFile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`image1`	Required. The bottom ColdFusion image.
`Image2`	Required. The ColdFusion image that is pasted on top of image1.
`x`	Required. The x coordinate where the upper-left corner of image2 is pasted.
`y`	Required. The *y* coordinate where the upper-left corner of image2 is pasted.

Usage

Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example

```
<!--- This example shows how to copy a small rectangular area of one image and paste it over
a larger image. --->
<!--- Create a ColdFusion image from an existing JPEG file and name it "myImage1". --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage1">
<!--- Create a ColdFusion image from an existing JPEG file and name it "myImage2". --->
<cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" name="myImage2">
<!--- Copy a rectangular region of myImage1. --->
<cfset resImage = ImageCopy(myImage1,1,1,50,50)>
<!--- Paste the rectangular area over myImage2. --->
<cfset ImagePaste(myImage2,resImage,100,100)>
<!--- Write the second ColdFusion image to result.jpg. --->
<cfimage source="#myImage2#" action="write" destination="test_myImage.jpg" overwrite="yes">
<!--- Display the two source images and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="../cfdocs/images/artgallery/maxwell01.jpg"/>
<img src="test_myImage.jpg"/>
```

ImageRead

Description
Reads the source pathname or URL and creates a ColdFusion image.

Returns
A ColdFusion image.

Category
Image functions

Function syntax
`ImageRead(path)`

History
ColdFusion 8: Added this function.

See also
`cfimage`, `ImageNew`, `ImageReadBase64`, `ImageWrite`, `IsImageFile`

Parameters

Parameter	Description
`path`	Required. Pathname or URL of the source image.

Usage
The `ImageRead` function performs the same operation as the `cfimage read` action. However, you cannot use the `cfimage` tag to read and create a ColdFusion image variable in the `cfscript` tag. Use the `ImageRead` function within the `cfscript` tag to read ColdFusion images.

The following example reads the image file aiden01.jpg into a variable called myImage and displays the image in the browser:

```
<cfset myImage=ImageRead("../cfdocs/images/artgallery/aiden01.jpg")>
<cfimage action="writeToBrowser" source="#myImage#">
```

For a list of valid image formats, see"Supported image file formats" on page 304. To retrieve a list of readable formats on the server where the ColdFusion application is deployed, use the `GetReadableImageFormats` function.

Example
```
<!--- This example shows how to create a script that reads an image from a URL. --->
<cfscript>
 myImage=ImageRead("http://www.google.com/images/logo.gif");
 ImageWrite(myImage,"google-logo.gif");
</cfscript>
<p>This image has been downloaded by ColdFusion:</p>
<img src="google-logo.gif">
<p>This is the original image:</p>
<img src="http://www.google.com/images/logo.gif"/>
```

ImageReadBase64

Description
Creates a ColdFusion image from a Base64 string.

Returns
A ColdFusion image.

Category
Image functions

Function syntax
`ImageReadBase64(string)`

See also
`ImageNew`, `ImageRead`, `ImageWrite`, `ImageWriteBase64`, `BinaryDecode`, `BinaryEncode`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`string`	Required. The ColdFusion variable or Base64 string.

Usage
Base64 is a way to describe binary data as a printable string of characters. The `ImageReadBase64` function takes Base64 strings as input and creates images from the strings.

The strings can be with or without the headers used to pass Base64 images to an HTML `<img src>` tag.

Use this function to convert any Base64 string to a ColdFusion image. Some databases store images as Base64 strings instead of BLOB data. You can query the database and use the `ImageReadBase64` function to convert the string into a ColdFusion image. This eliminates the intermediary step of converting images with the `BinaryEncode` and `BinaryDecode` functions.

Really Simple Syndication (RSS) feeds transfer images in the form of embedded Base64 strings in the XML file. Use the `ImageReadBase64` function to read these images in ColdFusion.

Example
Example 1

```
<!--- This example shows how to read a Base64 string with headers. --->

<cfset myImage =
ImageReadBase64(".............")>
<cfimage source="#myImage#" destination="test_my64.jpeg" action="write">
```

Example 2

```
<!--- This example shows how to read a Base64 string without headers. --->

<cfset myImage = ImageReadBase64("/9j/4AAQSkZJRgABAQA.............")>
<cfimage source="#myImage#" destination="test_my64.jpeg" action="write">
```

ImageResize

Description
Resizes a ColdFusion image.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageResize(name, width, height [, interpolation, blurFactor])`

See also
`cfimage`, `ImageSetAntialiasing`, `ImageScaleToFit`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`width`	Required. New width of the ColdFusion image.
	If this value is blank, the width is calculated proportionately to the height.
`height`	Required. New height of the ColdFusion image.
	If this value is blank, the height is calculated proportionately to the width.

Parameter	Description
interpolation	Optional. The interpolation method for resampling. You can specify a specific interpolation algorithm by name (for example, hamming), by image quality (for example, mediumQuality), or by performance (for example, highestPerformance). Valid values are: • highestQuality (default) • highQuality • mediumQuality • highestPerformance • highPerformance • mediumPerformance • nearest • bilinear • bicubic • bessel • blackman • hamming • hanning • hermite • lanczos • mitchell • quadratic See "Interpolation algorithms" on page 928 in the Usage section for more information.
blurFactor	Optional. The blur factor used for resampling. The higher the blur factor, the more blurred the image (also, the longer it takes to resize the image). Valid values are 1—10.

Usage

You can use this function to enlarge an image or create a thumbnail image.

To specify the height or width in pixels, enter the integer, for example, 100. To specify the height or width as a percentage, enter the percentage followed by the percent symbol, for example, 50%.

To resize an image by one dimension (for example, height), specify the height and leave width value blank (""). ColdFusion calculates the width proportionally to the height.

Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Interpolation algorithms

Interpolation algorithms let you fine-tune how images are resampled. Each algorithm balances image quality against performance: in general, the higher the image quality, the slower the performance. Quality and peformance differ based on image type and the size of the source file. The following table describes the algorithms and their named equivalents based on average test results:

Value	Named equivalents	Description
highestQuality (default)	lanczos	Highest image quality with low performance
highQuality, mediumPerformance	mitchell, quadratic	Good image quality with slightly higher performance

Value	Named equivalents	Description
mediumQuality, highPerformance	hamming, hanning, hermite	Medium quality image with medium performance
	blackman, bessel	Slightly distored image quality with high performance
highestPerformance	nearest, bicubic, bilinear	Poor image quality with highest performance

Example

```
<!--- This example shows how to resize an image to 50% of original size and resize it
proportionately to the new width. Notice that the height is blank.--->
<cfset myImage=ImageNew("http://www.google.com/images/logo_sm.gif")>
<cfset ImageResize(myImage,"50%","","blackman",2)>
<!--- Save the modified image to a file called "test_myImage.jpeg" and display the image in
a browser. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the thumbnail image. --->
<img src="http://www.google.com/images/logo_sm.gif"/>
<img src="test_myImage.jpeg"/>
```

ImageRotate

Description
Rotates a ColdFusion image at a specified point by a specified angle.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageRotate(name, angle [, x, y, interpolation])
```

See also
cfimage, ImageFlip, ImageSetAntialiasing, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
angle	Required. The rotation angle in degrees.
x	Optional. The x coordinate for the point of rotation. The default value is 2.
y	Optional. The y coordinate for the point of rotation. The default value is 2.
interpolation	Optional. Type of interpolation: • nearest: Applies the nearest neighbor method of interpolation. Image quality is lower than the other interpolation methods, but processing is fastest (default). • bilinear: Applies the bilinear method of interpolation. The quality of the image is less pixelated than the default, but processing is slower. • bicubic: Applies the bicubic method of interpolation. Generally, the quality of image is highest with this method and processing is slowest.

Usage
You must specify both the x and the y coordinates or neither. If you do not specify the x and y coordinates, the point of rotation is the center of the image, which is the default position. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
Example 1

```
<!--- This example shows how to rotate an image by 10 degrees. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfset myImage=ImageNew("../cfdocs/images/artgallery/jeff05.jpg")>
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Rotate the image by 10 degrees. --->
<cfset ImageRotate(myImage,10)>
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```
<!--- This example shows how to rotate an image by 10 degrees and change the interpolation
to bicubic for higher resolution. The image is rotated at the (10,90) coordinates. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<cfset ImageRotate(myImage,10,10,90,"bicubic")>
<cfimage source="#myImage#" destination="testMyImage.jpeg" action="write" overwrite="yes">
<img src="../cfdocs/images/artgallery/jeff05.jpg">
<img src="testMyImage.jpeg">
```

ImageRotateDrawingAxis

Description
Rotates all subsequent drawing on a ColdFusion image at a specified point by a specified angle.

Returns
A ColdFusion image.

Category
Image functions

Function syntax
```
ImageRotateDrawingAxis(name, angle [, x, y])
```

See also
ImageRotate, ImageSetAntialiasing, ImageSetBackgroundColor, ImageSetDrawingColor, ImageSetDrawingStroke, ImageSetDrawingTransparency, ImageShearDrawingAxis, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
angle	Required. The rotation angle in degrees.
x	Optional. The x coordinate for the point of rotation. The default value is 0.
y	Optional. The y coordinate for the point of rotation. The default value is 0.

Usage
The default position of the origin is 0,0. To revert to the original drawing axis, call the same (x,y) parameters with the negative of the original angle. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- This example shows to create an image with three shapes drawn on the same axis. --->
<!--- Use ImageNew to create a 300x300-pixel image. --->
<cfset myImage=ImageNew("",300,300)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the drawing axis to 30 degrees and the point of rotation at (10,10). --->
<cfset ImageRotateDrawingAxis(myImage,30,10,10)>
<!--- Set the drawing color to blue. --->
<cfset ImageSetDrawingColor(myImage,"blue")>
<!--- Draw three shapes with the same color and drawing axis. --->
<cfset ImageDrawRect(myImage,150,10,10,150,"yes")>
<cfset ImageDrawOval(myImage,200,40,45,65,"yes")>
<cfset ImageDrawRect(myImage,275,10,10,150,"yes")>
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageScaleToFit

Description
Creates a resized image with the aspect ratio maintained.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageScaleToFit(name, fitWidth, fitHeight [, interpolation , blurFactor])
```

See also
cfimage, ImageResize, ImageSetAntialiasing, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
fitWidth	Required. The width of the bounding box in pixels. You can specify an integer, or an empty string ("") if the fitHeight is specified. See the Usage section for more information.
fitHeight	Required. The height of the bounding box in pixels. You can specify an integer, or an empty string ("") if the fitWidth is specified. See the Usage section for more information.

Parameter	Description
interpolation	Optional. The interpolation method for resampling. You can specify a specific interpolation algorithm by name (for example, hamming), by image quality (for example, mediumQuality), or by performance (for example, highestPerformance). Valid values are: • highestQuality (default) • highQuality • mediumQuality • highestPerformance • highPerformance • mediumPerformance • nearest • bilinear • bicubic • bessel • blackman • hamming • hanning • hermite • lanczos • mitchell • quadratic See "Interpolation algorithms" on page 928 in the Usage section for more information.
blurFactor	Optional. The blur factor used for resampling. The higher the blur factor, the more blurred the image (also, the longer it takes to resize the image). Valid values are 1—10.

Usage

Use this operation to resize images or create thumbnail images while maintaining the aspect ratio. You must specify the fitWidth and FitHeight parameters; either the fitWidth or the fitHeight can be an empty string:

```
<cfset ImageScaleToFit(myImage,100,"")>
```

In this example, the ImageScaleToFit function resizes the image so that it fits in a 100x100-pixel square; the width of the resulting image is 100 pixels and the height is less than or equal to 100 pixels. For example, if the source image is 400x200 pixels, the resulting image is 100x50 pixels.

Likewise, if you specify the fitHeight parameter and an empty string for the fitWidth parameter, the ImageScaleToFit function resizes the image so that the height equals the fitHeight parameter and the width of the image is scaled proportionately:

```
<cfset ImageScaleToFit(myImage,"",100)>
```

In this example, a 400x200-pixel source image is resized to 200x100 pixels, and a 200x400-pixel image is resized to 50x100 pixels.

If you set both the fitWidth and the fitHeight parameters, the ImageScaleToFit function resizes the image proportionately so that both conditions are true: the width of the resulting image is less than or equal to the fitWidth, and the height is less than or equal to the fitHeight:

```
<cfset ImageScaleToFit(myImage,100,200)>
```

In this example, a 400x200-pixel source image is resized to 100x50 pixels, and a 200x400-pixel source image is resized to 100x200 pixels.

Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example

```
<!--- This example shows how to resize an image to fit a 100x100-pixel square while
maintaining the aspect ratio. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<cfset ImageScaleToFit(myImage,100,"","lanczos")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageSetAntialiasing

Description
Switches antialiasing on or off in rendered graphics.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageSetAntialiasing(name [, antialias])
```

See also
ImageRotateDrawingAxis, ImageSetBackgroundColor, ImageSetDrawingColor, ImageSetDrawingStroke, ImageSetDrawingTransparency, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
antialias	Optional. Antialiasing switch: • on (default) • off

Usage
The ImageSetAntialiasing function is used to turn antialiasing on and off when drawing shapes and text in an image. Antialiasing is a technique used to soften jagged edges. Turn on antialiasing when using other image functions, such as ImageDrawRoundRect and ImageRotate, to improve the quality of the rendered image. Notice that antialiasing decreases performance.

Example
Example 1

```
<!--- This example shows how to turn off antialiasing for a ColdFusion image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfset myImage=ImageNew("../cfdocs/images/artgallery/elecia02.jpg")>
<!--- Turn off antialiasing.--->
<cfset ImageSetAntialiasing(myImage,"off")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```
<!--- This example shows how to turn on antialiasing to improve the output of the
ImageDrawRoundRect function. --->
<!--- Create a 200x200-pixel image. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Set the drawing color for the image to blue. --->
```

```
<cfset ImageSetDrawingColor(myImage,"blue")>
<!--- Turn on antialiasing. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Draw a blue filled square with round corners of arcWidth=10 and arcHeight=2. --->
<cfset ImageDrawRoundRect(myImage,5,5,190,190,"yes",10,2)>
<!--- Rotate the image 30 degrees. --->
<cfset ImageRotate(myImage,30)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageSetBackgroundColor

Description

Sets the background color for the ColdFusion image. The background color is used for clearing a region. Setting the background color only affects the subsequent `ImageClearRect` calls.

Returns

Nothing.

Category

Image functions

Function syntax

`ImageSetBackgroundColor(name, color)`

See also

`ImageClearRect`, `ImageSetAntialiasing`, `IsImageFile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`color`	Required. Background color: • Hexadecimal value of RGB color. For example, specify the color white as `##FFFFFF` or `FFFFFF`. • String value of color (for example, `"black"`, `"red"`, `"green"`). The default value of the drawing color is `"black"`. • List of three numbers for (R,G,B) values. Each value must be in the range 0—255.

Usage

If the color value is a string, specify a supported named color; see the name list in "Valid HTML named colors" on page 305. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxx", where x = 0–9 or A–F; use two number signs or none.

Use this function in conjunction with the `ImageClearRect` function to clear a rectangular area of an image and set it to a specified color.

Example

```
<!--- This example shows how to set the background color, and then draw a rectangle on an
image filled with that color. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage name="myImage" source="../cfdocs/images/artgallery/maxwell01.jpg">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Set the background color to magenta. --->
<cfset ImageSetBackgroundColor(myImage,"magenta")>
<!--- Clear the rectangle specified on myImage with the background color specified for the
image. --->
<cfset ImageClearRect(myImage,36,45,100,100)>
<!--- Display the modified image in a browser. --->
```

```
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageSetDrawingColor

Description
Sets the current drawing color for ColdFusion images. All subsequent graphics operations use the specified color.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageSetDrawingColor(name, color)`

See also
`ImageSetAntialiasing`, `ImageSetBackgroundColor`, `ImageSetDrawingStroke`, `ImageSetDrawingTransparency`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`color`	Required. Drawing color: • Hexadecimal value of RGB color. For example, specify the color white as `"##FFFFFF"` or `"FFFFFF"`. • String value of color (for example, `"black"`, `"red"`, `"green"`). The default value of the drawing color is `"black"`. • List of three numbers for (R,G,B) values. Each value must be in the range 0—255.

Usage
Use the `ImageSetDrawingColor` function to control the color of all subsequent drawing objects in ColdFusion images. For example, you can use this function to set the drawing color to red once, and then draw a circle, a square, and 10 lines in that color.

If the color value is a string, specify a supported named color; see the name list in "Valid HTML named colors" on page 305. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxx", where x = 0-9 or A-F; use two number signs or none.

Example
```
<!--- This example shows how to set the drawing color and draw several objects in that color.
--->
<!--- Use the ImageNew function to create a ColdFusion image. --->
<cfset myImage=ImageNew("",200,300)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Set the drawing color to pink. --->
<cfset ImageSetDrawingColor(myImage,"pink")>
<!--- Draw a pink line. --->
<cfset ImageDrawLine(myImage,1,1,200,300)>
<!--- Draw a pink oval. --->
```

```
<cfset ImageDrawOval(myImage,40,50,80,40)>
<!--- Draw another pink oval. --->
<cfset ImageDrawOval(myImage,15,180,80,40)>
<!--- Draw a pink rectangle. --->
<cfset ImageDrawRect(myImage,100,100,45,65)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageSetDrawingStroke

Description
Sets the drawing stroke for points and lines in subsequent ColdFusion images.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageSetDrawingStroke(name [, attributeCollection])
```

See also
ImageDrawText, ImageSetAntialiasing, ImageSetBackgroundColor, ImageSetDrawingColor, ImageSetDrawingTransparency, IsImageFile

History

ColdFusion 8: Added this function.**Parameters**

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
attributeCollection	Optional. The structure used to specify the line attributes. See the Usage section.

Usage
Use the ImageSetDrawingStroke function to control the line attributes of all subsequent drawing objects in a ColdFusion image. For example, you can use this function to set the drawing stroke to a dash pattern once, and then create a rectangle, two ovals, and five lines with that pattern.

If a blank or no attribute structure is passed, the drawing stroke is reset to the default values.

attributeCollection

Element	Description
width	Pen width, which is measured perpendicularly to the pen trajectory.
endcaps	Decoration applied to the ends of unclosed subpaths and dash segments. Subpaths that start and end on the same point are considered unclosed if they do not have a close segment: • butt • round • square
lineJoins	Type of line joins: • bevel • miter • join

Element	Description
miterLimit	The limit to trim a line join that has a mitered join decoration. (Use only when lineJoins = "miter".) A line join is trimmed when the ratio of miter length to stroke width is greater than the miterLimit value. The miter length is the diagonal length of the miter, which is the distance between the inside corner and the outside corner of the intersection. The smaller the angle formed by two line segments, the longer the miter length and the sharper the angle of intersection. The default value is 10.0, which trims all angles less than 11 degrees. Trimming miters converts the decoration of the line join to bevel.
dashArray	An array of numbers that indicates the dash pattern. For example. if dashArray is (8,4), the dash pattern is 8 pixels solid, 4 pixels blank, 8 pixels solid, 4 pixels blank, and so on.
dash_phases	An offset into the dash pattern. For example, a dash_phase of 2, and a dashArray of (8,4) produces the dash pattern of 6 pixels solid, 4 pixels blank, 8 pixels solid, 4 pixels blank, and so on.

Example

Example 1

```
<!--- This example shows how to create an attribute collection for the ImageSetDrawingStroke
function and draws a line with those attributes.
    --->
<!--- Use the ImageNew function to create a ColdFusion image. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Create an attribute collection to pass to the ImageSetDrawingStroke function. Create
a stroke that is 10-pixels wide, has round endcaps, and has a dash pattern of (8,4). --->
<cfset attr = StructNew()>
<cfset attr.width = 2>
<cfset attr.endcaps = "round">
<cfset dashPattern = ArrayNew(1)>
<cfset dashPattern[1] = 8>
<cfset dashPattern[2] = 4>
<cfset attr.dashArray = dashPattern>
<!--- Apply the attribute collection to the ImageSetDrawingStroke function for the image. -
-->
<cfset ImageSetDrawingStroke(myImage,attr)>
<!--- Draw a line on the ColdFusion image with the drawing stroke attributes. --->
<cfset ImageDrawLine(myImage,20,20,40,150)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```
<!--- Use the ImageNew function to create a ColdFusion image. --->
<cfset myImage=ImageNew("",500,500)>
<!-- Set the drawing color of the image to cyan. --->
<cfset ImageSetDrawingColor(myImage,"cyan")>
<!--- Draw a line from (30,40) to (200,190). --->
<cfset ImageDrawLine(myImage,30,30,200,200)>
<!--- Create the attribute collection for the drawing stroke. --->
<cfset attr = StructNew()>
<cfset attr.width = 1>
<cfset attr.endcaps = "round">
<cfset dashPattern = ArrayNew(1)>
<cfset dashPattern[1] = 3>
<cfset dashPattern[2] = 4>
<cfset dashPattern[3] = 8>
<cfset attr.dashArray = dashPattern>
<!--- Pass the attribute collection as an argument to the set drawing stroke
function. --->
<cfset ImageSetDrawingStroke(myImage,attr)>
<!-- Set the drawing color of the image to yellow. --->
<cfset ImageSetDrawingColor(myImage,"yellow")>
```

```
<!--- Draw a rectangle with the drawing stroke specified. --->
<cfset ImageDrawRect(myImage,200,50,210,200)>
<!-- Reset the drawing stroke. -->
<cfset ImageSetDrawingStroke(myImage)>
<!--- Draw a green quadratic curve. --->
<cfset ImageSetDrawingColor(myImage,"green")>
<cfset ImageDrawQuadraticCurve(myImage,120,320,5,15,380,280)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageSetDrawingTransparency

Description
Specifies the degree of transparency of drawing functions.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageSetDrawingTransparency(name, percent)`

See also
ImageSetAntialiasing, ImageSetBackgroundColor, ImageSetDrawingColor, ImageSetDrawingStroke, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
percent	Required. Percent of transparency: • 0 = opaque • 100 = transparent Decimal values are valid.

Usage
By default drawing images are opaque. Use this function to create watermarks or other translucent images. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
Example 1

```
<!--- This example shows how to draw semitransparent text over an image.
    --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/austin01.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Set the drawing transparency to 40%. --->
<cfset ImageSetDrawingTransparency(myImage,40)>
<!--- Set the text drawing attributes. --->
<cfset attr = StructNew()>
<cfset attr.size = 40>
<cfset attr.style = "bold">
<!--- Specify the text string and the location of the text on the image.
    --->
<cfset ImageDrawText(myImage,"SOLD!",40,100,attr)>
```

```
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```
<!--- This example shows how to create a watermark from the a JPEG file.
--->
<!--- Create a ColdFusion image from a JPEG file. --->
<cfimage source="../cfdocs/getting_started/photos/somewhere.jpg" name="myImage"
action="read">
<!--- Set the drawing transparency to 75%. --->
<cfset ImageSetDrawingTransparency(myImage,75)>
<!--- Create a ColdFusion image from a picture in the cfartgallery. --->
<cfimage source="../cfdocs/images/artgallery/raquel05.jpg" name="myImage2" action="read">
<!--- Set the drawing transparency to 30%. --->
<cfset ImagesetDrawingTransparency(myImage,30)>
<!--- Paste the ColdFusion log over the picture at coordinates (0,0).--->
<cfset ImagePaste(myImage,myImage2,0,0)>
<!--- Display the two source images and the result. --->
<cfimage source="#myImage#" destination="watermark.jpg" action="write" overwrite="yes">
<img src="../cfdocs/getting_started/photos/somewhere.jpg">
<Img src="../cfdocs/images/artgallery/raquel05.jpg">
<img src="watermark.jpg">
```

Example 3

```
<!--- This code creates a ColdFusion image to be used as a watermark. --->
<cfimage action="read" name="logo" source="../cfdocs/getting_started/photos/somewhere.jpg">
<cfset imageGrayscale(logo)>
<cfset imageRotate(logo,45)>
<!--- This code creates the ColdFusion image to be used as the base image.
--->
<cfimage action="read" source="../cfdocs/images/artgallery/raquel05.jpg" name="baseImage">
<!--- This code sets the drawing transparency for the base image to 80%.
--->
<cfset ImageSetDrawingTransparency(baseImage,80)>
<!--- This code pastes the watermark image onto the base image at the coordinates (0,0). --->
<cfset ImagePaste(baseImage,logo,0,0)>
<!--- This code writes the result to a file. --->
<cfimage action="write" source="#baseImage#" destination="abc_watermark.jpg"
overwrite="yes">
<!--- This code displays the image used as a watermark and the result. --->
<img src="../cfdocs/getting_started/photos/somewhere.jpg"/>
<img src="abc_watermark.jpg"/>
```

ImageSharpen

Description
Sharpens a ColdFusion image by using the unsharp mask filter.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageSharpen(name [, gain])
```

See also
ImageBlur, ImageSetAntialiasing, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
gain	Optional. -1 <= gain <= 2. Gain values can be integers or real numbers. The default value is 1.0. The value determines whether the image is blurred or sharpened: • If > 0, the image is sharpened. • If = 0, no effect • If < 0, the image is blurred.

Usage
Use this function to sharpen outlines in photographs. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul01.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Sharpen myImage by 2. --->
<cfset ImageSharpen(myImage,2)>
<!--- Write the sharpened image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpg" overwrite="yes">
<!--- Display the original and the sharpened images. --->
<img src="../cfdocs/images/artgallery/paul01.jpg"/>
<img src="test_myImage.jpg"/>
```

ImageShear

Description
Shears an image either horizontally or vertically. For each pixel (x, y) of the destination, the source value at the fractional subpixel position (x', y') is constructed by means of an Interpolation object and written to the destination.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageShear(name, shear [, direction, interpolation])`

See also
`ImageSetAntialiasing`, `ImageShearDrawingAxis`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`shear`	Required. Shear value. Coordinates can be integers or real numbers.
`direction`	Optional. Shear direction: • `horizontal` (default) • `vertical`
`interpolation`	Optional. Type of interpolation: • `nearest`: Applies the nearest neighbor method of interpolation. Image quality is lower than the other interpolation methods, but processing is fastest (default). • `bilinear`: Applies the bilinear method of interpolation. The quality of the image is less pixelated than the default, but processing is slower. • `bicubic`: Applies the bicubic method of interpolation. Generally, the quality of image is highest with this method and processing is slowest.

Usage
Use this function to distort an image.

If the `direction` parameter is set to `horizontal`, x' = (x - y*shear) and y' = y.

If the `direction` parameter is set to `vertical`, x' = x and y' = (y - x*shear).

Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example
```
<!--- This example shows how to shear an image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
```

```
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Shear the image by a factor of 1 on a horizontal axis. --->
<cfset ImageShear(myImage,1,"horizontal")>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageShearDrawingAxis

Description
Shears the drawing canvas.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageShearDrawingAxis(name, shx, shy)`

See also
`ImageRotateDrawingAxis`, `ImageSetAntialiasing`, `ImageShear`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`shx`	Required. The multiplier by which coordinates are shifted in the positive x axis direction as a function of the y coordinate.
`shy`	Required. the multiplier by which coordinates are shifted in the positive y axis direction as a function of the x coordinate.

Usage
For each pixel (x,y) of the destination, the source value at the fractional subpixel position (x',y') is constructed by means of an interpolation object and written to the destination.

If the `direction` parameter is equal to `horizontal`, `x' = (x - y*shear)` and `y' = y`.

If the `direction` parameter is equal to `vertical`, `x' = x and y' = (y - x*shear)`.

Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example
```
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the shear drawing axis. --->
<cfset ImageShearDrawingAxis(myImage,0.5,0.5)>
<!--- Draw a rectangle on the image with the shear settings. --->
<cfset ImageDrawRect(myImage,50,50,50,50)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageTranslate

Description
Copies an image to a new location on the plane.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageTranslate(name, xTrans, yTrans [, interpolation])
```

See also
ImageSetAntialiasing, ImageShear, ImageTranslateDrawingAxis, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
xTrans	Required. Displacement in the x direction.
yTrans	Required. Displacement in the y direction.
interpolation	Optional. Type of interpolation used for resampling: • nearest: Applies the nearest neighbor method of interpolation. Image quality is lower than the other interpolation methods, but processing is fastest (default). • bilinear: Applies the bilinear method of interpolation. The quality of the image is less pixelated than the default, but processing is slower. • bicubic: Applies the bicubic method of interpolation. Generally, the quality of image is highest with this method and processing is slowest.

Usage
For each pixel (x, y) of the destination, the source value at the fractional subpixel position (x - xTrans, y - yTrans) is constructed by means of an interpolation object and written to the destination. If both xTrans and yTrans are integral, the operation wraps the source image to change the image's position in the coordinate plane.

Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
```
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/aiden01.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Offset the image's position to (20,10). --->
<cfset ImageTranslate(myImage,20,10)>
<!--- Display the offset image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageTranslateDrawingAxis

Description

Translates the origin of the image context to the point (x,y) in the current coordinate system. Modifies the image context so that its new origin corresponds to the point (x,y) in the image's original coordinate system.

Returns

Nothing.

Category

Image functions

Function syntax

`ImageTranslateDrawingAxis(name, x, y)`

See also

`ImageSetAntialiasing`, `ImageSetDrawingColor`, `ImageSetDrawingStroke`, `ImageSetDrawingTransparency`, `ImageShearDrawingAxis`, `ImageTranslate`, `IsImageFile`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`x`	x coordinate
`y`	y coordinate

Usage

All coordinates used in subsequent rendering operations on this image context are relative to the new origin. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example

```
<!--- Create a 200x200-pixel image. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Translate the origin to (100,20). --->
<cfset ImageTranslateDrawingAxis(myImage,100,20)>
<!--- Draw a rectangle at the offset location. --->
<cfset ImageDrawRect(myImage,50,60,40,50)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

ImageWrite

Description
Writes a ColdFusion image to the specified filename or destination.

Returns
Nothing.

Category
Image functions

Function syntax
`ImageWrite(name [, destination, quality])`

See also
`cfimage`, `GetWriteableImageFormats`, `ImageNew`, `ImageRead`, `IsImageFile`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion image on which this operation is performed.
`destination`	Optional. The absolute or relative pathname where you write the file.
	If you create the image with the `ImageNew` function or another operation where you do not specify the file-name, you must specify the `destination` parameter.
	The file format is derived from the extension of the filename. The default value for this parameter is the filename of the original image source.
`quality`	Optional. Defines the JPEG quality used to encode the image. This parameter applies only to destination files with an extension of JPG or JPEG. Valid values are fractions that range from 0 through 1 (the lower the number, the lower the quality). The default value is 0.75.

Usage
The file format is derived from the file extension, therefore, use this function to convert images.

For a list of valid formats to write, see"Supported image file formats" on page 304. To retrieve a list of writable formats on the server where the ColdFusion application is deployed, use the `GetWriteableImageFormats` function.

Note: *Converting images between one file format to another is time-consuming . Also, image quality can degrade; for example, PNG images support 24-bit color, but GIF images support only 256 colors. Converting transparent images (images with alpha) can degrade image quality.*

Example
```
<!--- This example shows how to convert a GIF image to a PNG image. --->
<!--- Use the ImageNew function to create a ColdFusion image. --->
<cfset myImage = ImageNew("http://www.google.com/images/logo_sm.gif")>
<!--- Convert the image to a PNG format. --->
<cfset ImageWrite(myImage,"google.png")>
<!--- Display the PNG image. --->
<img src="google.png">
```

ADOBE COLDFUSION 8 | 954
CFML Reference

ImageWriteBase64

Description
Writes Base64 images to the specified filename and destination.

Returns
Base64 string.

Category
Image functions

Function syntax
```
ImageWriteBase64(name, destination, format [, inHTMLFormat])
```

See also
cfimage, ImageReadBase64, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
destination	Required. The absolute or relative pathname where you write the file.
format	Required. The format
inHTMLFormat	Optional. Specify whether Base64 output includes the headers used by the Base64 images in the HTML tag ("data:image/base64,..."):• yes• no (default)

Usage
You use the ImageWriteBase64 function to encode image data as a string of printable characters. This is useful for several applications,including sending images by e-mail and storing images in database text fields.

If you do not specify a file format, ColdFusion cannot recognize the format required to encode the image before converting to Base64, and generates errors.

You can verify whether ColdFusion reads a Base64 string properly in the following ways:

- Use the cfdump tag. For example:
- Use the ImageInfo function. For example:
- Use the ImageWrite function and save the image as a JPEG file. Then open the JPEG file in a browser or imaging application.

Example
```




```

```
<cfset ImageWriteBase64(myImage,"jeffBase64.txt","jpg","yes")>
```

ImageXORDrawingMode

Description
Sets the paint mode of the image to alternate between the image's current color and the new specified color.

Returns
Nothing.

Category
Image functions

Function syntax
```
ImageXORDrawingMode(name, c1)
```

See also
ImageSetDrawingColor, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
name	Required. The ColdFusion image on which this operation is performed.
c1	Required. XOR alternation color: • Hexadecimal value of the RGB color. For example, specify the color white as "##FFFFFF" or "FFFFFF". • String value of color (for example, "black", "red", "green").

Usage
This function alternates pixels between the current color and a new specified XOR (exclusive Or) color.

When drawing operations are performed, pixels that are the current color are changed to the specified color, and vice versa.

Pixels that are of colors other than current color or the new specified color are changed in an unpredictable but reversible manner. If the same figure is drawn twice, all pixels are restored to their original values.

If the color value is a string, specify a supported named color; see the name list in "Valid HTML named colors" on page 305. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxx", where x = 0–9 or A–F; use two number signs or none.

Example
```
<!--- This example shows how to draw rectangles with alternating colors where they overlap.
--->
<!--- Use the ImageNew function to create a 300x200-pixel image in RGB format. --->
<cfset myImage = ImageNew("",300,200,"rgb")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Set the drawing color to white. --->
<cfset ImageSetDrawingColor(myImage,"white")>
<!--- Draw a white filled rectangle that is the size of the original image. --->
<cfset ImageDrawRect(myImage,0,0,300,200,"yes")>
<!--- Set the XOR drawing mode to white. --->
```

```
<cfset ImageXORDrawingMode(myImage,"white")>
<!--- Set the drawing color to red. --->
<cfset ImageSetDrawingColor(myImage,"red")>
<!--- Draw a filled red rectangle. --->
<cfset ImageDrawRect(myImage,50,50,150,100,"yes")>
<!--- Translate the drawing axis to (25,25). --->
<cfset ImageTranslateDrawingAxis(myImage,25,25)>
<!--- Set the drawing color to blue. --->
<cfset ImageSetDrawingColor(myImage,"blue")>
<!--- Draw a blue filled rectangle at the offset location. --->
<cfset ImageDrawRect(myImage,50,50,150,100,"yes")>
<!--- Save the ColdFusion image as a PNG file. --->
<cfset ImageWrite(myImage,"xortest.png")>
<!--- Display the PNG file.--->
<img src="xortest.png">
```

IncrementValue

Description

Adds one to an integer.

Returns

The integer part of *number,* incremented by one.

Category

Mathematical functions

Function syntax

`IncrementValue(`*`number`*`)`

See also

`DecrementValue`

Parameters

Parameter	Description
`number`	Number to increment

Example

```
<h3>IncrementValue Example</h3>
<p>Returns the integer part of a number incremented by one.
<p>IncrementValue(0): <cfoutput>#IncrementValue(0)#</cfoutput>
<p>IncrementValue("1"): <cfoutput>#IncrementValue("1")#</cfoutput>
<p>IncrementValue(123.35): <cfoutput>#IncrementValue(123.35)#</cfoutput>
```

InputBaseN

Description

Converts *string*, using the base specified by *radix*, to an integer.

Returns

A number in the range 2-36, as a string.

Category

Mathematical functions

Function syntax

`InputBaseN(string, radix)`

See also

`FormatBaseN`

Parameters

Parameter	Description
`string`	A string or a variable that contains one. String that represents a number, in the base specified by `radix`.
`radix`	Base of the number represented by `string`, in the range 2—36.

Example

```
<h3>InputBaseN Example</h3>

<p>FormatBaseN converts a number to a string in the base specified by Radix.
<p>
<cfoutput>
<br>FormatBaseN(10,2): #FormatBaseN(10,2)#
<br>FormatBaseN(1024,16): #FormatBaseN(1024,16)#
<br>FormatBaseN(125,10): #FormatBaseN(125,10)#
<br>FormatBaseN(10.75,2): #FormatBaseN(10.75,2)#
</cfoutput>
<h3>InputBaseN Example</h3>
<p>InputBaseN returns the number obtained by converting a string,
    using the base specified by Radix,.
<cfoutput>
<br>InputBaseN("1010",2): #InputBaseN("1010",2)#
<br>InputBaseN("3ff",16): #InputBaseN("3ff",16)#
<br>InputBaseN("125",10): #InputBaseN("125",10)#
<br>InputBaseN(1010,2): #InputBaseN(1010,2)#
</cfoutput>
```

Insert

Description

Inserts a substring in a string after a specified character position. If `position = 0`, prefixes the substring to the string.

Returns

A string.

Category

String functions

Function syntax

`Insert(substring, string, position)`

See also

`RemoveChars`, `Len`

Parameters

Parameter	Description
`substring`	A string or a variable that contains one. String to insert.
`string`	A string or a variable that contains one. String into which to insert `substring`.
`position`	Integer or variable; position in string after which to insert `substring`.

Example

```
<h3>Insert Example</h3>

<cfif IsDefined("FORM.myString")>
    <!--- if the position is longer than the length of the string, err --->
    <cfif FORM.insertPosition GT Len(MyString)>
        <cfoutput>
        <p>This string only has #Len(MyString)# characters; therefore,
    you cannot insert the substring #FORM.mySubString# at position
    #FORM.insertPosition#.
        </cfoutput>
    </cfif>
<cfelse>
    <cfoutput>
    <p>You inserted the substring #FORM.MySubstring# into the string
    #FORM.MyString#, resulting in the following string:
    <br>#Insert(FORM.MySubString, FORM.myString,
    FORM.insertposition)#
    </cfoutput>
</cfif>
```

Int

Description

Calculates the closest integer that is smaller than *number*. For example, it returns 3 for `Int(3.3)` and for `Int(3.7)`; it returns -4 for `Int(-3.3)` and for `Int(-3.7)`

Returns

An integer, as a string.

Category

Mathematical functions

Function syntax

`Int(number)`

See also

`Ceiling`, `Fix`, `Round`

Parameters

Parameter	Description
`number`	Real number to round down to an integer.

Example

```
<h3>Int Example</h3>
<p>Int returns the closest integer smaller than a number.

<p>Int(11.7) : <cfoutput>#Int(11.7)#</cfoutput>
<p>Int(-11.7) : <cfoutput>#Int(-11.7)#</cfoutput>
<p>Int(0) : <cfoutput>#Int(0)#</cfoutput>
```

IsArray

Description
Determines whether a value is an array.

Returns
True, if *value* is an array, or a query column object.

Category
Array functions, Decision functions

Function syntax
```
IsArray(value [, number ])
```

See also
Array functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History
ColdFusion MX:

- Changed behavior: if the *value* parameter contains a reference to a query result column, this function now returns True. For example: `isArray(MyQuery['Column1'])` returns True. (In earlier releases, it returns False.)
- Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
value	Variable or array name
number	Dimension; function tests whether the array has exactly this dimension

Usage
Use this function to determine whether a value is an array or query column. This function evaluates a Java array object, such as a vector object, as having one dimension.

Example
```
<h3>IsArray Example</h3>

<!--- Make an array --->
<cfset MyNewArray = ArrayNew(1)>
<!--- set some elements --->
<cfset MyNewArray[1] = "element one">
<cfset MyNewArray[2] = "element two">
<cfset MyNewArray[3] = "element three">
<!--- is it an array? --->
<cfoutput>
    <p>Is this an array? #IsArray(MyNewArray)#
    <p>It has #ArrayLen(MyNewArray)# elements.
    <p>Contents: #ArrayToList(MyNewArray)#
</cfoutput>
```

IsAuthenticated

Description

This function is obsolete. Use the newer security tools; see "Conversion functions" on page 641 and "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*.

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later ColdFusion releases.

IsAuthorized

Description

This function is obsolete. Use the newer security tools; see "Conversion functions" on page 641 and "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later releases.

IsBinary

Description

Determines whether a value is stored as binary data.

Returns

True, if the value is binary; False, otherwise.

Category

Decision functions

Function syntax

`IsBinary(value)`

See also

`ToBinary`, `ToBase64`, `IsNumeric`, `YesNoFormat`

Parameters

Parameter	Description
`value`	Number or string

Example

```
<!--- Create a string of all ASCII characters (32-255)
 and concatenate them together. --->
<cfset charData ="">
<cfloop index="data" from="32" to="255">
    <cfset ch=chr(data)>
    <cfset charData=charData & ch>
</cfloop>

<b>The following string is the concatenation of all characters (32 to 255) from the ASCII
table.</b><br><br>
<cfoutput>#htmleditformat(charData)#</cfoutput>
<br><br>
<!--- Create a Base 64 representation of this string. --->
<cfset data64=toBase64(charData)>
<!--- Convert string to binary. --->
<cfset binaryData=toBinary(data64)>
<!--- Check to see if it really is a binary value. --->
<cfif IsBinary(binaryData)>
The binaryData variable is binary!<br>
</cfif>
<!--- Convert binary data back to Base 64. --->
<cfset another64=toBase64(binaryData)>
<cfif Not IsBinary(another64)>
The another64 variable is NOT binary!<br>
</cfif>
<!--- Compare another64 with data64 to ensure that they are equal. --->
<cfif another64 eq data64>
    Base 64 representation of binary data is identical to the Base 64
    representation of string data.
<cfelse>
    <h3>Conversion error.</h3>
</cfif>
```

IsBoolean

Description

Determines whether a value can be converted to Boolean

Returns

True, if the parameter can be converted to Boolean; False, otherwise.

Category

Decision functions

Function syntax

`IsBoolean(value)`

See also

`IsNumeric`, `IsValid`, `YesNoFormat`

Parameters

Parameter	Description
`value`	Number or string

Example

```
<h3>IsBoolean Example</h3>

<cfif IsDefined("FORM.theTestValue")>
    <cfif IsBoolean(FORM.theTestValue)>
        <h3>The expression <cfoutput>#DE(FORM.theTestValue)#</cfoutput> is
    Boolean</h3>
    <cfelse>
        <h3>The expression <cfoutput>#DE(FORM.theTestValue)#</cfoutput> is not Boolean</h3>
    </cfif>
</cfif>

<form action = "isBoolean.cfm">
<p>Enter an expression, and discover whether it can be evaluated to a
    Boolean value.

<input type = "Text" name = "TheTestValue" value = "1">
<input type = "Submit" value = "Is it Boolean?" name = "">
</form>
```

IsCustomFunction

Description

Determines whether a name represents a custom function.

Returns

True, if *name* can be called as a custom function; False, otherwise.

Category

Decision functions

Function syntax

`IsCustomFunction(name)`

Parameters

Parameter	Description
`name`	Name of a custom function. Must not be in quotation marks. If not a defined variable or function name, ColdFusion generates an error.

Usage

The `IsCustomFunction` function returns True for any function that can be called as a custom function, including functions defined using CFScript `function` declarations and `cffunction` tags, and functions that are ColdFusion component methods. For CFC methods, you must first instantiate the component.

Note: *To prevent undefined variable exceptions, always precede* `IsCustomFunction` *with an* `IsDefined` *test, as shown in the example.*

Example

```
<h3>IsCustomFunction Example</h3>
<cfscript>
function realUDF() {
    return 1;
}
</cfscript>
<cfset X = 1>

<!--- Example that fails existence test --->
<cfif IsDefined("Foo") AND IsCustomFunction(Foo)>
    Foo is a UDF.<br>
</cfif>

<!--- Example that passes existence test but fails IsCustomFunction --->
<cfif IsDefined("X") AND IsCustomFunction(X)>
    X is a UDF.<br>
</cfif>

<!--- Example that passes both tests--->
<cfif IsDefined("realUDF") AND IsCustomFunction(realUDF)>
    realUDF is a function.<br>
</cfif>

<!--- Example using a CFC, defined in TestCFC.cfc--->
<cfobject component="TestCFC" name="myTestCFCobject">
<CFIF IsDefined("myTestCFCobject.testFunc") AND
        IsCustomFunction(myTestCFCobject.testFunc)>
```

```
    myTestCFCobject.testFunc is a function.
</CFIF>
```

IsDate

Description

Determines whether a string or Java object can be converted to a date/time value.

Returns

True, if *string* can be converted to a date/time value; False, otherwise. ColdFusion converts the Boolean return value to its string equivalent, "Yes" or "No."

Category

Date and time functions, Decision functions

Function syntax

`IsDate(string)`

See also

`CreateDateTime`, `IsNumericDate`, `IsValid`, `LSDateFormat`, `LSIsDate`, `ParseDateTime`

Parameters

Parameter	Description
`string`	A string or a variable that contains one.

Usage

This function checks against U.S. date formats only. For other date support, see `LSDateFormat`.

A date/time object falls in the range 100 AD–9999 AD.

Example

```
<h3>IsDate Example</h3>
<cfif IsDefined("FORM.theTestValue")>
    <cfif IsDate(FORM.theTestValue)>
    <h3>The string <cfoutput>#DE(FORM.theTestValue)#</cfoutput>
     is a valid date</h3>
    <cfelse>
    <h3>The string <cfoutput>#DE(FORM.theTestValue)#</cfoutput>
     is not a valid date</h3>
    </cfif>
</cfif>
<form action = "isDate.cfm" method="post">
<p>Enter a string, find whether it can be evaluated to a date value.
<p><input type = "Text" name = "TheTestValue" value = "<cfoutput>#Now()#
    </cfoutput>">
<input type = "Submit" value = "Is it a Date?" name = "">
</form>
```

IsDDX

Description
Determines whether a DDX file exists and is valid, or if a string contains valid DDX instructions.

Returns
True, if the value represents a valid DDX file or string. False, otherwise.

Category
Decision functions

Function syntax
```
IsDDX("path or string")
```

See also
IsPDFObject, IsPDFFile, cfpdf

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
path or string	Pathname to the DDX file or a string of DDX instructions. The pathname can be absolute or relative to the CFM page that calls it and must be enclosed in quotation marks.

Usage
This function returns False if the pathname to the DDX file is invalid, the pathname to the DDX file is null, the DDX file does not conform to the schema supported by ColdFusion, or the DDX instructions are invalid.

Example
```
<cfif IsDDX("TOCformat.ddx")>

    <cfset inputStruct=StructNew()>
    <cfset inputStruct.Doc0="title.pdf">
    <cfset inputStruct.Doc1="Chap1.pdf">
    <cfset inputStruct.Doc2="Chap2.pdf">
    <cfset inputStruct.Doc3="Chap3.pdf">
    <cfset inputStruct.Doc4="Chap4.pdf">

    <cfset outputStruct=StructNew()>
    <cfset outputStruct.Out1="myBook.pdf">

    <cfpdf action="processddx" ddxfile="TOCformat.ddx" inputfiles="#inputStruct#"
outputfiles="#outputStruct#" name="ddxVar">

    <cfoutput>#ddxVar.Out1#</cfoutput>

<cfelse>
    <p>This is not a valid DDX file.</p>
</cfif>
```

IsDebugMode

Description
Determines whether debugging output is enabled.

Returns
True, if debugging mode is set in the ColdFusion Administrator; False if debugging mode is disabled.

Category
Decision functions

Function syntax
```
IsDebugMode()
```

See also
cfsetting

Description
If debugging output is enabled in ColdFusion Administrator and has not been overridden by setting the `cfsetting` tag `showDebugOutput` attribute to No, the `IsDebugMode` function returns Yes; No, otherwise.

Example
```
<h3>IsDebugMode Example</h3>
<cfif IsDebugMode()>
 <h3>Debugging is set in the ColdFusion Administrator</h3>
<cfelse>
 <h3>Debugging is disabled</h3>
</cfif>
```

IsDefined

Description

Evaluates a string value to determine whether the variable named in it exists.

This function is an alternative to the `ParameterExists` function, which is deprecated.

Returns

True, if the variable is found, or, for a structure, if the specified key is defined; False, otherwise.

The return value is False for an array or structure element referenced using bracket notation. For example, `IsDefined("myArray[3]")` always returns False, even if the array element *myArray[3]* has a value.

Category

Decision functions

Function syntax

`IsDefined("variable_name")`

See also

`Evaluate`

History

ColdFusion MX: Changed behavior: this function can process only the following constructs:

- A simple variable
- A named variable with dot notation
- A named structure with dot notation (for example: `mystruct.key`)

Parameters

Parameter	Description
`variable_name`	String, enclosed in quotation marks. Name of variable to test for.

Usage

When working with scopes that ColdFusion exposes as structures, the `StructKeyExists` function can sometimes replace this function. The following lines are equivalent:

```
if(isDefined("form.myVariable"))
if(structKeyExists(form,"myVariable"))
```

Example

```
<cfif IsDefined("form.myString")>
    <p>The variable form.myString has been defined, so show its contents.
    This construction allows us to place a form and its resulting action code
    on the same page and use IsDefined to control the flow of execution.</p>
    <p>The value of "form.myString" is <b><i>
    <cfoutput>#form.myString#</cfoutput></i></b>
<cfelse>
    <p>During the first time through this template, the variable "form.myString"
    has not yet been defined, so we do not try to evaluate it.
</cfif>

<form action="#CGI.Script_Name" method="POST">
```

```
<input type="Text" name="MyString" value="My sample value">
<input type="Submit" name="">
</form>
```

IsImage

Description
Determines whether a variable returns a ColdFusion image.

Returns
True, if the value is a ColdFusion image; False, otherwise.

Category
Image functions

Function syntax
`IsImage(name)`

See also
`cfimage`, `ImageGetBlob`, `ImageInfo`, `ImageNew`

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
`name`	Required. The ColdFusion variable that is checked.

Usage
Use this function to determine whether a variable returns a ColdFusion image.

Example
```
<cfif IsImageFile("images/#form.art#")>
<cfset myImage=ImageNew("images/#form.art#")>
...
<cfset IsImage("#myImage#")>
<cfimage action="writeToBrowser" source="#myImage#">
</cfif>
```

IsImageFile

Description
Verifies whether an image file is valid.

Returns
True, if the value is a valid image file; False, otherwise.

Category
Image functions

Function syntax
```
IsImageFile("path")
```

See also
cfimage, ImageGetBlob, ImageInfo, ImageNew, IsImage

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
path	Required. The pathname of the file to be checked. The pathname can be absolute or relative to the CFM page and must be enclosed in quotation marks.

Usage
Use this function to determine whether an image file is valid. This function returns a False value if the image file format is not supported by the server where ColdFusion is deployed, or if the pathname to the image file is null or invalid.

For a list of standard image file formats supported by ColdFusion, see "Supported image file formats" on page 304. To determine which image file formats are supported on the server where ColdFusion is deployed, use the "GetReadableImageFormats" on page 852 and "GetWriteableImageFormats" on page 869.

Example
```
<!--- Use the IsImageFile function to determine whether an image retrieved from the artwork
table in the cfartgallery database is a valid image file. --->
<cfif IsImageFile("images/#artwork.largeImage#")>
    <cfset myImage=ImageNew("images/#artwork.largeImage#")>
    <cfset ImageResize(myImage,50,"")>
    <cfimage action="writeToBrowser" source="#myImage#">
<cfelse>
    <p>I'm sorry, there is no image associated with the title you selected. Please click the
Back button and try again.</p>
    </p>
</cfif>
```

IsInstanceOf

Description

Determines whether an object is an instance of a Coldfusion interface or component, or of a Java class.

Returns

Returns true if any of the following is true:

- The object specified by the first parameter is an instance of the interface or component specified by the second parameter.
- The Java object specified by the first parameter was created by using the `cfobject` tag or `CreateObject` method and is an instance of the Java class specified by the second parameter.
- The object specified by the first parameter is an instance of a component that extends the component specified in the second parameter.
- The object specified by the first parameter is an instance of a component that extends a component that implements the interface specified in the second parameter.
- The Java object specified by the first parameter was created by using the `cfobject` tag or `CreateObject` method and is an instance of a Java class that extends the class specified by the second parameter.

Returns false otherwise.

Note: *The `isInstanceOf` function returns `false` if the CFC specified by the `object` parameter does not define any functions.*

Category

Decision functions, Extensibility functions

Function syntax

`IsInstanceOf(`*`object, typeName`*`)`

See also

`cfcomponent`, `cfinterface`, `cfobject`

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`object`	The CFC instance or Java object that you are testing
`typeName`	The name of the interface, component, or Java class of which the object might be an instance

Usage

For Java objects, the comparison is valid only if you created the object specified in the first parameter by using a `cfobject` tag or `CreateObject` method.

Example

```
<!--- to use this example, create an I1.cfc interface, as follows: --->

<cfinterface>
```

```
    <cffunction name = "method1">
    </cffunction>
</cfinterface>

<!--- Create a C1.cfc component that implements the I1.cfc interface, as
    follows: --->
<cfcomponent implements=I1>
    <cffunction name = "method1">
        <cfoutput>C1.method1 called</cfoutput>
    </cffunction>
</cfcomponent>

<!--- Create a test.cfm file as follows and display the page. --->
<!--- Create an instance of the C1 CFC, which implements the I1 interface.
    --->
<cfset c1obj = CreateObject("Component", "C1")>
<!--- Create a Java object --->
<cfset javaObj = createobject("java", "java.lang.System")>

<cfoutput>
    IsInstanceOf(c1obj,"C1") = #IsInstanceOf(c1obj,"C1")#
        (Expected = YES)<br><br>
    IsInstanceOf(c1obj,"I1") = #IsInstanceOf(c1obj,"I1")#
        (Expected = YES)<br><br>
    IsInstanceOf(c1obj,"C2") = #IsInstanceOf(c1obj,"C2")#
        (Expected = NO)<br><br>
    IsInstanceOf(javaObj,"java.lang.System") =
            #IsInstanceOf(javaObj,"java.lang.System")# (Expected = YES)<br><br>
    IsInstanceOf(javaObj,"java.lang.String") =
            #IsInstanceOf(javaObj,"java.lang.String")# (Expected = NO)<br><br>
</cfoutput>
```

IsJSON

Description

Evaluates whether a string is in valid JSON (JavaScript Object Notation) data interchange format.

Returns

True if the parameter is a valid JSON value.

False if the parameter is not a valid JSON data representation.

Category

Conversion functions

Syntax

`IsJSON(var)`

See also

DeserializeJSON, SerializeJSON, cfajaxproxy, "Using Ajax Data and Development Features" on page 648 in the *ColdFusion Developer's Guide*, http://www.json.org

History

ColdFusion 8: Added function

Parameters

Parameter	Description
var	A string or variable that represents one.

Example

This example checks whether the data feed that is generated by the example for the SerializeJSON function contains valid JSON data.

The feed is in the form of a JavaScript function call where the parameter is a JSON string that contains the feed data. The example does the following operations:

1. Uses a cfhttp tag to get the feed (in the cfhttp.fileContent variable).
2. Strips the function call wrapper from the text.
3. Uses the IsJSON function to check whether the result of the previous step is a valid JSON format string.
4. Displays a message that indicates whether the text is valid JSON data, followed by the feed text string.

To run this example, put this file and the example for the SerializeJSON function in an appropriate location under your ColdFusion web root, replace the URL with the correct URL for the serialization example, and run this page.

```
<!--- Get the JSON Feed --->
<cfhttp url="http://localhost:8500/My_Stuff/Ajax/Books/CreateJSON_NEW.cfm">

<!--- JSON data is often distributed as a JavaScript function.
     The following REReplace functions strip the function wrapper. --->
<cfset theData=REReplace(cfhttp.FileContent,
        "^\s*[[:word:]]*\s*\(\s*","")>
<cfset theData=REReplace(theData, "\s*\)\s*$", "")>

<!--- Test to make sure you have JSON data. --->
```

```
<cfif isJSON(theData)>
    <h3>The URL you requested provides valid JSON</h3>
<cfelse>
    <h3>The URL you requested does not provide valid JSON</h3>
</cfif>
<!--- Display the data. --->
<cfoutput>#theData#</cfoutput>
```

For a more complex example that then converts the JSON data, see DeserializeJSON.

IsK2ServerABroker

Description

This function is deprecated.

Returns

True, if K2Broker is in configured with ColdFusion; False, otherwise.

Category

Decision functions, Full-text search functions, Query functions

Function syntax

```
IsK2ServerABroker()
```

See also

GetK2ServerDocCountLimit, IsK2ServerDocCountExceeded, IsK2ServerOnline

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.

ColdFusion MX: Added this function.

Example

```
<cfoutput>IsK2ServerABroker =
    $*#IsK2ServerABroker()#*$</cfoutput>
```

IsK2ServerDocCountExceeded

Description

This function is deprecated.

Returns

True, if the document count limit is exceeded; False, otherwise.

Category

Decision functions, Full-text search functions, Query functions

Function syntax

```
IsK2ServerDocCountExceeded()
```

See also

GetK2ServerDocCountLimit, IsK2ServerABroker

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.

ColdFusion 5: Added this function.

Example

```
<cfoutput>IsK2ServerDocCountExceeded =
    $*#IsK2ServerDocCountExceeded()#*$</cfoutput>
```

IsK2ServerOnline

Description

This function is deprecated because the K2Server is always running when ColdFusion is running.

Returns

True, if the K2Server is available to perform a search; False, otherwise.

Category

Decision functions, Full-text search functions, Query functions

Function syntax

```
IsK2ServerOnline()
```

See also

IsK2ServerABroker

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.

ColdFusion MX: Added this function.

Example

```
<cfoutput>IsK2ServerOnline =
    $*#IsK2ServerOnline()#*$</cfoutput>
```

IsLeapYear

Description

Determines whether a year is a leap year.

Returns

True, if *year* is a leap year; False, otherwise.

Category

Date and time functions, Decision functions

Function syntax

`IsLeapYear(year)`

See also

`DaysInYear`

Parameters

Parameter	Description
`year`	Number representing a year

Example

```
<h3>IsLeapYear Example</h3>

<cfif IsDefined("FORM.theTestValue")>
    <cfif IsLeapYear(FORM.theTestValue)>
        <h3>The year value <cfoutput>#DE(FORM.theTestValue)#</cfoutput> is a Leap Year</h3>
    <cfelse>
        <h3>The year value <cfoutput>#DE(FORM.theTestValue)#</cfoutput> is not a Leap
Year</h3>
    </cfif>
</cfif>

<form action = "isLeapYear.cfm">
<p>Enter a year value, and find out whether it is a Leap Year.
<p><input type = "Text" name = "TheTestValue" value = "
    <cfoutput>#Year(Now())#</cfoutput>">
<input type = "Submit" value = "Is it a Leap Year?" name = "">
</form>
```

IsLocalHost

Description
Determines whether the specified IP address is the localhost. This supports both IPv4 and IPv6 addresses.

Returns
True, if the IP address is the localhost; False, otherwise.

Category
Decision functions

Function syntax
`IsLocalHost(ipaddress)`

See also
`GetLocalHostIP`

History
ColdFusion MX 7.01 : Added this function.

Parameters

Parameter	Description
ipaddress	Valid IP address.

Example
```
<h3>IsLocalHost Example</h3>

<cfif IsDefined("FORM.theTestIPAddress")>
    <cfif IsLocalHost(FORM.theTestIPAddress)>
        <h3>The IP address <cfoutput>#FORM.theTestIPAddress)#</cfoutput> is the
localhost</h3>
    <cfelse>
        <h3>The IP address <cfoutput>#DE(FORM.theTestIPAddress)#</cfoutput> is not the
localhost.</h3>
    </cfif>
</cfif>

<form action = "isIPAddressLocalHost.cfm">
<p>Enter an IP address to find out if it is the localhost.
<p><input type = "Text" name = "TheTestIPAddress" value = "127.0.0.1"
<input type = "Submit" value = "Is this the localhost?" name = "">
</form>
```

IsNumeric

Description

Determines whether a string can be converted to a numeric value. Supports numbers in U.S. number format. For other number support, use LSIsNumeric.

Returns

True, if *string* can be converted to a number; False, otherwise.

Category

Decision functions

Function syntax

IsNumeric(*string*)

See also

IsBinary, IsValid

Parameters

Parameter	Description
string	A string or a variable that contains one.

Example

```
<h3>IsNumeric Example</h3>
<cfif IsDefined("FORM.theTestValue")>
    <cfif IsNumeric(FORM.theTestValue)>
    <h3>The string <cfoutput>#DE(FORM.theTestValue)#</cfoutput> can be
    converted to a number</h3>
    <cfelse>
    <h3>The string <cfoutput>#DE(FORM.theTestValue)#</cfoutput> cannot be
    converted to a number</h3>
    </cfif>
</cfif>

<form action = "isNumeric.cfm">
<p>Enter a string, and find out whether it can be evaluated to a numeric value.

<p><input type = "Text" name = "TheTestValue" value = "123">
<input type = "Submit" value = "Is it a Number?" name = "">
</form>
```

IsNumericDate

Description
Evaluates whether a real number is a valid representation of a date (date/time object).

Returns
True, if the parameter represents a valid date/time object; False, otherwise.

Category
Date and time functions, Decision functions

Function syntax
`IsNumericDate(number)`

See also
`IsDate`, `ParseDateTime`

Parameters

Parameter	Description
`number`	A real number

Usage
ColdFusion, by default, evaluates any input parameter and attempts to convert it to a real number before it passes the parameter to the `IsNumericDate` function. As a result, parameter values such as 12/12/03 and {ts '2003-01-14 10:04:13'} return True, because ColdFusion converts valid date string formats to date/time objects, which are real numbers.

Example
```
<h3>IsNumericDate Example</h3>
<cfif IsDefined("form.theTestValue")>
<!--- test if the value represents a number or a pre-formatted Date value --->

    <cfif IsNumeric(form.theTestValue) or IsDate(form.theTestValue)>
<!--- if this value is a numericDate value, then pass --->
        <cfif IsNumericDate(form.theTestValue)>
            <h3>The string <cfoutput>#DE(form.theTestValue)#</cfoutput> can be converted to
a valid numeric date</h3>
        <cfelse>
            <h3>The string <cfoutput>#DE(form.theTestValue)#</cfoutput> can not be converted
to a valid numeric date</h3>
        </cfif>
    <cfelse>
        <h3>The string <cfoutput>#DE(form.theTestValue)#</cfoutput> is not a valid numeric
date</h3>
    </cfif>

</cfif>

<form action="#cgi.script_name#" method="POST">
<p>Enter a value, and discover if it can be evaluated to a date value.
<p>
<input type="Text" name="TheTestValue" value="<CFOUTPUT>#Now()#</CFOUTPUT>">
<input type="Submit" value="Is it a Date?" name="">
</form>
```

IsObject

Description
Determines whether a value is an object.

Returns
True, if the value represents a ColdFusion object. False if the value is any other type of data, such as an integer, string, date, or struct.

Category
Decision functions

Function syntax
`IsObject(value)`

See also
IsDate, IsImage, IsNumeric, IsNumericDate, IsQuery, IsSimpleValue, IsStruct, IsWDDX, IsXmlDoc, IsXmlElem, IsXmlRoot

History
ColdFusion MX: Added this function.

Parameters

Parameter	Description
value	A value, typically the name of a variable.

Usage
This function returns False for query and XML objects.

Example
```
<!--- to use this example, create a color.cfc component as follows: --->
<!---
<cfcomponent>
<cffunction name="myFunction" access="public" returntype="string">
<!--- Create a structure object --->
        <cfset myColor = "Blue">
        <cfreturn myColor>
</cffunction>
</cfcomponent>
--->

<!--- Create an instance of the color.cfc component --->
<cfobject name="getColor" component="color">

<cfif IsObject(getColor)>
    <!--- Invoke the myFunction method --->
    <cfinvoke
        component="#getColor#"
        method="myFunction"
        returnVariable="myColor">
    </cfinvoke>

    <cfif IsDefined("myColor")>
```

```
        <!--- Output the returned variable --->
        The value of myColor = <cfoutput>#myColor#</cfoutput><p>
    </cfif>
</cfif>
```

IsPDFFile

Description
Verifies whether a PDF file is valid.

Returns
True, if the value returns a valid PDF file. False, otherwise.

Category
Decision functions

Function syntax
`IsPDFFile("path")`

See also
IsDate, IsImage, IsImageFile, IsNumeric, IsNumericDate, IsObject, IsPDFObject, IsQuery, IsSimpleValue, IsStruct, IsWDDX, IsXmlDoc, IsXmlElem, IsXmlRoot, cfpdf, cfpdfform, cfprint

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
path	Pathname to a PDF file. The pathname can be absolute or relative to the CFM page and must be enclosed in quotation marks.

Usage
This function returns False if the value is not a valid pathname to a PDF file, the pathname is null, the PDF file is not valid, or the PDF file is corrupted.

Example
```
<!--- The following code shows the action page for a form where a user chooses a PDF document
to print. --->

<cfif IsPDFFile("#form.printMe#")>
        <cfprint type="PDF" source="#myPDF#">
<cfelse>
    <p>This is not a valid PDF file or the PDF document you have chosen is not available.</p>
</cfif>
```

IsPDFObject

Description
Determines whether a value is a PDF object.

Returns
True, if the value represents a PDF object. False if the value is any other type of data, such as an integer, string, date, or structure.

Category
Decision functions

Function syntax
`IsPDFObject(value)`

See also
IsDate, IsImage, IsNumeric, IsNumericDate, IsObject, IsPDFFile, IsQuery, IsSimpleValue, IsStruct, IsWDDX, IsXmlDoc, IsXmlElem, IsXmlRoot, cfpdf, cfpdfform

History
ColdFusion 8: Added this function.

Parameters

Parameter	Description
value	A value, typically the PDF object stored as a variable name.

Usage
This function returns False for query and XML objects.

Example
```
<cfpdf source="c:\forms\quoteform.pdf" action="read" name="myPDFform"/>
<cfif IsPDFObject(myPDFform)>
    <cfpdf source=#myPDFform# action="write" destination = "c:\forms\newPDFForm.pdf">
<cfelse>
    <p>This is not a PDF.</p>
</cfif>
```

IsProtected

Description

This function is obsolete. Use the newer security tools; see "Conversion functions" on page 641 and "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later releases.

IsQuery

Description

Determines whether *value* is a query.

Returns

True, if *value* is a query.

Category

Decision functions, Query functions

Function syntax

`IsQuery(value)`

See also

`QueryAddRow`

Parameters

Parameter	Description
`value`	Query variable

Example

```
<!--- Shows an example of IsQuery and IsSimpleValue --->
<h3>IsQuery Example</h3>
<!--- define a variable called "GetEmployees" --->
<CFPARAM name = "GetEmployees" DEFAULT = "#Now()#">

<p>Before the query is run, the value of GetEmployees is
    <cfoutput>#GetEmployees#</cfoutput>

<cfif IsSimpleValue(GetEmployees)>
    <p>GetEmployees is currently a simple value
</cfif>
<!--- make a query on the snippets datasource --->
<cfquery name = "GetEmployees" datasource = "cfdocexamples">
SELECT *
FROM employees
</cfquery>

<p>After the query is run, GetEmployees contains a number of rows
    that look like this (display limited to three rows):
<cfoutput QUERY = "GetEmployees" MaxRows = "3">
<pre>#Emp_ID# #FirstName# #LastName#</pre>
</cfoutput>
<cfif IsQuery(GetEmployees)>
    GetEmployees is no longer a simple value, but the name of a query
</cfif>
```

IsSimpleValue

Description

Determines the type of a value.

Returns

True, if value is a string, number, Boolean, or date/time value; False, otherwise.

Category

Decision functions

Function syntax

`IsSimpleValue(value)`

See also

`IsValid`

Parameters

Parameter	Description
`value`	Variable or expression

Example

```
<!--- Shows an example of IsQuery and IsSimpleValue --->
<h3>IsSimpleValue Example</h3>
<!--- define a variable called "GetEmployees" --->
<cfparam name = "GetEmployees" default = "#Now()#">

<p>Before the query is run, the value of GetEmployees is
    <cfoutput>#GetEmployees#</cfoutput>

<cfif IsSimpleValue(GetEmployees)>
    <p>GetEmployees is currently a simple value
</cfif>
<!--- make a query on the snippets datasource --->
<cfquery name = "GetEmployees" datasource = "cfdocexamples">
    SELECT *
    FROM employees
</cfquery>
<p>After the query is run, GetEmployees contains a number of rows
    that look like this (display limited to three rows):
<cfoutput QUERY = "GetEmployees" MaxRows = "3">
<pre>#Emp_ID# #FirstName# #LastName#</pre>
</cfoutput>

<cfif IsQuery(GetEmployees)>
    GetEmployees is no longer a simple value, but the name of a query
</cfif>
```

IsSOAPRequest

Description
Determines whether a CFC is being called as a web service.

Returns
True if CFC is being called as a web service; False, otherwise.

Category
XML functions

History
ColdFusion MX 7: Added this function.

Function syntax
`IsSOAPRequest()`

See also
`AddSOAPRequestHeader`, `AddSOAPResponseHeader`, `GetSOAPRequest`, `GetSOAPRequestHeader`, `GetSOAPResponse`, `GetSOAPResponseHeader`; "Basic web service concepts" on page 903 in the *ColdFusion Developer's Guide*

Usage
Call this function within a CFC to determine if it is running as a web service.

Example
This example creates a CFC web service that illustrates the operation of the `IsSOAPRequest` function and also provides a web service that illustrates the operation of other ColdFusion SOAP functions.

Save the following code as headerservice.cfc in a folder called soapheaders under your web root. Test its operation, and specifically the operation of the `IsSOAPRequest` function, by executing the examples that invoke this web service. For example, see the example for `AddSOAPRequestHeader`.

```
<h3>IsSOAPRequest Example</h3>
<cfcomponent displayName="tester" hint="Test for SOAP headers">

<cffunction name="echo_me"
            access="remote"
            output="false"
            returntype="string"
            displayname="Echo Test" hint="Header test">

<cfargument name="in_here" required="true" type="string">

<cfset isSOAP = isSOAPRequest()>
<cfif isSOAP>

    <!--- Get the first header as a string and as XML --->
    <cfset username = getSOAPRequestHeader("http://mynamespace/", "username")>
    <cfset return = "The service saw username: " & username>
    <cfset xmlusername = getSOAPRequestHeader("http://mynamespace/", "username", "TRUE")>
    <cfset return = return & "<br> as XML: " & xmlusername>

    <!--- Get the second header as a string and as XML --->
    <cfset password = getSOAPRequestHeader("http://mynamespace/", "password")>
    <cfset return = return & "The service saw password: " & password>
```

```
    <cfset xmlpassword = getSOAPRequestHeader("http://mynamespace/", "password", "TRUE")>
    <cfset return = return & "<br> as XML: " & xmlpassword>

    <!--- Add a header as a string --->
<cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED
VALUE", false)>

    <!--- Add a second header using a CFML XML value --->
    <cfset doc = XmlNew()>
    <cfset x = XmlElemNew(doc, "http://www.tomj.org/myns", "returnheader2")>
    <cfset x.XmlText = "hey man, here I am in XML">
    <cfsetx.XmlAttributes["xsi:type"] = "xsd:string">
    <cfset tmp = addSOAPResponseHeader("ignoredNameSpace", "ignoredName", x)>

<cfelse>
    <!--- Add a header as a string - Must generate error!
<cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED
VALUE", false)>
    --->
<cfset return = "Not invoked as a web service">
</cfif>

<cfreturn return>

</cffunction>

</cfcomponent>
```

IsStruct

Description

Determines whether a variable is a structure.

Returns

True, if *variable* is a ColdFusion structure or is a Java object that implements the java.lang.Map interface. The return value is False if the object in *variable* is a user-defined function (UDF).

Category

Decision functions, Structure functions

Function syntax

`IsStruct(variable)`

See also

Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
variable	Variable name

Example

```
<!--- This view-only example shows the use of IsStruct. --->
<p>This file is similar to addemployee.cfm, which is called by StructNew,
    StructClear, and StructDelete. It is an example of a custom tag used to
    add employees. Employee information is passed through the employee
    structure (the EMPINFO attribute). In UNIX, you must also add the Emp_ID.
<!---
<cfswitch expression = "#ThisTag.ExecutionMode#">
<cfcase value = "start">
 <cfif IsStruct(attributes.EMPINFO)>
    <cfoutput>Error. Invalid data.</cfoutput>
    <cfexit method = "ExitTag">
 <cfelse>
    <cfquery name = "AddEmployee" datasource = "cfdocexamples">
    INSERT INTO Employees
    (FirstName, LastName, Email, Phone, Department)
    VALUES
    <cfoutput>
    (
    '#StructFind(attributes.EMPINFO, "firstname")#' ,
    '#StructFind(attributes.EMPINFO, "lastname")#' ,
    '#StructFind(attributes.EMPINFO, "email")#' ,
    '#StructFind(attributes.EMPINFO, "phone")#' ,
    '#StructFind(attributes.EMPINFO, "department")#'
    )
    </cfoutput>
    </cfquery>
</cfif>
<cfoutput><hr>Employee Add Complete</cfoutput>
```

```
</cfcase>
</cfswitch> --->
```

IsUserInAnyRole

Description

Determines whether an authenticated user belongs to any Role.

Returns

True, if the authenticated user, belongs to any Role; False, otherwise.

Category

Security functions, Decision functions

Function syntax

```
IsUserInAnyRole()
```

See also

cflogin, cfloginuser, cflogout, GetAuthUser, GetUserRoles, IsUserInRole, IsUserLoggedIn, "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function.

Usage

Role names are not case-sensitive.

To check if a user is in multiple roles, specify them in a comma-delimited list, such as "Admin,HR". Lists with multiple roles cannot contain spaces as separators; for example, do not use "Admin, HR".

Example

```
<cfif IsUserInAnyRole() >
    <cfoutput>Authenticated user is in these roles: #GetUserRoles()#</cfoutput>
<cfelseif >
    <cfoutput>Authenticated user is in no roles</cfoutput>
</cfif>
```

IsUserInRole

Description
Determines whether an authenticated user belongs to the specified Role.

Returns
True, if the authenticated user, belongs to the specified Role; False, otherwise.

Category
Security functions, Decision functions

Function syntax
`IsUserInRole("role_name")`

See also
`cflogin`, `cfloginuser`, `GetAuthUser`, `GetUserRoles`, `IsUserInAnyRole`, `IsUserLoggedIn`, “Securing Applications” on page 312 in the *ColdFusion Developer’s Guide*

History
ColdFusion MX: Added this function.

Parameters

Parameter	Description
role_name	Name of a security role

Usage
Role names are not case-sensitive.

To check if a user is in multiple roles, specify them in a comma-delimited list, such as "Admin,HR". Lists with multiple roles cannot contain spaces as separators; for example, do not use "Admin, HR".

Example
```
<cfif IsUserInRole("Admin") >
    <cfoutput>Authenticated user is an administrator</cfoutput>
<cfelse IsUserInRole("User") >
    <cfoutput>Authenticated user is a user</cfoutput>
</cfif>
```

IsUserLoggedIn

Description
Determines whether a user is logged in.

Returns
True, if the user, is logged in; False, otherwise.

Category
Security functions, Decision functions

Function syntax
```
IsUserLoggedIn()
```

See also
`cflogin`, `cfloginuser`, `GetAuthUser`, `GetUserRoles`, `IsUserInAnyRole`, `IsUserInRole`, "Securing Applications" on page 312 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Added this function.

Example
```
<cfif IsUserLoggedIn() >
    <cfinclude template="welcome.cfm">
<cfelse>
    <cfinclude template="loginform.cfm">
    <cfabort>
</cfif>
```

IsValid

Description

Tests whether a value meets a validation or data type rule.

Returns

True, if the value conforms to the rule; False, otherwise.

Category

Decision functions

Function syntax

```
IsValid(type, value)
isValid("range", value, min, max)
isValid("regex" or "regular_expression", value, pattern)
```

See also

cfparam, cfform, IsBoolean, IsDate, IsNumeric, IsSimpleValue; "Validating data with the IsValid function and the cfparam tag" on page 573 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `component` value for to the `type` attribute.

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
type	The valid format for the data; one of the following. For detailed information on validation algorithms, see "Validating form data using hidden fields" on page 566 in the *ColdFusion Developer's Guide*. • any: any simple value. Returns false for complex values, such as query objects;; equivalent to the IsSimpleValue function. • array: an ColdFusion array; equivalent to the IsArray function. • binary: a binary value;; equivalent to the IsBinary function. • boolean: a Boolean value: yes, no, true, false, or a number; equivalent to the IsBoolean function. • component: a ColdFusion component (CFC). • creditcard: a 13-16 digit number conforming to the mod10 algorithm. • date or time: any date-time value, including dates or times;; equivalent to the IsDate function.. • email: a valid email address. • eurodate: any date-time value, including US date formats and time values, • float or numeric: a numeric value; equivalent to the IsNumeric function. • guid: a Universally Unique Identifier of the form "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" where 'X' is a hexadecimal number. • integer: an integer. • query: a query object; equivalent to the IsQuery function. • range: a numeric range, specified by the min and max attributes. • regex *or* regular_expression: matches input against pattern attribute. • ssn *or* social_security_number: A U.S. social security number. • string: a string value, including single characters and numbers • struct: a structure; equivalent to the IsStruct function. • telephone: a standard US telephone number. • URL: an http, https, ftp, file, mailto, or news URL, • UUID: a ColdFusion Universally Unique Identifier, formatted 'XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXXXXXX', where 'X' is a hexadecimal number. See CreateUUID. • USdate: a U.S. date of the format mm/dd/yy, with 1-2 digit days and months, 1-4 digit years. • variableName: a string formatted according to ColdFusion variable naming conventions. • zipcode: U.S., 5- or 9-digit format ZIP codes. •
value	The value to test
min	The minimum valid value; used only for range validation
max	The maximum valid value; used only for range validation
pattern	A JavaScript regular expression that the parameter must match; used only for regex or regular_expression validation.

Usage

The IsValid function lets you assure that validation is performed on the server. You can use the cfparam tag to perform equivalent validation.

Example

The following example checks whether a user has submitted a numeric ID and a valid email address and phone number. If any of the submitted values does not meet the validation test, it displays an error message.

```
<cfif isDefined("form.saveSubmit")>
    <cfif isValid("integer", form.UserID) and isValid("email", form.emailAddr)
            and isValid("telephone", form.phoneNo)>
        <cfoutput>
            <!--- Application code to update the database goes here --->
            <h3>The email address and phone number for user #Form.UserID#
                have been added</h3>
        </cfoutput>
    <cfelse>
        <H3>You must supply a valid User ID, phone number, and email address.</H2>
    </cfif>
    <cfelse>
</cfif>

<cfform action="#CGI.SCRIPT_NAME#">
    User ID:<cfinput type="Text" name="UserID"><br>
    Phone: <cfinput type="Text" name="phoneNo"><br>
    email: <cfinput type="Text" name="emailAddr"><br>
    <cfinput type="submit" name="saveSubmit" value="Save Data"><br>
</cfform>
```

IsWDDX

Description
Determines whether a value is a well-formed WDDX packet.

Returns
True, if the value is a well-formed WDDX packet; False, otherwise.

Category
Decision functions, XML functions

Syntax
`IsWDDX(value)`

See also
"Using WDDX" on page 896 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: if the `value` parameter is not a WDDX packet, ColdFusion returns False. (In earlier releases, ColdFusion threw an error.)

Parameters

Parameter	Description
`value`	A WDDX packet

Usage
This function processes a WDDX packet with a validating XML parser, which uses the WDDX Document Type Definition (DTD).

To prevent CFWDDX deserialization errors, you can use this function to validate WDDX packets from unknown sources.

Example
```
<cfset packet="
    <wddxPacket version='1.0'>
    <header></header>
    <data>
        <struct>
        <var name='ARRAY'>
            <array length='3'>
                <string>one</string>
            <string>two</string>
            </array>
        </var>
        <var name='NUMBER'>
            <string>5</string>
        </var>
        <var name='STRING'>
            <string>hello</string>
        </var>
        </struct>
    </data>
    </wddxPacket>"
```

```
    >
<hr>
<xmp>
<cfoutput>#packet#
</xmp>
IsWDDX() returns #IsWDDX(packet)#<br>
</cfoutput>
```

IsXML

Description
Determines whether a string is well-formed XML text.

Returns
True, if the function parameter is a string that contains well-formed XML text; False, otherwise.

Category
Decision functions, XML functions

Function syntax
`IsXML(value)`

See also
`IsXmlAttribute`, `IsXmlDoc`, `IsXmlElem`, `IsXmlNode`, `IsXmlRoot`, `XmlParse`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`value`	A string containing the XML document text

Usage
This function determines whether text is well-formed XML, that is, it conforms to all XML syntax and structuring rules. The string does not have to be a complete XML document. The function does not validate against a Document Type Definition (DTD) or XML Schema.

Example
The following example creates two strings, and tests whether they are well-formed XML text:

```
<!--- A well formed XML string --->
<cfset xmlString1='<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
    </items>
</order>'
>

<!--- An invalid XML string, missing the </item> close tag --->
<cfset xmlString2='<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
    </items>
```

```
</order>'
>

<!--- Test the strings to see if they are well formed XML --->
<cfoutput>
xmlString1 contains the following text:<br><br>
#HTMLCodeFormat(xmlstring1)#
Is it well formed XML text? #IsXML(xmlString1)#<br><br>
<hr>
xmlString2 contains the following text:<br><br>
#HTMLCodeFormat(xmlstring2)#
Is it well formed XML text? #IsXML(xmlString2)#
</cfoutput>
```

IsXmlAttribute

Description
Determines whether the function parameter is an XML Document Object Model (DOM) attribute node.

Returns
True, if the function argument is an XML attribute node; False, otherwise.

Category
Decision functions, XML functions

Function syntax
`IsXmlAttribute(value)`

See also
`IsXML`, `IsXmlDoc`, `IsXmlElem`, `IsXmlNode`, `IsXmlRoot`, `XmlGetNodeType`, `XmlValidate`, "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`value`	Name of an XML attribute

Usage
This function determines whether the parameter is an XML DOM attribute node, a node with an `XMLType` value of `ATTRIBUTE`. It is useful for determining whether a value returned by the `XmlSearch` function is an XML attribute.

The DOM, and therefore ColdFusion, treats XML attributes as properties of an element and does not directly expose them as DOM nodes. For this reason, the `XmlAttributes` entries in ColdFusion XML document objects do not represent DOM attribute nodes, and tests such as the following always return False:

```
IsXmlAttribute(myxmlelement.XMlAttributes);
IsXmlAttribute(myxmlelement.XMlAttributes.myattribute);
```

The `XmlSearch` function does return attributes as XML DOM attribute nodes. For example, the following line returns an array of attribute nodes containing the quantity attributes in the xmlobject document object:

```
quantities = XmlSearch(xmlobject, '//@quantity');
```

Example
The following example creates an XML document object and gets parts of it. It then tests whether these parts are attribute nodes.

```
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
```

```
    </items>
</order>
</cfxml>

<!--- Get an array with all lastname quantity DOM attribute nodes
    (In this example there is only one entry) --->
<cfset lastnames = XmlSearch(xmlobject, '//@lastname')>

<!--- Test objects to see if they are attributes --->
<cfoutput>
<h3>Are the following XML Attribute nodes?</h3>
<!--- The order element id attribute.
         This a simple variable, not a DOM attribute node.--->
node.xmlobject.order.XmlAttributes.id:
        #IsXmlAttribute(xmlobject.order.XmlAttributes.id)#<br>
<!--- The items element --->
xmlobject.order.items: #IsXmlAttribute(xmlobject.order.items)#<br>
lastnames[1] returned by XmlSearch:
    #isXmlAttribute(lastnames[1])#<br>
</cfoutput>
```

IsXmlDoc

Description
Determines whether the function parameter is a ColdFusion XML document object.

Returns
True, if the function argument is an XML document object; False, otherwise.

Category
Decision functions, XML functions

Function syntax
`IsXmlDoc(value)`

See also
`IsXML`, `IsXmlAttribute`, `IsXmlElem`, `IsXmlNode`, `IsXmlRoot`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Added this function.

Parameters

Parameter	Description
value	Name of an XML document object

Example
The following example creates an XML Document object and a Java object and tests whether they are XML document objects:

```
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
    </items>
</order>
</cfxml>

<!--- Create a Java object --->
<cfobject type="JAVA" action="create" class="java.lang.Error" name="javaobject" >

<!--- Test the objects --->
<cfoutput>
Is xmlobject an XML document object? #IsXmlDoc(xmlobject)#<br>
Is javaobject an XML document object? #IsXmlDoc(javaobject)#<br>
</cfoutput>
```

IsXmlElem

Description
Determines whether the function parameter is an XML document object element.

Returns
True, if the function argument is an XML document object element; False, otherwise.

Category
Decision functions, XML functions

Function syntax
`IsXmlElem(value)`

See also
`IsXML`, `IsXmlAttribute`, `IsXmlDoc`, `IsXmlNode`, `IsXmlRoot`, `XmlGetNodeType`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Added this function.

Parameters

Parameter	Description
`value`	Name of an XML document object element

Example
The following example tests whether an XML document object, the document root, and an element are elements:

```
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
    </items>
</order>
</cfxml>

<!--- Test parts of the document object to see if they are elements --->
<cfoutput>
    <h3>Are the following XML document object elements?</h3>
    xmlobject: #IsXmlElem(xmlobject)#<br>
    xmlobject.XMLRoot: #IsXmlElem(xmlobject.XMLRoot)#<br>
    xmlobject.order.items: #IsXmlElem(xmlobject.order.items)#<br>
</cfoutput>
```

IsXmlNode

Description
Determines whether the function parameter is an XML document object node.

Returns
True, if the function argument is an XML document object node, including an element; False, otherwise.

Category
Decision functions, XML functions

Function syntax
`IsXmlNode(value)`

See also
`IsXML`, `IsXmlAttribute`, `IsXmlDoc`, `IsXmlElem`, `IsXmlRoot`, `XmlGetNodeType`, `XmlSearch`, `XmlValidate`;
"Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`value`	Name of an XML document object node.

Usage
This function returns True for the following components of an XML document object:

- The document object
- Elements in the object
- XmlNode objects in an element's XmlNodes array

It also returns True for XML node objects returned by the `XmlSearch` function. It does not return True for most entries in an element, including XmlText, XmlComment, XmlCdata, or the XmlAttributes array (or individual XML attributes).

Example
The following example tests whether an XML document object, an element, an attribute in the object, and an attribute returned by an `XmlSearch` function are nodes:

```
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<?xml version="1.0" encoding="UTF-8"?>
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
    </items>
</order>
```

```
</cfxml>

<!--- use XmlSearch to get an attribute node. --->
<cfset lastnames = XmlSearch(xmlobject, '//@lastname')>

<!--- Test the objects to see if they are XML nodes--->
<cfoutput>
<h3>Are the following XML nodes?</h3>
xmlobject: #IsXmlNode(xmlobject)#<br>
<!--- The items element --->
xmlobject.order.items: #IsXmlNode(xmlobject.order.items)#<br>
<!--- The order element id attribute; a simple variable, not a DOM node.--->
xmlobject.order.XmlAttributes.id:
    #IsXmlNode(xmlobject.order.XmlAttributes.id)#<br>
lastnames[1] returned by XmlSearch:
#isXmlNode(lastnames[1])#
</cfoutput>
```

IsXmlRoot

Description
Determines whether the function parameter is the root element of an XML document object.

Returns
True, if the function argument is the root object of an XML document object; False, otherwise.

Category
Decision functions, XML functions

Function syntax
`IsXmlRoot(value)`

See also
`IsXML`, `IsXmlAttribute`, `IsXmlDoc`, `IsXmlElem`, `IsXmlNode`, `XmlGetNodeType`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Added this function.

Parameters

Parameter	Description
value	Name of an XML document object

Example
The following example tests whether an XML document object, its root element, and a child element are XML root elements:

```
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<?xml version="1.0" encoding="UTF-8"?>
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
    </items>
</order>
</cfxml>

<!--- Test objects to see if they are XML root elements --->
<cfoutput>
<h3>Are the following the XML Root?</h3>
xmlobject: #IsXmlRoot(xmlobject)#<br>
xmlobject.order: #IsXmlRoot(xmlobject.order)#<br>
<!--- The order element id attribute --->
xmlobject.order.XmlAttributes.id:
    #IsXmlRoot(xmlobject.order.XmlAttributes.id)#<br>
</cfoutput>
```

JavaCast

Description

Converts the data type of a ColdFusion variable to a specified Java type to pass as an argument to Java or .NET object. Use only for scalar, string, and array arguments.

Returns

The variable, as type *type*.

Category

String functions

Function syntax

`JavaCast(type, variable)`

History

ColdFusion MX 8: Added support for bigdecimal, byte, char, and short data types and for casting Arrays.

ColdFusion MX 7: Added support for nulls.

See also

`CreateObject`, `cfobject`, "Converting between .NET and ColdFusion data types" on page 960 in and"Resolving ambiguous data types with the JavaCast function" on page 944 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`type`	Data type to which to convert variable: • bigdecimal (converts to java.math.BigDecimal) • boolean • byte • char • int • long • float • double • short • string • null • *xxx*[] where *xxx* is one of the following: • any of the preceding types, except for null • a Java class name
`variable`	A ColdFusion variable that holds a scalar or string type. Must be `""` if type is null.

Usage

Use this method to specify the Java type to use for a variable that you use when calling a Java or .NET method when the conversion between types is ambiguous; for example, if a method is overloaded and differs only in parameter type or a .NET method is declared as taking a System.Object class parameter.

Use after creating a Java object with the `cfobject` tag, before calling one of its methods. If the method takes more than one overloaded argument, you must call `JavaCast` for each one. Use `JavaCast` only when a method is overloaded (because its arguments can take more than one data type, not because the method can take a variable number of arguments).

`JavaCast` cannot be used to cast between complex objects, nor to cast to a super-class.

Because there is not a one-to-one correspondence between internally stored ColdFusion types and Java scalar types, some conversions cannot be performed.

Use the result of this function only on calls to Java or .NET objects. The following example shows the use when calling a Java method.

```
<cfscript>
     x = CreateObject("java", "test.Hello");
     x.init();
     ret = x.sayHello(JavaCast("null", ""));
</cfscript>
```

Note: *Do not assign the results of `JavaCast("null","")` to a ColdFusion variable. Unexpected results will occur.*

The format `JavaCast(type[], variable)` casts a ColdFusion Array variable to a single dimensional Array of the specified type. It cannot convert multi-dimensional arrays. You can specify a primitive type or the name of a Class as the type to cast to. For example, you can use the following format to cast a ColdFusion Array to an Array of vom.x.yMyClass objects.

```
javacast("vom.x.y.MyClass[]", myCFArr)
```

Use an array in the first `JavaCast` parameter in any of the following circumstances:

- You have two functions with signatures with the same number of parameters, and a parameter takes different types of Arrays in different signatures; for example, if you have both of the following functions: `foo(int[] x)` and `foo(String[] strs)`.
- The method parameter requires a class array in its signature; for example, `foo(com.x.y.MyClass[])`.
- The method parameter requires an Object in its signature and you need to pass an array of any particular type.

The following example shows the use of the `JavaCast` cunction to cast arrays:

You might have a fooClass class that defines the following two methods, each with two arguments where the first argument differs in the type of the array:

```
public class fooClass {
 public fooClass () {
 }
 public String foo(long[] arg) {
return "Argument was a long array";
 }
 public String foo(int[] arg) {
return "Argument was an Integer array";
 }
}
```

To be able to use these functions in your CFML, you must use the `JavaCast` function to convert the ColdFusion Arrray to the array type required by one of the functions, as shown in the following code snippet:

```
<cfset arr = [1,2,4,20,10]>
<cfset fooObj = createObject("java", "fooClass")>

<cfset fooObj.foo(javacasr("int[]", arr))>
<cfset fooObj.foo(javacast("long[]", arr))>
```

Example

The method fooMethod in the class fooClass takes one overloaded argument. The fooClass class is defined as follows:

```
public class fooClass {
    public fooClass () {
    }
    public String fooMethod(String arg) {
        return "Argument was a String";
    }
    public String fooMethod(int arg) {
        return "Argument was an Integer";
    }

}
```

Within ColdFusion, you use the following code:

```
<cfobject
action="create"
type = "java"
class = "fooClass"
name = obj>

<!--- ColdFusion can treat this as a string or a real number --->
<cfset x = 33>

Perform an explicit cast to an int and call fooMethod:<br>
<cfset myInt = JavaCast("int", x)>
<cfoutput>#obj.fooMethod(myInt)#</cfoutput>
<br><br>
Perform an explicit cast to a string and call fooMethod:<br>
<cfset myString = javaCast("String", x)>
<cfoutput>#obj.fooMethod(myString)#</cfoutput>
```

JSStringFormat

Description

Escapes special JavaScript characters, such as single-quotation mark, double-quotation mark, and newline.

Returns

A string that is safe to use with JavaScript.

Category

String functions

Function syntax

`JSStringFormat(string)`

Parameters

Parameter	Description
`string`	A string or a variable that contains one.

Usage

Escapes special JavaScript characters, so you can put arbitrary strings safely into JavaScript.

Example

```
<!--- This example shows the use of the JSStringFormat function. ----->
<h3>JSStringFormat</h3>
<cfset stringValue = "An example string value with a tab chr(8),
    a newline (chr10) and some ""quoted"" 'text'">

<p>This is the string we have created:<br>
<cfoutput>#stringValue#</cfoutput>
</p>
<cfset jsStringValue = JSStringFormat(#stringValue#)>
<!----- Generate an alert from the JavaScript string jsStringValue. ----->
<SCRIPT>
s = "<cfoutput>#jsStringValue#</cfoutput>";
alert(s);
</SCRIPT>
```

LCase

Description

Converts the alphabetic characters in a string to lowercase.

Returns

A string, converted to lowercase.

Category

String functions

Function syntax

`LCase(string)`

See also

`UCase`

Parameters

Parameter	Description
`string`	A string or a variable that contains one

Example

```
<h3>LCase Example</h3>

<cfif IsDefined("FORM.sampleText")>
    <cfif FORM.sampleText is not "">
        <cfoutput>
        <p>Your text, <b>#FORM.sampleText#</b>, returned in lowercase is
        <b>#LCase(FORM.sampleText)#</b>.</p>
    <   /cfoutput>
    <cfelse>
        <   p><b><i>Please enter some text.</i></b></p>
    </cfif>
</cfif>

<p>Enter your text. Press "submit" to see it returned in lowercase: </p>

<form method="post" action = "<cfoutput>#cgi.script_name#</cfoutput>" name="lcaseForm">
<input type = "Text" name = "SampleText" value = "SAMPLE">
<input type = "Submit" name = "" value = "submit">
</form>
```

Left

Description

Returns the leftmost *count* characters in a string.

Returns

String; the first *count* characters in the *string* parameter.

Category

String functions

Function syntax

```
Left(string, count)
```

See also

Right, Mid, Len

Parameters

Parameter	Description
string	A string or a variable that contains one.
count	A positive integer or a variable that contains one. Number of characters to return.

Example

```
<h3>Left Example</h3>

<cfif IsDefined("Form.myText")>
<!--- If len returns 0 (zero), then show error message. --->
    <cfif Len(Form.myText)>
        <cfif Len(Form.myText) LTE Form.RemoveChars>
            <cfoutput><p style="color: red; font-weight: bold">Your string #Form.myText#
only has #Len(Form.myText)# characters. You cannot output
    the #Form.removeChars# leftmost characters of this string because it is
    not long enough.</p></cfoutput>
        <cfelse>
            <cfoutput><p>Your original string: <strong>#Form.myText#</strong></p>
                <p>Your changed string, showing only the <strong>#Form.removeChars#
                    </strong> leftmost characters:
                    <strong>#Left(Form.myText, Form.removeChars)#</strong></p>
            </cfoutput>
        </cfif>
    <cfelse>
        <p style="color: red; font-weight: bold">Please enter a string of more than 0 (zero)
characters.</p>
</cfif>
</cfif>

<form action="<cfoutput>#CGI.ScriptName#</cfoutput>" method="POST">
<p>Type in some text<br />
<input type="Text" name="myText"></p>
<p>How many characters from the left do you want to show?
<select name="RemoveChars">
<option value="1">1
<option value="3" selected>3
<option value="5">5
<option value="7">7
```

```
<option value="9">9</select>
<input type="Submit" name="Submit" value="Remove characters"></p>
</form>
```

Len

Description
Determines the length of a string or binary object.

Returns
Number; length of a string or a binary object.

Category
String functions

Function syntax
`Len(string or binary object)`

See also
`ToBinary`, `Left`, `Right`, `Mid`

History
ColdFusion MX: Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode character values 0–65535. (ColdFusion 5 and earlier releases supported ASCII values 1–255. When calculating a length, some string-processing functions processed the ASCII 0 (NUL) character, but did not process subsequent characters of the string.)

Parameters

Parameter	Description
`string`	A string, the name of a string, or a binary object

Example
```
<h3>Len Example</h3>

<cfif IsDefined("Form.MyText")>
    <!--- If len returns 0 (zero), then show error message. --->
    <cfif Len(FORM.myText)>
        <cfoutput><p>Your string, <strong>"#FORM.myText#"</strong>,
        has <strong>#Len(FORM.myText)#</strong> characters.</cfoutput>
    <cfelse>
        <p style="color: red; font-weight: bold">Please enter a string of more
        than 0 characters.</p>
    </cfif>
</cfif>

<form action = "<cfoutput>#CGI.SCRIPT_NAME#</cfoutput>" method="POST">
<p>Type in some text to see the length of your string.</p>

<input type = "Text" name = "MyText"><br />
<input type = "Submit" name="Submit" value = "Count characters"><br>
</form>
```

ListAppend

Description
Concatenates a list or element to a list.

Returns
A copy of the list, with *value* appended. If `delimiter = ""`, returns a copy of the list, unchanged.

Category
List functions

Function syntax
```
ListAppend(list, value [, delimiters ])
```

See also
`ListPrepend`, `ListInsertAt`, `ListGetAt`, `ListLast`, `ListSetAt`; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`list`	A list or a variable that contains one.
`value`	An element or a list of elements.
`delimiters`	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion uses only the first character.

Usage
ColdFusion inserts a delimiter character before *value*.

The following table shows examples of `ListAppend` processing:

Statement	Output	Comment
`ListAppend('elem1,elem2', '' )`	elem1,elem2,	Appended element is empty; delimiter is last character in list; list length is 2.
`ListAppend('', 'elem1,elem2' )`	elem1,elem2	List length is 2.
`ListAppend("one___two", "three", "___")`	"one___two_three"	Inserted the first character of `delimiters` before "three."

Example
```
<h3>ListAppend Example</h3>
<!--- First, query to get some values for our list elements--->
<cfquery name = "GetParkInfo" datasource = "cfdocexamples">
    SELECT PARKNAME,CITY,STATE
    FROM PARKS WHERE PARKNAME LIKE 'AL%'
</cfquery>
<cfset temp = ValueList(GetParkInfo.ParkName)>
<cfoutput>
<p>The original list: #temp#
</cfoutput>
<!--- now, append a park name to the list --->
<cfset temp2 = ListAppend(Temp, "ANOTHER PARK")>
```

ListChangeDelims

Description

Changes a list delimiter.

Returns

A copy of the list, with each delimiter character replaced by *new_delimiter*.

Category

List functions

Function syntax

```
ListChangeDelims(list, new_delimiter [, delimiters ])
```

See also

ListFirst, ListQualify; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
new_delimiter	Delimiter string or a variable that contains one. Can be an empty string. ColdFusion processes the string as one delimiter.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Example

```
<h3>ListChangeDelims Example</h3>
<p>ListChangeDelims lets you change the delimiters of a list.
<!--- First, query to get some values for our list elements--->
<CFQUERY NAME="GetParkInfo" DATASOURCE="cfdocexamples">
    SELECT PARKNAME,CITY,STATE
    FROM Parks
    WHERE PARKNAME LIKE 'BA%'
</CFQUERY>
<CFSET temp = ValueList(GetParkInfo.ParkName)>
<cfoutput>
<p>The original list: <p>#temp#
</cfoutput>
<!--- Change the delimiters in the list --->
<CFSET temp2 = ListChangeDelims(Temp, "|:P|", ",")>
<cfoutput>
<p>After executing the statement
    <strong>ListChangeDelims(Temp, "|:P|", ",")</strong>,
    the updated list: <p>#temp2#
</cfoutput>
```

ListContains

Description
Determines the index of the first list element that contains a specified substring.

Returns
Index of the first list element that contains *substring*. If not found, returns zero.

Category
List functions

Function syntax
```
ListContains(list, substring [, delimiters ])
```

See also
ListContainsNoCase, ListFind; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
substring	A string or a variable that contains one. The search is case-sensitive.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<!--- This example shows differences between ListContains and ListFind --->
<!--- Create a list composed of the elements one, two, three. ----->
<cfset aList = "one">
<cfset aList = ListAppend(aList, "two")>
<cfset aList = ListAppend(aList, "three")>
<p>Here is a list: <cfoutput>#aList#</cfoutput>
<p><strong>ListContains</strong> checks for substring "wo" in the list elements:
<cfoutput>
    <p>   Substring "wo" is in
    <B>element #ListContains(aList, "wo")#</B> of list.
</cfoutput>
<p>ListFind cannot check for a substring within an element; therefore, in the
    code, it does not find substring "wo" (it returns 0):
<cfoutput>
    <p>   Substring "wo" is in <b>element #ListFind(aList, "wo")#
    </b> of the list.</cfoutput>
<p><p>If you specify a string that exactly equals an entire list element, such
    as "two", both ListContains and ListFind find it, in the second element:
    <p>   <strong>ListContains</strong>:
<cfoutput>
The string "two" is in <b>element #ListContains(aList, "two")#</b> of the list.
</cfoutput>
   <p><strong>ListFind</strong>:
<cfoutput>
```

```
The string "two" is in <b>element #ListFind(aList, "two")#</b> of the list.
</cfoutput>
```

ListContainsNoCase

Description

Determines the index of the first list element that contains a specified substring.

Returns

Index of the first list element that contains *substring*, regardless of case. If not found, returns zero.

Category

List functions

Function syntax

```
ListContainsNoCase(list, substring [, delimiters ])
```

See also

ListContains, ListFindNoCase; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
substring	A string or a variable that contains one. The search is case-insensitive.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example

```
<h3>ListContainsNoCase Example</h3>
<cfif IsDefined("form.letter")>
    <!--- First, query to get some values for our list --->
    <cfquery name="GetParkInfo" datasource="cfdocexamples">
        SELECT PARKNAME,CITY,STATE
        FROM Parks
        WHERE PARKNAME LIKE '#form.letter#%'
    </cfquery>
    <cfset tempList = #ValueList(GetParkInfo.City)#>
    <cfif ListContainsNoCase(tempList, form.yourCity) is not 0>
        There are parks in your city!
    <cfelse>
        <p>Sorry, there were no parks found for your city.
        Try searching under a different letter.
    </cfif>
</cfif>
```

ListDeleteAt

Description
Deletes an element from a list.

Returns
A copy of the list, without the specified element.

Category
List functions

Function syntax
```
ListDeleteAt(list, position [, delimiters ])
```

See also
ListGetAt, ListSetAt, ListLen; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
position	A positive integer or a variable that contains one. Position at which to delete element. The first list position is 1.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
To use this and other functions with the default delimiter (comma), you can code as follows:

```
<cfset temp2 = ListDeleteAt(temp, "3")>
```

To specify another delimiter, you code as follows:

```
<cfset temp2 = ListDeleteAt(temp, "3", ";")>
```

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<!--- First, query to get some values for our list elements. --->
<CFQUERY NAME="GetParkInfo" DATASOURCE="cfdocexamples">
    SELECT PARKNAME,CITY,STATE
    FROM Parks
    WHERE PARKNAME LIKE 'CHI%'
</CFQUERY>
<CFSET temp = ValueList(GetParkInfo.ParkName)>
<CFSET deleted_element = ListGetAt(temp, "3", ",")>
<cfoutput><p>The original list: #temp#</p></cfoutput>
<!--- Delete the third element from the list. --->
<CFSET temp2 = ListDeleteAt(Temp, "3")>
<cfoutput>
<p>The changed list: #temp2#
<p><I>This list element:<br>#deleted_element#<br> is no longer present
    at position three of the list.</I> </cfoutput>
```

ListFind

Description
Determines the index of the first list element in which a specified value occurs. Case-sensitive.

Returns
Index of the first list element that contains *value*, with matching case. If not found, returns zero. The search is case-sensitive.

Category
List functions

Function syntax
`ListFind(list, value [, delimiters ])`

See also
`ListContains`, `ListFindNoCase`; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`list`	A list or a variable that contains one
`value`	A string, a number, or a variable that contains one. Item for which to search. The search is case-sensitive.
`delimiters`	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<!--- Uses ListFind and ListFindNoCase to see if a substring exists
    in a list --->
<form action="./listfind.cfm" method="POST">
    <p>Try changing the case in Leary's last name:
    <br><input type="Text" size="25" name="myString" value="Leary">
    <p>Pick a search type:
        <select name="type">
            <option value="ListFind" selected>Case-Sensitive
            <option value="ListFindNoCase">Case-Insensitive
        </select>
    <input type="Submit" name="" value="Search Employee List">
</form>

<!--- wait to have a string for searching defined --->
<cfif IsDefined("form.myString") and IsDefined("form.type")>

<cfquery name="SearchEmpLastName" datasource="cfdocexamples">
    SELECT FirstName, RTrim(LastName) AS LName, Phone, Department
    FROM Employees
</cfquery>

<cfset myList = ValueList(SearchEmpLastName.LName)>
<!--- Is this case-sensitive or case-insensitive searching --->
```

```
<cfif form.type is "ListFind">
    <cfset temp = ListFind(myList, form.myString)>
        <cfif temp is 0>
            <h3>An employee with that exact last name was not found</h3>
        <cfelse>
            <cfoutput>
            <p>Employee #ListGetAt(ValueList(SearchEmpLastName.FirstName), temp)#
            #ListGetAt(ValueList(SearchEmpLastName.LName), temp)#, of the
            #ListGetAt(ValueList(SearchEmpLastName.Department), temp)# Department,
            can be reached at #ListGetAt(ValueList(SearchEmpLastName.Phone),
            temp)#.
            <p>This was the first employee found under this case-sensitive last name
            search.
            </cfoutput>
        </cfif>
        <cfelse>
            <cfset temp = ListFindNoCase(myList, form.myString)>
            <cfif temp is 0>
                <h3>An employee with that exact last name was not found</h3>
            <cfelse>
                <cfoutput>
                <p>Employee #ListGetAt(ValueList(SearchEmpLastName.FirstName), temp)#
                #ListGetAt(ValueList(SearchEmpLastName.LName), temp)#, of the
                #ListGetAt(ValueList(SearchEmpLastName.Department), temp)#
                Department, can be reached at
                #ListGetAt(ValueList(SearchEmpLastName.Phone), temp)#.
                <p>This was the first employee found under this case-insensitive last
                name search.
            </cfoutput>
        </cfif>
    </cfif>
</cfif>
```

ListFindNoCase

Description
Determines the index of the first list element in which a specified value occurs.

Returns
Index of the first list element that contains *value*. If not found, returns zero. The search is case-insensitive.

Category
List functions

Function syntax
```
ListFindNoCase(list, value [, delimiters ])
```

See also
ListContains, ListFind; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
value	Number or string for which to search. The search is case-insensitive.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<!--- Uses ListFind and ListFindNoCase to see if a substring exists
    in a list --->
<form action="./listfind.cfm" method="POST">
    <p>Try changing the case in Leary's last name:
    <br><input type="Text" size="25" name="myString" value="Leary">
    <p>Pick a search type:
        <select name="type">
            <option value="ListFind" selected>Case-Sensitive
            <option value="ListFindNoCase">Case-Insensitive
        </select>
    <input type="Submit" name="" value="Search Employee List">
</form>

<!--- wait to have a string for searching defined --->
<cfif IsDefined("form.myString") and IsDefined("form.type")>

<cfquery name="SearchEmpLastName" datasource="cfdocexamples">
    SELECT FirstName, RTrim(LastName) AS LName, Phone, Department
    FROM Employees
</cfquery>

<cfset myList = ValueList(SearchEmpLastName.LName)>
<!--- Is this case-sensitive or case-insensitive searching --->
<cfif form.type is "ListFind">
```

```
    <cfset temp = ListFind(myList, form.myString)>
        <cfif temp is 0>
            <h3>An employee with that exact last name was not found</h3>
        <cfelse>
            <cfoutput>
            <p>Employee #ListGetAt(ValueList(SearchEmpLastName.FirstName), temp)#
            #ListGetAt(ValueList(SearchEmpLastName.LName), temp)#, of the
            #ListGetAt(ValueList(SearchEmpLastName.Department), temp)# Department,
            can be reached at #ListGetAt(ValueList(SearchEmpLastName.Phone),
            temp)#.
            <p>This was the first employee found under this case-sensitive last name
            search.
            </cfoutput>
        </cfif>
        <cfelse>
            <cfset temp = ListFindNoCase(myList, form.myString)>
            <cfif temp is 0>
                <h3>An employee with that exact last name was not found</h3>
            <cfelse>
                <cfoutput>
                <p>Employee #ListGetAt(ValueList(SearchEmpLastName.FirstName), temp)#
                #ListGetAt(ValueList(SearchEmpLastName.LName), temp)#, of the
                #ListGetAt(ValueList(SearchEmpLastName.Department), temp)#
                Department, can be reached at
                #ListGetAt(ValueList(SearchEmpLastName.Phone), temp)#.
                <p>This was the first employee found under this case-insensitive last
                name search.
            </cfoutput>
        </cfif>
    </cfif>
</cfif>
```

ListFirst

Description
Gets the first element of a list.

Returns
The first element of a list. If the list is empty, returns an empty string.

Category
List functions

Function syntax
```
ListFirst(list [, delimiters ])
```

See also
ListGetAt, ListLast, ListQualify; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains a list.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<h3>ListFirst Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
    SELECT Username, Subject, Posted
    FROMMessages
</cfquery>

<cfset temp = ValueList(GetMessageUser.Username)>
<p>Before editing the list, it is: 
<cfoutput>#ValueList(GetMessageUser.Username)#</cfoutput>.
<p>(Users who posted more than once are listed more than once.)
<!--- Show the first user in the list --->
<p>The first user in the list is:
<cfoutput>#ListFirst(temp)#</cfoutput>
<p>The rest of the list is: <cfoutput>#ListRest(temp)#</cfoutput>.
<p>(Users who posted more than once are listed more than once.)
<p>The last user in the list is: <cfoutput>#ListLast(temp)#</cfoutput>
```

ListGetAt

Description
Gets a list element at a specified position.

Returns
Value of the list element at position *position*.

Category
List functions

Function syntax
```
ListGetAt(list, position [, delimiters ])
```

See also
ListFirst, ListLast, ListQualify, ListSetAt; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
position	A positive integer or a variable that contains one. Position at which to get element. The first list position is 1.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
If you use list functions on strings that are delimited by a delimiter character and a space, a returned list element might contain a leading space; you use the `trim` function to remove such spaces from a returned element. For example, consider this list:

```
<cfset myList = "one hundred, two hundred, three hundred">
```

To get a value from this list, use the `trim` function to remove the space before the returned value:

```
<cfset MyValue = #trim(listGetAt(myList, 2))#>
```

With this usage, `MyValue` = "`two hundred`", not " `two hundred`", and spaces within a list element are preserved.

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<h3>ListGetAt Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
    SELECT Username, Subject, Posted
    FROMMessages
</cfquery>
<cfset temp = ValueList(GetMessageUser.Username)>
<!--- loop through the list and show it with ListGetAt --->
<h3>This list of usernames who have posted messages numbers
<cfoutput>#ListLen(temp)#</cfoutput> users.</h3>
<ul>
<cfloop From = "1" To = "#ListLen(temp)#" index = "Counter">
    <cfoutput><li>Username #Counter#: #ListGetAt(temp, Counter)# </cfoutput>
```

```
</cfloop>
</ul>
```

ListInsertAt

Description
Inserts an element in a list.

Returns
A copy of the list, with *value* inserted at the specified position.

Category
List functions

Function syntax
```
ListInsertAt(list, position, value [, delimiters ])
```

See also
ListDeleteAt, ListAppend, ListPrepend, ListSetAt; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
position	A positive integer or a variable that contains one. Position at which to insert element. The first list position is 1.
value	An element or a list of elements.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
When inserting an element, ColdFusion inserts a delimiter. If delimiters contains more than one delimiter, ColdFusion uses the first delimiter in the string; if delimiters is omitted, ColdFusion uses a comma.

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<!--- This example shows ListInsertAt --->
<cfquery name = "GetParkInfo" datasource = "cfdocexamples">
SELECT PARKNAME,CITY,STATE
FROM PARKS
WHERE PARKNAME LIKE 'DE%'
</cfquery>
<cfset temp = ValueList(GetParkInfo.ParkName)>
<cfset insert_at_this_element = ListGetAt(temp, "3", ",")>
<cfoutput>
<p>The original list: #temp#
</cfoutput>
<cfset temp2 = ListInsertAt(Temp, "3", "my Inserted Value")>
```

ListLast

Description
Gets the last element of a list.

Returns
The last element of the list.

Category
List functions

Function syntax
```
ListLast(list [, delimiters ])
```

See also
ListGetAt, ListFirst; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains a list.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter; you cannot specify a multicharacter delimiter.

Usage
If you use list functions on strings that separated by a delimiter character and a space, a returned list element might contain a leading space; use the trim function to remove leading and trailing spaces from a returned element. For example, consider this list:

```
<cfset myList = "one hundred, two hundred, three hundred">
```

To get a value from this list, use the trim function to remove the space before the returned value:

```
<cfset MyValue = #trim(ListLast(myList)#>
```

With this usage, the MyValue variable gets the value "three hundred", not " three hundred", and spaces within a list element are preserved.

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<h3>ListFirst, ListLast, and ListRest Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
    SELECT Username, Subject, Posted
    FROMMessages
</cfquery>
<cfset temp = ValueList(GetMessageUser.Username)>
<p>Before editing the list, it is: 
<cfoutput>#ValueList(GetMessageUser.Username)#</cfoutput>.
<p>(Users who posted more than once are listed more than once.)
<!--- Show the first user in the list --->
<p>The first user in the list is: <cfoutput>#ListFirst(temp)#</cfoutput>
<p>The rest of the list is: <cfoutput>#ListRest(temp)#</cfoutput>.
```

```
<p>(Users who posted more than once are listed more than once.)
<p>The last user in the list is: <cfoutput>#ListLast(temp)#</cfoutput>
```

ListLen

Description

Determines the number of elements in a list.

Integer; the number of elements in a list.

Category

List functions

Function syntax

```
ListLen(list [, delimiters ])
```

See also

ListAppend, ListDeleteAt, ListInsertAt, ListPrepend; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Here are some examples of ListLen processing:

Statement	Output	Comment
ListLen('a,b, c,,,d')	4	Third element is " c"
ListLen('a,b, c,,,d',',')	4	Fourth element is "d'"
ListLen('elem_1___elem_2___elem_3')	1	
ListLen('elem*1***elem*2***elem*3')	1	
ListLen('elem_1___elem_2___elem_3','_')	6	

Example

```
<h3>ListLen Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
    SELECT Username, Subject, Posted
    FROMMessages
</cfquery>
<cfset temp = ValueList(GetMessageUser.Username)>
<!--- loop through the list and show it with ListGetAt --->
<h3>This is a list of usernames who have posted messages
<cfoutput>#ListLen(temp)#</cfoutput> users.</h3>
<ul>
<cfloop From = "1" TO = "#ListLen(temp)#" INDEX = "Counter">
    <cfoutput><li>Username #Counter#:
     #ListGetAt(temp, Counter)#</cfoutput>
```

```
</cfloop>
</ul>
```

ListPrepend

Description
Inserts an element at the beginning of a list.

Returns
A copy of the list, with *value* inserted at the first position.

Category
List functions

Function syntax
`ListPrepend(list, value [, delimiters ])`

See also
`ListAppend`, `ListInsertAt`, `ListSetAt`; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`list`	A list or a variable that contains one.
`value`	An element or a list of elements.
`delimiters`	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion only uses the first character and ignores the others.

Usage
When prepending an element to a list, ColdFusion inserts a delimiter. If `delimiters` *contains more* than one delimiter character, ColdFusion uses the first delimiter in the string; if `delimiters` is omitted, ColdFusion uses a comma.

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

If the `delimiters` parameter is the empty string (""), ColdFusion returns the contents of the `value` parameter.

Example
```
<!--- This example shows ListPrepend --->
<cfquery name = "GetParkInfo" datasource = "cfdocexamples">
    SELECT PARKNAME,CITY,STATE
    FROM PARKS
    WHERE PARKNAME LIKE 'DE%'
</cfquery>
<cfset temp = ValueList(GetParkInfo.ParkName)>
<cfset first_element = ListFirst(temp)>
<cfoutput><p>The original list: #temp#</cfoutput>
<!--- now, insert an element at position 1--->
<cfset temp2 = ListPrepend(Temp, "my Inserted Value")>
```

ListQualify

Description
Inserts a string at the beginning and end of list elements.

Returns
A copy of the list, with *qualifier* before and after the specified element(s).

Category
List functions

Function syntax
`ListQualify(list, qualifier [, delimiters, elements ])`

See also
"Lists" on page 28 in "Using ColdFusion Variables" on page 24 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: as the `elements` parameter value, you must specify "`all`" or "`char`"; otherwise, ColdFusion throws an exception. (In earlier releases, the function ignored an invalid value, and used "`all`"; this was inconsistent with other functions.)

Parameters

Parameter	Description
`list`	A list or a variable that contains one.
`qualifier`	A string or a variable that contains one. Character or string to insert before and after the list elements specified in the `elements` parameter.
`delimiters`	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion uses the first character as the delimiter and ignores the remaining characters.
`elements`	• all: all elements • char: elements that are composed of alphabetic characters

Usage
The new list might not preserve all of the delimiters in the list.

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
SELECT FirstName, LastName
FROM Employees
</cfquery>

<h3>ListQualify Example</h3>
<p>This example uses ListQualify to put the full names of the
    employees in the query within quotation marks.</p>
<cfset myArray = ArrayNew(1)>

<!--- loop through query; append these names successively
    to the last element --->
```

```
<cfloop query = "GetEmployeeNames">
    <cfset temp = ArrayAppend(myArray, "#FirstName# #LastName#")>
</cfloop>

<!--- sort that array descending alphabetically --->
<cfset myAlphaArray = ArraySort(myArray, "textnocase")>

<!--- show the resulting array as a list --->
<cfset myList = ArrayToList(myArray, ",")>

<cfoutput>
    <p>The contents of the unqualified list are as follows:</p>
    #myList#
</cfoutput>

<!--- show the resulting alphabetized array as a qualified list with
    single quotation marks around each full name.--->
<cfset qualifiedList1 = ListQualify(myList,"'",",","CHAR")>

<!--- output the array as a list --->
<cfoutput>
    <p>The contents of the qualified list are as follows:</p>
    <p>#qualifiedList1#</p>
</cfoutput>

<!--- show the resulting alphabetized array as a qualified list with quotation
    marks around each full name. We use &quot; to denote quotation marks
    because the quotation mark character is a control character. --->
<cfset qualifiedList2 = ListQualify(myList,"&quot;",",","CHAR")>

<!--- output the array as a list --->
<cfoutput>
    <p>The contents of the second qualified list are:</p>
    <p>#qualifiedList2#</p>
</cfoutput>
```

ListRest

Description
Gets a list, without its first element.

Returns
A copy of *list*, without the first element. If *list* has one element, returns an empty list.

Category
List functions

Function syntax
```
ListRest(list [, delimiters ])
```

See also
ListFirst, ListGetAt, ListLast; “Lists” on page 28 in the *ColdFusion Developer’s Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
If the list begins with one or more empty entries, this function drops them, as well as the first element.

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<h3>ListFirst, ListLast, and ListRest Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
    SELECT Username, Subject, Posted
    FROMMessages
</cfquery>

<cfset temp = ValueList(GetMessageUser.Username)>
<p>Before editing the list, it is: 
<cfoutput>#ValueList(GetMessageUser.Username)#</cfoutput>.
<p>(Users who posted more than once are listed more than once.)
<p>The first user in the list is:
<cfoutput>#ListFirst(temp)# </cfoutput>
<p>The rest of the list is: <cfoutput>#ListRest(temp)#</cfoutput>.
<p>(Users who posted more than once are listed more than once.)
<p>The last user in the list is: <cfoutput>#ListLast(temp)#</cfoutput>
```

ListSetAt

Description
Replaces the contents of a list element.

Returns
A copy of a list, with a new value assigned to the element at a specified position.

Category
List functions

Function syntax
```
ListSetAt(list, position, value [, delimiters ])
```

See also
ListDeleteAt, ListGetAt, ListInsertAt; "Lists" on page 28 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed delimiter modification: ColdFusion MX does not modify delimiters in the list. (In earlier releases, in some cases, replaced delimiters with the first character in the `delimiters` parameter.)

Parameters

Parameter	Description
`list`	A list or a variable that contains one.
`position`	A positive integer or a variable that contains one. Position at which to set a value. The first list position is 1.
`value`	An element or a list of elements.
`delimiters`	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Usage
When assigning an element to a list, ColdFusion inserts a delimiter. If `delimiters` contains more than one delimiter, ColdFusion uses the first delimiter in the string, or, if `delimiters` was omitted, a comma.

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example
```
<h3>ListSetAt Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
SELECT Username, Subject, Posted
FROMMessages
</cfquery>

<cfset temp = ValueList(GetMessageUser.Subject)>

<!--- loop through the list and show it with ListGetAt --->
<h3>This is a list of <cfoutput>#ListLen(temp)#</cfoutput>
subjects posted in messages.</h3>

<cfset ChangedElement = ListGetAt(temp, 2)>
```

```
<cfset TempToo = ListSetAt(temp, 2, "I changed this subject", ",")>
<ul>
<cfloop From = "1" To = "#ListLen(temptoo)#" INDEX = "Counter">
    <cfoutput><li>(#Counter#) SUBJECT: #ListGetAt(temptoo, Counter)#
    </cfoutput>
</cfloop>
</ul>
<p>Note that element 2, "<cfoutput>#changedElement#</cfoutput>",
    has been altered to "I changed this subject" using ListSetAt.
```

ListSort

Description
Sorts list elements according to a sort type and sort order.

Returns
A copy of a list, sorted.

Category
List functions

Function syntax
```
ListSort(list, sort_type [, sort_order, delimiters ])
```

See also
"Lists" on page 28 in "Using ColdFusion Variables" on page 24 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed the order in which sorted elements are returned: in a `textnocase`, descending sort, this function might return elements in a different sort order than in earlier releases. If `sort_type = "textnocase"` and `sort_order = "desc"`, ColdFusion MX processes elements that *differ only in case* differently from earlier releases. ColdFusion MX outputs the elements in the reverse of the ascending order. Earlier releases do not change order of elements that differ only in case. Both operations are correct. The new operation ensures that an ascending and descending sort output elements in exactly reverse order.

For example, in a `textnocase`, `desc` sort of `d,a,a,b,A`, the following occurs:

- ColdFusion MX returns `d,b,A,a,a`
- Earlier ColdFusion releases return `d,b,a,a,A`

(In a `textnocase`, *asc* sort, all ColdFusion releases return `a,a,A,b,d`.)

Parameters

Parameter	Description
`list`	A list or a variable that contains one.
`sort_type`	• numeric: sorts numbers • text: sorts text alphabetically, taking case into account (also known as case sensitive). All letters of one case precede the first letter of the other case: - aabzABZ, if `sort_order = "asc"` (ascending sort) - ZBAzbaa, if `sort_order = "desc"` (descending sort) • textnocase: sorts text alphabetically, without regard to case (also known as case-insensitive). A letter in varying cases precedes the next letter: - aAaBbBzzZ, in an ascending sort; preserves original intra-letter order - ZzzBbBaAa, in a descending sort; reverses original intra-letter order

Parameter	Description
sort_order	• asc - ascending sort order. Default. - aabzABZ or aAaBbBzzZ, depending on value of sort_type, for letters - from smaller to larger, for numbers • desc - descending sort order. - ZBAzbaa or ZzzBbBaAa, depending on value of sort_type, for letters - from larger to smaller, for numbers
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion uses the first character in the string as the delimiter, and ignores the rest.

Usage

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example

```
<h3>ListSort Example</h3>

<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
SELECT   Username, Subject, Posted
FROM     Messages
</cfquery>

<cfset myList = ValueList(GetMessageUser.UserName)>
<p>Here is the unsorted list. </p>
<cfoutput>#myList#
    </cfoutput>
<p>Here is the list sorted alphabetically:</p>
<cfset sortedList = ListSort(myList, "Text")>
<cfoutput>#sortedList#
    </cfoutput>

<p>Here is a numeric list that is to be sorted in descending order.</p>
<cfset sortedNums = ListSort("12,23,107,19,1,65","Numeric", "Desc")>
<cfoutput>#sortedNums# </cfoutput>

<p>Here is a list that must be sorted numerically, since it contains
    negative and positive numbers, and decimal numbers. </p>
<cfset sortedNums2 = ListSort("23.75;-34,471:100,-9745","Numeric", "ASC", ";,:")>
<cfoutput>#sortedNums2# </cfoutput>

<p>Here is a list to be sorted alphabetically without consideration of case.</p>
<cfset sortedMix =
    ListSort("hello;123,HELLO:jeans,-345,887;ColdFusion:coldfusion",
        "TextNoCase", "ASC", ";,:")>
<cfoutput>#sortedMix# </cfoutput>
```

ListToArray

Description
Copies the elements of a list to an array.

Returns
An array

Category
Array functions, Conversion functions, List functions

Function syntax
```
ListToArray(list [, delimiters, includeEmptyFields])
```

See also
ArrayToList; "Using Arrays and Structures" on page 68 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one. You define a list variable with a cfset statement.
delimiters	A string or a variable that contains one. ColdFusion treats each character in the string as a delimiter. The default value is comma.
includeEmptyFields	A Boolean value specifying whether to create empty array entries if there are two delimiters in a row. • false (Default) ignore empty elements in a list; for example, convert a,,c into an array with only two elements. • true Convert empty elements in a list to empty array entries; for example, convert a,,c into an array with three elements, the second of which is empty.

Usage
ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

ColdFusion treats each character in the *delimiters* parameter as a separate delimiter. Therefore, if the parameter is ",+" ColdFusion will break the list at *either* a comma or a plus sign.

Example
```
<h3>ListToArray Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
SELECT Username, Subject, Posted
FROMMessages
</cfquery>
<cfset myList = ValueList(GetMessageUser.UserName)>
<p>My list is a list with <cfoutput>#ListLen(myList)#</cfoutput>
 elements.
<cfset myArrayList = ListToArray(myList)>
<p>My array list is an array with <cfoutput>#ArrayLen(myArrayList)#
 </cfoutput> elements.
```

ListValueCount

Description

Counts instances of a specified value in a list. The search is case-sensitive.

Returns

The number of instances of *value* in the list.

Category

List functions, String functions

Function syntax

```
ListValueCount(list, value [, delimiters ])
```

See also

ListValueCountNoCase; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
value	String or number, or a variable that contains one. Item for which to search. The search is case-sensitive.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Example

```
<cfquery name = "SearchByDepartment" datasource = "cfdocexamples">
SELECT Department
FROM Employees
</cfquery>
<h3>ListValueCount Example</h3>
<p>This example uses ListValueCount to count employees in a department.

<form action = "listvaluecount.cfm">
<p>Select a department:</p>
    <select name = "departmentName">
            <option value = "Accounting">
                Accounting
            </OPTION>
            <option value = "Administration">
                Administration
            </OPTION>
            <option value = "Engineering">
                Engineering
            </OPTION>
            <option value = "Sales">
                Sales
            </OPTION>
    </select>
<input type = "Submit" name = "Submit" value = "Search Employee List">
</form>

<!--- wait to have a string for searching defined --->
<cfif IsDefined("FORM.Submit") and IsDefined("FORM.departmentName")>
```

```
    <cfset myList = ValueList(SearchByDepartment.Department)>
    <cfset numberInDepartment = ListValueCount(myList, FORM.departmentName)>

    <cfif numberInDepartment is 0>
        <h3>There are no employees in <cfoutput>#FORM.departmentName#</cfoutput></h3>
    <cfelseIf numberInDepartment is 1>
        <cfoutput><p>There is only one person in #FORM.departmentName#.
        </cfoutput>
    <cfelse>
        <cfoutput><p>There are #numberInDepartment# people in #FORM.departmentName#.
        </cfoutput>
    </cfif>
</cfif>
```

ListValueCountNoCase

Description

Counts instances of a specified value in a list. The search is case-insensitive.

Returns

The number of instances of *value* in the list.

Category

List functions

Function syntax

```
ListValueCountNoCase(list, value [, delimiters ])
```

See also

ListValueCount; "Lists" on page 28 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
list	A list or a variable that contains one.
value	String or number, or a variable that contains one. Item for which to search. The search is case-insensitive.
delimiters	A string or a variable that contains one. Character(s) that separate list elements. The default value is comma.
	If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.

Example

```
<cfquery name = "SearchByDepartment" datasource = "cfdocexamples">
SELECT Department
FROM Employees
</cfquery>

<h3>ListValueCountNoCase Example</h3>
<p>This example uses ListValueCountNoCase to count employees in a department.

<form action = "listvaluecountnocase.cfm">
<p>Select a department:</p>
    <select name = "departmentName">
            <option value = "Accounting">
                Accounting
            </OPTION>
            <option value = "Administration">
                Administration
            </OPTION>
            <option value = "Engineering">
                Engineering
            </OPTION>
            <option value = "Sales">
                Sales
            </OPTION>
    </select>
    </select>
<input type = "Submit" name = "Submit" value = "Search Employee List">
</form>
<!--- wait to have a string for searching defined --->
```

```
<cfif IsDefined("FORM.Submit") and IsDefined("FORM.departmentName")>
    <cfset myList = ValueList(SearchByDepartment.Department)>
    <cfset numberInDepartment = ListValueCountNoCase(myList,
        FORM.departmentName)>

    <cfif numberInDepartment is 0>
        <h3>There are no employees in <cfoutput>#FORM.departmentName#</cfoutput></h3>
    <cfelseIf numberInDepartment is 1>
        <cfoutput><p>There is only one person in #FORM.departmentName#.
        </cfoutput>
    <cfelse>
        <cfoutput><p>There are #numberInDepartment# people in #FORM.departmentName#.
        </cfoutput>
    </cfif>
</cfif>
```

LJustify

Description

Left justifies characters in a string of a specified length.

Returns

A copy of a string, left-justified.

Category

Display and formatting functions, String functions

Function syntax

`LJustify(string, length)`

See also

`CJustify`, `RJustify`

Parameters

Parameter	Description
`string`	A string or a variable that contains one
`length`	Length of field in which to justify string

Example

```
<!--- This example shows how to use LJustify --->
<cfparam name = "jstring" default = "">

<cfif IsDefined("FORM.justifyString")>
    <cfset jstring = LJustify(FORM.justifyString, 35)>
</cfif>
<html>
<head>
    <title>LJustify Example</title>
</head>
<body>

<h3>LJustify Function</h3>
<p>Enter a string, and it will be left justified within the sample field

<form action = "ljustify.cfm">
<p><input type = "Text" value = "<cfoutput>#jString#</cfoutput>"
 size = 35 name = "justifyString">

<p><input type = "Submit" name = ""> <input type = "RESET">
</form>
```

Log

Description

Calculates the natural logarithm of a number. Natural logarithms are based on the constant e (2.71828182845904).

Returns

The natural logarithm of a number.

Category

Mathematical functions

Function syntax

Log(*number*)

See also

Exp, Log10

Parameters

Parameter	Description
number	Positive real number for which to calculate the natural logarithm

Example

```
<h3>Log Example</h3>

<cfif IsDefined("FORM.number")>
    <cfoutput>
        <p>Your number, #FORM.number#
        <br>#FORM.number# raised to the E power: #exp(FORM.number)#
        <cfif FORM.number LTE 0><br>Enter a positive real number to get its
    natural logarithm
            <cfelse><br>The natural logarithm of #FORM.number#: #log(FORM.number)#
        </cfif>
        <cfif FORM.number LTE 0><br>Enter a positive real number to get its
    logarithm to base 10
            <cfelse><br>The logarithm of #FORM.number# to base 10: #log10(FORM.number)#
        </cfif>
    </cfoutput>
</cfif>
<cfform action = "log.cfm">
Enter a number to see its value raised to the E power, its natural logarithm,
    and the logarithm of number to base 10.
<cfinput type = "Text" name = "number" message = "You must enter a number"
    validate = "float" required = "No">
<input type = "Submit" name = "">
</cfform>
```

Log10

Description

Calculates the logarithm of *number*, to base 10.

Returns

Number; the logarithm of *number*, to base 10.

Category

Mathematical functions

Function syntax

`Log10(number)`

See also

`Exp`, `Log`

Parameters

Parameter	Description
`number`	Positive real number for which to calculate the logarithm

Example

```
<h3>Log10 Example</h3>
<cfif IsDefined("FORM.number")>
    <cfoutput>
        <p>Your number, #FORM.number#
        <br>#FORM.number# raised to the E power: #exp(FORM.number)#
        <cfif FORM.number LTE 0><br>You must enter a positive real number to
    see the natural logarithm of that number
        <cfelse><br>The natural logarithm of #FORM.number#: #log(FORM.number)#
        </cfif>
        <cfif #FORM.number# LTE 0><br>You must enter a positive real number to
    see the logarithm of that number to base 10
        <cfelse><br>The logarithm of #FORM.number# to base 10: #log10(FORM.number)#
        </cfif>
    </cfoutput>
</cfif>
<cfform action = "log10.cfm">
Enter a number to find its value raised to the E power, its natural
    logarithm, and the logarithm of number to base 10.
<cfinput type = "Text" name = "number" message = "You must enter a number"
    validate = "float" required = "No">
<input type = "Submit" name = "">
</cfform>
```

LSCurrencyFormat

Description

Formats a number in a locale-specific currency format. For countries that use the euro, the result depends on the JVM.

Returns

A formatted currency value.

Category

Display and formatting functions, International functions

Function syntax

`LSCurrencyFormat(number [, type, locale])`

See also

`LSEuroCurrencyFormat`, `LSIsCurrency`, `LSParseCurrency`, `LSParseEuroCurrency`, `SetLocale`; "Handling data in ColdFusion" on page 347 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. If a negative number is passed to it, it returns a negative number. If `type = "local"`, it returns the value in the current locale's standard format. If `type = "international"`, it returns the value in the current locale's international standard format. This function uses Java standard locale formatting rules on all platforms.

Parameters

Parameter	Description
`number`	Currency value
`type`	• local: the currency format and currency symbol used in the locale. - With JDK 1.3, the default for Euro Zone countries is their local currency. - With JDK 1.4, the default for Euro Zone countries is the euro. • international: the international standard currency format and currency symbol of the locale. • none: the currency format used in the locale; no currency symbol
`locale`	Locale to use instead of the locale of the page when processing the function

Usage

This function uses Java standard locale formatting rules on all platforms.

Note: *With a Sun 1.3.1-compliant JVM, use the `LSEuroCurrencyFormat` function to format euro currency values.*

Currency output

The following table shows sample currency output. For locales that use Euro, the Local and International columns contains two entries. The first is entry is the result with a Sun the 1.4.1-compliant JVM, the second entry is the result with a 1.3.1-compliant JVM.

Locale	Type = Local	Type = International	Type = None
Chinese (China)	¥100,000.00	CNY100,000.00	100,000.00
Chinese (Hong Kong)	HK$100,000.00	HKD100,000.00	100,000.00
Chinese (Taiwan)	NT$100,000.00	TWD100,000.00	100,000.00
Dutch (Belgian)	100.000,00 ¤ 100.000,00 BF	BEF100.000,00 EUR100.000,00	100.000,00
Dutch (Standard)	¤ 100.000,00 fl 100.000,00	NLG100.000,00 EUR100.000,00	100.000,00
English (Australian)	$100,000.00	AUD100,000.00	100,000.00
English (Canadian)	$100,000.00	CAD100,000.00	100,000.00
English (New Zealand)	$100,000.00	NZD100,000.00	100,000.00
English (UK)	£100,000.00	GBP100,000.00	100,000.00
English (US)	$100,000.00	USD100,000.00	100,000.00
French (Belgian)	100.000,00 ¤ 100.000,00 FB	EUR100.000,00 BEF100.000,00	100.000,00
French (Canadian)	100 000,00 $	CAD100 000,00	100 000,00
French (Standard)	100 000,00 ¤ 100 000,00 F	EUR100 000,00 FRF100 000,00	100 000,00
French (Swiss)	SFr. 100'000.00	CHF100'000.00	100'000.00
German (Austrian)	¤ 100.000,00 öS 100.000,00	EUR100.000,00 ATS100.000,00	100.000,00
German (Standard)	100.000,00 ¤ 100.000,00 DM	EUR100.000,00 DEM100.000,00	100.000,00
German (Swiss)	SFr. 100'000.00	CHF100'000.00	100'000.00
Italian (Standard)	¤ 100.000,00 L. 10.000.000	EUR10.000.000 ITL10.000.000	10.000.000
Italian (Swiss)	SFr. 100'000.00	CHF100'000.00	100'000.00
Japanese	¥100,000	JPY100,000	JPY100,000
Korean	₩100,000	KRW100,000	100,000
Norwegian (Bokmal)	kr 100 000,00	NOK100 000,00	100 000,00
Norwegian (Nynorsk)	kr 100 000,00	NOK100 000,00	100 000,00
Portuguese (Brazilian)	R$100.000,00	BRC100.000,00	100.000,00
Portuguese (Standard)	100.000,00 ¤ R$100.000,00	EUR100.000,00 BRC100.000,00	100.000,00
Spanish (Mexican)	$100,000.00	MXN100,000.00	100,000.00

Locale	Type = Local(Continued)	Type = International	Type = None
Spanish (Modern)	100.000,00 ¤ 10.000.000 Pts	EUR10.000.000 ESP10.000.000	10.000.000
Spanish (Standard)	100.000,00 ¤ 10.000.000 Pts	ESP10.000.000 EUR10.000.000	10.000.000
Swedish	100.000,00 kr	SEK100.000,00	100.000,00

Note: *ColdFusion maps Spanish (Modern) to the Spanish (Standard) format.*

To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

Example

```
<h3>LSCurrencyFormat Example</h3>
<p>LSCurrencyFormat returns a currency value using the locale
    convention. Default value is "local."
<!--- loop through list of locales; show currency values for 100,000 units --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><b><I>#locale#</I></b><br>
        Local: #LSCurrencyFormat(100000, "local")#<br>
        International: #LSCurrencyFormat(100000, "international")#<br>
        None: #LSCurrencyFormat(100000, "none")#<br>
        <hr noshade>
    </cfoutput>
</cfloop>
```

LSDateFormat

Description

Formats the date part of a date/time value in a locale-specific format.

Returns

A formatted date/time value. If no mask is specified, the value is formatted according to the locale setting of the client computer.

Category

Date and time functions, Display and formatting functions, International functions

Function syntax

`LSDateFormat(date [, mask, locale])`

See also

`LSParseDateTime`, `LSTimeFormat`, `DateFormat`, `SetLocale`; "Handling data in ColdFusion" on page 347 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX:

- Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.
- Added support for the following *mask* parameter options: `short`, `medium`, `long`, and `full`.

Parameters

Parameter	Description
date	A date/time object, in the range 100 AD–9999 AD.
mask	Characters that show how ColdFusion displays the date: • d: Day of month. Digits; no leading zero for single-digit days • dd: Day of month. Digits; leading zero for single-digit days • ddd: Day of week, abbreviation • dddd: Day of week. Full name • m: Month. Digits; no leading zero for single-digit months • mm: Month. Digits; leading zero for single-digit months • mmm: Month. abbreviation (if appropriate) • mmmm: Month. Full name • y: Year. Last two digits; no leading zero for years less than 10 • yy: Year. Last two digits; leading zero for years less than 10 • yyyy: Year. Four digits • gg: Period/era string. Not processed. Reserved for future use The following conform to Java locale-specific time encoding standards. Their exact formats depend on the locale: • short: dd, mm, and yy separated by / marks • medium: text format using mmm, d, and yyyy • long: text format using mmmm, d, and yyyy • full: text format using dddd, mmmm, d, and yyyy The default value is medium For more information on formats, see LSParseDateTime.
locale	Locale to use instead of the locale of the page when processing the function

Usage

This function uses Java standard locale formatting rules on all platforms.

When passing date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

To calculate a difference between time zones, use the GetTimeZoneInfo function.

Example

```
<h3>LSDateFormat Example</h3>
<p>LSDateFormat formats the date part of a date/time value using the
    locale convention.
<!--- loop through a list of locales; show date values for Now()--->
<cfloop list = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>

    <cfoutput><p><B><I>#locale#</I></B><br>
        #LSDateFormat(Now(), "mmm-dd-yyyy")#<br>
        #LSDateFormat(Now(), "mmmm d, yyyy")#<br>
        #LSDateFormat(Now(), "mm/dd/yyyy")#<br>
        #LSDateFormat(Now(), "d-mmm-yyyy")#<br>
```

```
        #LSDateFormat(Now(), "ddd, mmmm dd, yyyy")#<br>
        #LSDateFormat(Now(), "d/m/yy")#<br>
        #LSDateFormat(Now())#<br>
        <hr noshade>
    </cfoutput>
</cfloop>
```

LSEuroCurrencyFormat

Description
Formats a number in a locale-specific currency format.

Returns
A formatted currency value. For countries in the Euro currency zone, the function uses the locale's rule's for formatting currency in euros.

Category
Display and formatting functions, International functions

Function syntax
`LSEuroCurrencyFormat(currency-number [, type, locale])`

See also
`LSParseEuroCurrency`, `LSCurrencyFormat`, `SetLocale`; "Locale-specific content" on page 348 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java locale formatting rules on all platforms, except that it uses the rule detailed in the Usage section for countries in the Euro currency zone. As a result, it format currencies for non Euro zone locales using the country's currency, not euros.

Parameters

Parameter	Description
`currency-number`	Currency value.
`locale`	Locale to use instead of the locale of the page when processing the function
`type`	• local: the currency format used in the locale. (Default.) • international: the international standard currency format of the locale. For example, EUR10.00 • none: the currency format used in the locale; no currency symbol

Usage
This function uses euro currency formatting rules for all JVM versions, as follows:

• If the country of the current locale belongs to the Euro Zone (whose members have converted to the euro) the formatted output for the local type includes the Euro currency sign (€); for the international type, the output includes the euro currency symbol (EUR). If the value is negative, the format includes a negative sign before the value or parentheses around the value, according to the formatting rules of the current locale.

• If the country of the current locale is not in the Euro Zone, the currency sign or symbol of the current locale displays. If the value is negative, the format includes a negative sign before the value or parentheses around the value, according to the formatting rules of the current locale.

For a list of the locale options that ColdFusion supports, and information on setting the default display format of date, time, number, and currency values, see "SetLocale" on page 1154.

Currency output

The following table shows examples of currency output:

Locale	Type = Local	Type = International	Type = None
Chinese (China)	¥100,000.00	CNY100,000.00	100,000.00
Chinese (Hong Kong)	HK$100,000.00	HKD100,000.00	100,000.00
Chinese (Taiwan)	NT$100,000.00	TWD100,000.00	100,000.00
Dutch (Belgian)	100.000,00 ¤	EUR100.000,00	100.000,00
Dutch (Standard)	¤ 100.000,00	EUR100.000,00	100.000,00
English (Australian)	$100,000.00	AUD100,000.00	100,000.00
English (Canadian)	$100,000.00	CAD100,000.00	100,000.00
English (New Zealand)	$100,000.00	NZD100,000.00	100,000.00
English (UK)	£100,000.00	GBP100,000.00	100,000.00
English (US)	$100,000.00	USD100,000.00	100,000.00
French (Belgian)	100.000,00 ¤	EUR100.000,00	100.000,00
French (Canadian)	100 000,00 $	CAD100 000,00	100 000,00
French (Standard)	100 000,00 ¤	EUR100 000,00	100 000,00
French (Swiss)	SFr. 100'000.00	CHF100'000.00	100'000.00
German (Austrian)	¤ 100.000,00	EUR100.000,00	100.000,00
German (Standard)	100.000,00 ¤	EUR100.000,00	100.000,00
German (Swiss)	SFr. 100'000.00	CHF100'000.00	100'000.00
Italian (Standard)	¤ 100.000,00	EUR10.000.000	10.000.000
Italian (Swiss)	SFr. 100'000.00	CHF100'000.00	100'000.00
Japanese	¥100,000	JPY100,000	JPY100,000
Korean	₩100,000	KRW100,000	100,000
Norwegian (Bokmal)	kr 100 000,00	NOK100 000,00	100 000,00
Norwegian (Nynorsk)	kr 100 000,00	NOK100 000,00	100 000,00
Portuguese (Brazilian)	R$100.000,00	BRC100.000,00	100.000,00
Portuguese (Standard)	100.000,00 ¤	EUR100.000,00	100.000,00
Spanish (Mexican)	$100,000.00	MXN100,000.00	100,000.00
Spanish (Modern)	100.000,00 ¤	EUR10.000.000	10.000.000
Spanish (Standard)	100.000,00 ¤	ESP10.000.000	10.000.000
Swedish	100.000,00 kr	SEK100.000,00	100.000,00

Note: *ColdFusion uses the Spanish (Standard) formats for Spanish (Modern) and Spanish (Standard).*

The following example shows how the function formats negative values. The format includes a negative sign before the value, or parentheses around the value, according to the formatting rules of the current locale.

Input value	Output if locale = French (Standard)	Output if locale = English (US)
-1234.56	-1 234,56 ¤	($1,234.56)

Example

```
<h3>LSEuroCurrencyFormat Example</h3>
<p>LSEuroCurrencyFormat returns a currency value using the locale
    convention. Default value is "local."
<!--- Loop through list of locales, show currency values for 100,000 units --->
<cfloop list = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><B><I>#locale#</I></B><br>
        Local: #LSEuroCurrencyFormat(100000, "local")#<br>
        International: #LSEuroCurrencyFormat(100000, "international")#<br>
        None: #LSEuroCurrencyFormat(100000, "none")#<br>
        <Hr noshade>
    </cfoutput>
</cfloop>
```

LSIsCurrency

Description

Determines whether a string is a valid representation of a currency amount in the current locale.

Returns

True, if the parameter is formatted as a valid currency amount, including the appropriate currency indicator. The return value is True for amounts in the local, international, or none currency formats.

Category

Display and formatting functions, Decision functions, International functions

Function syntax

`LSIsCurrency(string [, locale])`

See also

`GetLocale`, `SetLocale`, `LSCurrencyFormat`

History

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return a different result than in earlier releases. This function uses Java standard locale formatting rules on all platforms; the results might vary depending upon the JVM; for example, Sun JVM 1.4.1 requires euro format the local currency if the current locale's country belongs to the Euro Zone.

Parameters

Parameter	Description
`string`	A currency string or a variable that contains one.
`locale`	Locale to use instead of the locale of the page when processing the function

Usage

For examples of ColdFusion code and output that shows differences between earlier ColdFusion releases and ColdFusion MX in accepting input formats and displaying output, see `LSCurrencyFormat` .

Note: *If the locale belongs to a Euro zone country and the currency is a correctly formatted euro value for the locale, this function returns True for all JVMs, including Sun 1.3.1. As a result, with 1.3.1-compliant JVMs, the LSIsCurrency function does not ensure that* `LSParseCurrency` *returns a value. If a currency uses the older country-specific format for Euro Zone locales, the LSIsCurrency function returns False for newer JVMs, such as Sun 1.4.1, and True for older JVMs, such as Sun 1.3.1.*

Note: *To set the default display format of date, time, number, and currency values, use the* `SetLocale` *function.*

Example

```
<h3>LSIsCurrency Example</h3>

<cfif IsDefined("FORM.locale")>
<!--- if locale is defined, set locale to that entry --->
<cfset NewLocale = SetLocale(FORM.locale)>

<p>Is the value "<cfoutput>#FORM.myValue#</cfoutput>"
    a proper currency value for <cfoutput>#GetLocale()#</cfoutput>?
```

```
<p>Answer: <cfoutput>#LSIsCurrency(FORM.myValue)#</cfoutput>
</cfif>

<p><form action = "LSIsCurrency.cfm">
<p>Select a locale for which you would like to check a currency value:
<!--- check the current locale for server --->
<cfset serverLocale = GetLocale()>
```

LSIsDate

Description
Determines whether a string is a valid representation of a date/time value in the current locale.

Returns
True, if the string can be formatted as a date/time value in the current locale; False, otherwise.

Category
Date and time functions, Display and formatting functions, International functions

Function syntax
`LSIsDate(string [, locale])`

See also
`CreateDateTime`, `GetLocale`, `IsNumericDate`, `LSDateFormat`, `ParseDateTime`, `SetLocale`; "Handling data in ColdFusion" on page 347 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Added the `locale` parameter.

ColdFusion MX:

- Changed formatting behavior: this function might return a different result than in earlier releases. This function uses Java standard locale formatting rules on all platforms.
- Changed behavior: this function accepts a dash or hyphen character only in the Dutch(Standard) and Portuguese (Standard) locales. If called this way (for example, LsIsDate("3-1-2002") in any other locale, this function returns False. (Earlier releases returned True.)
- Changed behavior: when using the SUN JRE 1.3.1 on an English(UK) locale, this function returns False for a date that has a one-digit month or day (for example, 1/1/01). To work around this, insert a zero in a one-digit month or day (for example, 01/01/01).

Parameters

Parameter	Description
`string`	A string or a variable that contains one
`locale`	Locale to use instead of the locale of the page when processing the function

Usage
A date/time object is in the range 100 AD–9999 AD.

To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

Example
```
<h3>LSIsDate Example</h3>
<cfif IsDefined("FORM.locale")>
    <!--- if locale is defined, set locale to that entry --->
    <cfset NewLocale = SetLocale(FORM.locale)>
    <p>Is the value "<cfoutput>#FORM.myValue#</cFOUTPUT>" a proper date
    value for <cfoutput>#GetLocale()#</cfoutput>?
    <p>Answer: <cfoutput>#LSIsDate(FORM.myValue)#</cfoutput>
</cfif>
```

```
<p><form action = "LSIsDate.cfm">
<p>Select a locale for which you would like to check a date value:
<!--- check the current locale for server --->
<cfset serverLocale = GetLocale()>
```

LSIsNumeric

Description
Determines whether a string is a valid representation of a number in the current locale.

Returns
True, if the string represents a number the current locale; False, otherwise.

Category
Decision functions, International functions, String functions

Function syntax
`LSIsNumeric(string [, locale])`

See also
`GetLocale`, `SetLocale`; "Handling data in ColdFusion" on page 347 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return a different result than in earlier releases. This function uses Java standard locale formatting rules on all platforms.

Parameters

Parameter	Description
`string`	A string or a variable that contains one
`locale`	Locale to use instead of the locale of the page when processing the function

Usage
To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

Example
```
<h3>LSIsNumeric Example</h3>

<cfif IsDefined("FORM.locale")>
<!--- if locale is defined, set locale to that entry --->
<cfset NewLocale = SetLocale(FORM.locale)>

<p>Is the value "<cfoutput>#FORM.myValue#</cFOUTPUT>"
a proper numeric value for <cfoutput>#GetLocale()#</cfoutput>?

<p>Answer: <cfoutput>#LSIsNumeric(FORM.myValue)#</cfoutput>
</cfif>

<p><form action = "LSIsNumeric.cfm">

<p>Select a locale for which to check a numeric value:
...
```

LSNumberFormat

Description

Formats a number in a locale-specific format.

Returns

A formatted number.

- If no mask is specified, it returns the number formatted as an integer
- If no mask is specified, truncates the decimal part; for example, it truncates 34.57 to 35
- If the specified mask cannot correctly mask a number, it returns the number unchanged
- If the parameter value is "" (an empty string), it returns 0.

Category

Display and formatting functions, International functions

Function syntax

`LSNumberFormat(number [, mask, locale])`

See also

`GetLocale`, `SetLocale`; "Handling data in ColdFusion" on page 347 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX:

- Changed behavior: if the specified mask format cannot correctly mask a number, this function returns the number unchanged. (In earlier releases, it truncated the number or threw an error.) (If no mask is specified, ColdFusion MX truncates the decimal part as ColdFusion 5 does. For example, it truncates 1234.567 to 1235.)
- Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.

Parameters

Parameter	Description
`number`	Number to format
`mask`	`LSNumberFormat` mask characters apply, except: dollar sign, comma, and dot are mapped to their locale-specific equivalents.
`locale`	Locale to use instead of the locale of the page when processing the function

The following table lists the `LSNumberFormat` mask characters:

Character	Meaning
_	(Underscore.) Digit placeholder.
9	Digit placeholder. (Shows decimal places more clearly than _ .)
.	Location of a mandatory decimal point (or locale-appropriate symbol).
0	Located to the left or right of a mandatory decimal point. Pads with zeros.

Character	Meaning
( )	If number is less than zero, puts parentheses around the mask.
+	Puts plus sign before positive number; minus sign before negative number.
-	Puts space before positive number; minus sign before negative number.
,	Separates every third decimal place with a comma (or locale-appropriate symbol).
L,C	Left-justifies or center-justifies number within width of mask column. First character of mask must be L or C. The default value is right-justified.
$	Puts a dollar sign (or locale-appropriate symbol) before formatted number. First character of mask must be the dollar sign ($).
^	Separates left and right formatting.

Note: *If you do not specify a sign for the mask, positive and negative numbers do not align in columns. To put a plus sign or space before positive numbers and a minus sign before negative numbers, use the plus or hyphen mask character, respectively.*

Usage

This function uses Java standard locale formatting rules on all platforms.

The position of symbols in format masks determines where the codes take effect. For example, if you put a dollar sign at the far left of a format mask, ColdFusion displays a dollar sign at the left edge of the formatted number. If you separate the dollar sign on the left edge of the format mask by at least one underscore, ColdFusion displays the dollar sign just to the left of the digits in the formatted number.

These examples show how symbols determine formats:

Number	Mask	Result
4.37	$____.__	"$ 4.37"
4.37	_$___.__	" $4.37"

The positioning can also show where to put a minus sign for negative numbers:

Number	Mask	Result
-4.37	-____.__	"- 4.37"
-4.37	_-___.__	" -4.37"

The positions for a symbol are: far left, near left, near right, and far right. The left and right positions are determined by the side of the decimal point on which the code character is shown. For formats that do not have a fixed number of decimal places, you can use a caret (^) to separate the left fields from the right.

An underscore determines whether the code is placed in the far or near position. Most code characters' effect is determined by the field in which they are located. This example shows how to specify where to put parentheses to display negative numbers:

Number	Mask	Result
3.21	C(__^__)	"(3.21)"
3.21	C__(^__)	" (3.21)"
3.21	C(__^)__	"(3.21) "
3.21	C__(^)__	" (3.21) "

To set the default display format of date, time, number, and currency values, use the SetLocale function.

When converting from string to double, to prevent rounding errors, this function adds a rounding factor of 1.5543122344752E-014 to the converted number. For example, without adding the rounding factor, converting the string value 1.275 to double with two digits of precision results in a value of 1.27499999999999999, which would be rounded up to 1.27. By adding the rounding factor, the conversion correctly results in a value of 1.28.

If you round off a double, such as 1.99499999999999999999999999999, where the last decimal is 10E-14, the rounding factor can cause an incorrect result.

Example

```
<h3>LSNumberFormat Example</h3>
<p>LSNumberFormat returns a number value using the locale convention.
<!--- loop through a list of locales and show number values --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><b><i>#locale#</i></b><br>
        #LSNumberFormat(-1234.5678, "__________")#<br>
        #LSNumberFormat(-1234.5678, "__________.___")#<br>
        #LSNumberFormat(1234.5678, "__________")#<br>
        #LSNumberFormat(1234.5678, "__________.___")#<br>
        #LSNumberFormat(1234.5678, "$_(__________.___)")#<br>
        #LSNumberFormat(-1234.5678, "$_(__________.___)")#<br>
        #LSNumberFormat(1234.5678, "+__________.___")#<br>
        #LSNumberFormat(1234.5678, "-__________.___")#<br>
    </cfoutput>
</cfloop>
```

LSParseCurrency

Description

Converts a locale-specific currency string into a formatted number. Attempts conversion by comparing the string with each the three supported currency formats (none, local, international) and using the first that matches.

Returns

A formatted number (string representation of a number) that matches the value of the parameter.

Category

International functions, String functions

Function syntax

`LSParseCurrency(string [, locale])`

See also

`LSParseEuroCurrency`, `LSCurrencyFormat`, `LSEuroCurrencyFormat`, `LSIsCurrency`; "Locale-specific content" on page 348 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.

Parameters

Parameter	Description
`string`	A locale-specific string a variable that contains one
`locale`	Locale to use instead of the locale of the page when processing the function

Usage

This function uses the locale formatting rules of the JVM (specified in the ColdFusion Administrator Java and JVM page) on all platforms. These rules changed between Sun JVM 1.3.1 and JVM 1.4.1:

- JVM 1.3.1 requires that the local and international versions of currencies of countries in the Euro zone be formatted using the older, country-specific designations, such as 100.000,00 DM or DEM100.000,00 for the German (Standard) locale. Use the `LSParseEuroCurrency` function to parse euro currencies in these locales with JVM 1.3.1.
- JVM 1.4.1 requires that currencies for Euro zone countries be expressed as euros; for example 100.000,00 € or EUR100.000,00.

Note: *The `LSIsCurrency` function always returns True if the locale is in the Euro currency zone and the currency is expressed in euros, including when using JVM 1.3.1. As a result, with older JVMs, `LSIsCurrency` does not ensure that `LSParseCurrency` returns a value.*

To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

For a list of the locale-specific formats used to parse the currency, see `LSCurrencyFormat`.

Example

```
<h3>LSParseCurrency Example</h3>
<p>LSParseCurrency coverts a locale-specific currency string to a number.
```

```
    Attempts conversion through each of the three default currency formats.
<!--- loop through a list of locales; show currency values for 123,456 units --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><B><I>#locale#</I></B><br>
        Local: #LSCurrencyFormat(123456.78, "local")#<br>
        Parsed local Currency:
    #LSParseCurrency(LSCurrencyFormat(123456,"local"))#<br>
        International: #LSCurrencyFormat(123456.78999, "international")#<br>
        Parsed International Currency:
    #LSParseCurrency(LSCurrencyFormat(123456.78999,"international"))#<br>
        None: #LSCurrencyFormat(123456.78999, "none")#<br>
        Parsed None formatted currency:
    #LSParseCurrency(LSCurrencyFormat(123456.78999,"none"))#<br>
        <hr noshade>
    </cfoutput>
</cfloop>
```

LSParseDateTime

Description
Converts a string that is a valid date/time representation in the current locale into a date/time object.

Returns
A date/time object.

Category
Date and time functions, Display and formatting functions, International functions, String functions

Function syntax
`LSParseDateTime(date/time-string [, locale])`

See also
`LSDateFormat`, `ParseDateTime`, `SetLocale`, `GetLocale`; "Locales" on page 341 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Added the `locale` parameter.

ColdFusion MX:

- Changed formatting behavior: this function might not parse string formats that worked with earlier releases. This function uses Java standard locale formatting rules on all platforms.
- Changed how the `date/time-string` parameter value is processed: ColdFusion processes the `date/time-string` parameter value time zone information differently than in earlier releases, as described in the Usage section.

Parameters

Parameter	Description
`date/time-string`	A string a variable that contains one, in a format that is readable in the current locale.
`locale`	Locale to use instead of the locale of the page when processing the function

Usage
This function can parse any date, time, or date/time combination that conforms to Java standard locale formatting rules for the current locale.

The following table lists some of the date/time values you can pass to this function in the English (US) locale. You can also pass only the date or the time parts of these formats:

Format	Example
m/dd/yy h:mm:ss	1/30/02 7:02:33
m/dd/yy h:mm tt	1/30/02 7:02 AM
m/dd/yyyy h:mm	1/30/2002 7:02 AM
mmm dd, yyyy h:mm:ss tt	Jan 30, 2002 7:02:12 AM

Format	Example
mmmm dd, yyyy h:mm:ss tt zzz	January 30, 2002 7:02:23 AM PST
ddd, mmm dd, yyyy hh:mm:ss	Wed, Jan 30, 2002 07:02:12
dddd, mmmm dd, yyyy h:mm:ss tt zzz	Wednesday, January 30, 2002 7:02:12 AM PST

Valid dates are in the range 100 AD–9999 AD. Two digit years in the range 00-29 are interpreted as being 2000-2029. Two digit years in the range 30-99 are interpreted as being 1930-1999

This function corrects for differences between the current time zone and any time zone specified in the input parameter.

- If a time zone specified in the `date/time-string` parameter is different from the time zone setting of the computer, ColdFusion adjusts the time value to its equivalent in the computer time zone.
- If a time zone is not specified in the `date/time-string` parameter, ColdFusion does not adjust the time value.

Note: *This function does not accept POP dates, which include a time zone offset value.*

Example

```
<h3>LSParseDateTime Example - returns a locale-specific date/time object</h3>
<!--- loop through a list of locales and show date values for Now()--->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><B><I>#locale#</I></B><br>
        <p>Locale-specific formats:
        <br>#LSDateFormat(Now(), "mmm-dd-yyyy")# #LSTimeFormat(Now())#<br>
        #LSDateFormat(Now(), "mmmm d, yyyy")# #LSTimeFormat(Now())#<br>
        #LSDateFormat(Now(), "mm/dd/yyyy")# #LSTimeFormat(Now())#<br>
        #LSDateFormat(Now(), "d-mmm-yyyy")# #LSTimeFormat(Now())#<br>
        #LSDateFormat(Now(), "ddd, mmmm dd, yyyy")# #LSTimeFormat(Now())#<br>
        #LSDateFormat(Now(), "d/m/yy")# #LSTimeFormat(Now())#<br>
        #LSDateFormat(Now())# #LSTimeFormat(Now())#<br>
    <p>Standard Date/Time:
    #LSParseDateTime("#LSDateFormat(Now())# #LSTimeFormat(Now())#")#<br>
    </cfoutput>
</cfloop>
```

LSParseEuroCurrency

Description

Formats a locale-specific currency string as a number. Attempts conversion through each of the default currency formats (none, local, international). Ensures correct handling of euro currency for Euro zone countries.

Returns

A formatted number that matches the value of the string.

Category

International functions, String functions

Function syntax

`LSParseEuroCurrency(currency-string [, locale])`

See also

`LSParseCurrency`, `LSEuroCurrencyFormat`, `SetLocale`; "Locale-specific content" on page 348 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java locale formatting rules on all platforms, except that it uses the rule detailed in the Usage section for countries in the Euro currency zone.

Parameters

Parameter	Description
`currency-string`	Locale-specific string or a variable that contains one.
`locale`	Locale to use instead of the locale of the page when processing the function

Usage

This function determines whether the current locale's country belongs to the Euro Zone, whose members have converted to the euro; if so, the `currency-string` parameter must be formatted in euros on all JVMs, including Sun JVM 1.3.1. If the country is not in the Euro zone, the string must follow the locale formatting rules of the JVM. For examples of valid currency formats in all supported locales, see "LSEuroCurrencyFormat" on page 1063.

For a list of the locale options that ColdFusion supports, and information on setting the default display format of date, time, number, and currency values, see `SetLocale`.

Example

```
<h3>LSParseEuroCurrency Example</h3>
<p>Loop through all available locales. Create string representations of the value
    123,456 in the three supported currency formats,
    and parse the results back to numbers.<p>
<cfloop list="#Server.Coldfusion.SupportedLocales#" index="locale" delimiters=",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p>Current Locale: <b><i>#locale#</i></b><br>
    <cfset localCurrency = LSEuroCurrencyFormat(123456, "local")>
        Value in local currency: #localCurrency#<br>
        Parsed using LSParseEuroCurrency:
        #LSParseEuroCurrency(localCurrency)#<br>
```

```
    <cfset IntlCurrency = LSEuroCurrencyFormat(123456, "international")>
        Value with International currency formatting: #IntlCurrency#<br>
        Parsed using LSParseEuroCurrency:
        #LSParseEuroCurrency(IntlCurrency)#<br>
    <cfset Currency = LSEuroCurrencyFormat(123456, "none")>
        Value with no currency formatting: #currency#<br>
        Parsed using LSParseEuroCurrency:
        #LSParseEuroCurrency(Currency)#<br>
        <hr noshade>
    </cfoutput>
</cfloop>
```

LSParseNumber

Description

Converts a string that is a valid numeric representation in the current locale into a formatted number.

Returns

A formatted number that matches the value of the string.

Category

International functions, String functions

Function syntax

`LSParseNumber(string [, locale])`

See also

`LSParseDateTime`, `SetLocale`; "Locales" on page 341 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.

Parameters

Parameter	Description
`string`	A string or a variable that contains one
`locale`	Locale to use instead of the locale of the page when processing the function

Usage

This function uses Java standard locale formatting rules on all platforms.

To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

Example

```
<h3>LSParseNumber Example</h3>
<p>LSParseNumber converts a locale-specific string to a number.
    Returns the number matching the value of string.
<!--- loop through a list of locales and show number values --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>

    <cfoutput><p><B><I>#locale#</I></B><br>
        #LSNumberFormat(-1234.5678, "__________")#<br>
        #LSNumberFormat(-1234.5678, "__________.___")#<br>
        #LSNumberFormat(1234.5678, "__________")#<br>
        #LSNumberFormat(1234.5678, "__________.___")#<br>
        #LSNumberFormat(1234.5678, "$_(__________.___)")#<br>
        #LSNumberFormat(-1234.5678, "$_(__________.___)")#<br>
        #LSNumberFormat(1234.5678, "+__________.___")#<br>
        #LSNumberFormat(1234.5678, "-__________.___")#<br>
        The actual number:
            #LSParseNumber(LSNumberFormat(1234.5678, "__________"))#<br>
```

```
        <hr noshade>
    </cfoutput>
</cfloop>
```

LSTimeFormat

Description

Formats the time part of a date/time string into a string in a locale-specific format.

Returns

A string representing the time value.

Category

Date and time functions, Display and formatting functions, International functions

Function syntax

`LSTimeFormat(time [, mask ])`

See also

`LSParseDateTime`, `LSDateFormat`, `TimeFormat`; "Locales" on page 341 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 6.1: Added the mask character L or l to represent milliseconds.

ColdFusion MX:

- Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.
- Added support for the following *mask* parameter options: `short`, `medium`, `long`, and `full`.

Parameters

Parameter	Description
string	• A date/time value • A string that is convertible to a time value A date/time object is in the range 100 AD–9999 AD.
mask	Masking characters that determine the format: • h: Hours; no leading zero for single-digit hours (12-hour clock) • hh: Hours; leading zero for single-digit hours. (12-hour clock) • H: Hours; no leading zero for single-digit hours (24-hour clock) • HH: Hours; leading zero for single-digit hours (24-hour clock) • m: Minutes; no leading zero for single-digit minutes • mm: Minutes; leading zero for single-digit minutes • s: Seconds; no leading zero for single-digit seconds • ss: Seconds; leading zero for single-digit seconds • l: Milliseconds • t: One-character time marker string, such as A or P. • tt: Multiple-character time marker string, such as AM or PM The following conform to Java locale-specific time encoding standards. Their exact formats depend on the locale: • short: includes hours, minutes; may include AM or PM • medium: includes hours, minutes; may include AM or PM • long: medium plus time zone • full: long, may also include an hour designator The default value is short.

Usage

This function uses Java standard locale formatting rules on all platforms.

When passing date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

To calculate a difference between time zones, use the `GetTimeZoneInfo` function.

To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

If no seconds value is passed to this function, and the mask value is `s`, the default output seconds format is one zero; for example, `lstimeformat(6:39, "h:m:s")` returns `6:39:0`. If the mask value is `ss`, it returns `6:39:00`.

Example

```
<h3>LSTimeFormat Example</h3>

<p>LSTimeFormat returns a time value using the locale convention.

<!--- loop through a list of locales and show time values --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
```

```
    <cfoutput><p><B><I>#locale#</I></B><br>
    #LSTimeFormat(Now())#<br>
    #LSTimeFormat(Now(), 'hh:mm:ss')#<br>
    #LSTimeFormat(Now(), 'hh:mm:sst')#<br>
    #LSTimeFormat(Now(), 'hh:mm:sstt')#<br>
    #LSTimeFormat(Now(), 'HH:mm:ss')#<br>
        <hr noshade>
    </cfoutput>

</cfloop>
```

LTrim

Description

Removes leading spaces from a string.

Returns

A copy of the string, without leading spaces.

Category

Display and formatting functions, String functions

Function syntax

`LTrim(string)`

See also

RTrim, ToBase64

Parameters

Parameter	Description
string	A string or a variable that contains one

Example

```
<h3>LTrim Example</h3>

<cfif IsDefined("FORM.myText")>
<cfoutput>
<pre>
Your string:"#FORM.myText#"
Your string:"#LTrim(FORM.myText)#"
(left trimmed)
</pre>
</cfoutput>
</cfif>

<form action = "ltrim.cfm">
<p>Type in some text, and it will be modified by LTrim to remove
    leading spaces from the left
<p><input type = "Text" name = "myText" value = " TEST">

<p><input type = "Submit" name = "">
</form>
```

Maxfilename

Description

Determines the greater of two numbers.

Returns

The greater of two numbers.

Category

Mathematical functions

Function syntax

`Max(number1, number2)`

See also

`Min`

Parameters

Parameter	Description
`number1, number2`	Numbers

Example

```
<h3>Max Example</h3>
<cfif IsDefined("FORM.myNum1")>
    <cfif IsNumeric(FORM.myNum1) and IsNumeric(FORM.myNum2)>
    <p>The maximum of the two numbers is <cfoutput>#Max(FORM.myNum1,
     FORM.myNum2)#</cfoutput>
    <p>The minimum of the two numbers is <cfoutput>#Min(FORM.myNum1,
     FORM.myNum2)#</cfoutput>
    <cfelse>
    <p>Please enter two numbers
    </cfif>
</cfif>

<form action = "max.cfm">
<h3>Enter two numbers, see the maximum and minimum of them</h3>

Number 1 <input type = "Text" name = "MyNum1">
<br>Number 2 <input type = "Text" name = "MyNum2">

<br><input type = "Submit" name = "" value = "See results">
</form>
```

Mid

Description

Extracts a substring from a string.

Returns

A string; the set of characters from *string*, beginning at *start*, of length *count*.

Category

String functions

Function syntax

```
Mid(string, start, count)
```

See also

Left, Len, Right

Parameters

Parameter	Description
string	A string or a variable that contains one. Must be single-quotation mark or double-quotation mark delimited.
start	A positive integer or a variable that contains one. Position at which to start count. Positions start with 1, not 0.
count	A positive integer or a variable that contains one. Number of characters to return. (Zero is not valid, but it does not throw an error.)

Example

```
<h3>Mid Example</h3>

<cfif IsDefined("Form.myText")>
    <!--- If len returns 0 (zero), then show error message. --->
    <cfif Len(Form.myText)>
        <cfif Len(Form.myText) LTE Form.RemoveChars>
            <cfoutput><p style="color: red; font-weight: bold">Your string
            #Form.myText# only has #Len(Form.myText)# characters. You cannot output
            the #Form.removeChars# middle characters of this string because it is
            not long enough.</p></cfoutput>
        <cfelseif Form.startPos GTE Len(Form.myText)>
            <cfoutput><p style="color: red; font-weight: bold">Your string
            #Form.myText# only has #Len(Form.myText)# characters. You cannot start
            at position #Form.startPos#.</p></cfoutput>
        <cfelse>
            <cfoutput><p>Your original string: <strong>#Form.myText#</strong></p>
            <p>Your changed string, showing only the <strong>#Form.removeChars#
            </strong> middle characters: <strong>#Mid(Form.myText,
            Form.startPos, Form.removeChars)#</strong></p></cfoutput>
        </cfif>
    <cfelse>
        <p style="color: red; font-weight: bold">Please enter a string of more
        than 0 (zero) characters.</p>
    </cfif>
</cfif>

<form action="<cfoutput>#CGI.ScriptName#</cfoutput>" method="POST">
<p>Type in some text<br />
<input type="Text" name="myText"></p>
```

```
<p>Enter a starting position (from the beginning of the entered text)<br />
<input name="startPos" type="text" size="1"></p>
<p>How many characters do you want to show?
<select name="RemoveChars">
<option value="1">1
<option value="3" selected>3
<option value="5">5
<option value="7">7
<option value="9">9</select>
<input type="Submit" name="Submit" value="Remove characters"></p>
</form>
```

Min

Description

Determines the lesser of two numbers.

Returns

The lesser of two numbers.

Category

Mathematical functions

Function syntax

`Min(number1, number2)`

See also

`Maxfilename`

Parameters

Parameter	Description
`number1, number2`	Numbers

Example

```
<h3>Min Example</h3>
<cfif IsDefined("FORM.myNum1")>
    <cfif IsNumeric(FORM.myNum1) and IsNumeric(FORM.myNum2)>
    <p>The maximum of the two numbers is <cfoutput>#Max(FORM.myNum1,
     FORM.myNum2)#</cfoutput>
    <p>The minimum of the two numbers is <cfoutput>#Min(FORM.myNum1,
     FORM.myNum2)#</cfoutput>
    <cfelse>
    <p>Please enter two numbers
    </cfif>
</cfif>

<form action = "min.cfm">
<h3>Enter two numbers, and see the maximum and minimum of the two numbers</h3>

Number 1 <input type = "Text" name = "MyNum1">
<br>Number 2 <input type = "Text" name = "MyNum2">

<br><input type = "Submit" name = "" value = "See results">
</form>
```

Minute

Description

Extracts the minute value from a date/time object.

Returns

The ordinal value of the minute, in the range 0–59.

Category

Date and time functions

Function syntax

`Minute(date)`

See also

`DatePart`, `Hash`, `Second`

Parameters

Parameter	Description
`date`	A date/time object, in the range 100 AD–9999 AD.

Usage

When passing a date/time value as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Example

```
<h3>Minute Example</h3>

<cfoutput>
The time is currently #TimeFormat(Now())#.
We are in hour #Hour(Now())#, Minute #Minute(Now())#
and Second #Second(Now())# of the day.
</cfoutput>
```

Month

Description
Extracts the month value from a date/time object.

Returns
The ordinal value of the month, in the range 1 (January) – 12 (December).

Category
Date and time functions

Function syntax
`Month(date)`

See also
`DatePart`, `MonthAsString`, `Quarter`

Parameters

Parameter	Description
`date`	Date/time object, in the range 100 AD–9999 AD.

Usage
When passing a date/time value as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Note: *You can pass the `CreateDate` function or the `Now` function as the date parameter of this function; for example:* `#Month(CreateDate(2001, 3, 3))#`.

Example
```
<h3>Month Example</h3>
<cfif IsDefined("FORM.year")>
More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
    <p>Your date, #DateFormat(yourDate)#.
    <br>It is #DayofWeekAsString(DayOfWeek(yourDate))#, day
     #DayOfWeek(yourDate)# in the week.
    <br>This is day #Day(yourDate)# in the month of
     #MonthAsString(Month(yourDate))#, which has
     #DaysInMonth(yourDate)# days.
    <br>We are in week #Week(yourDate)# of #Year(yourDate)#
     (day #DayofYear(yourDate)# of #DaysinYear(yourDate)#). <br>
    <cfif IsLeapYear(Year(yourDat"))>This is a leap year
        <cfelse>This is not a leap year
    </cfif>
</cfoutput>
</cfif>
```

MonthAsString

Description

Determines the name of the month that corresponds to *month_number*.

Returns

A string; the name of the specified month, in the current locale.

Category

Date and time functions, String functions

Function syntax

```
MonthAsString(month_number [, locale])
```

See also

DatePart, Month, Quarter

History

ColdFusion 8: Added the `locale` parameter.

Parameters

Parameter	Description
`month_number`	An integer in the range 1 – 12.
`locale`	Locale to use instead of the locale of the page when processing the function

Example

```
<h3>MonthAsString Example</h3>

<cfif IsDefined("FORM.year")>
<p>More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>

<cfoutput>
<p>Your date, #DateFormat(yourDate)#.
<br>It is #DayofWeekAsString(DayOfWeek(yourDate))#, day
 #DayOfWeek(yourDate)# in the week.
<br>This is day #Day(YourDate)# in the month of
 #MonthAsString(Month(yourDate))#, which has
#DaysInMonth(yourDate)# days.
<br>We are in week #Week(yourDate)# of #Year(yourDate)#
    (day #DayofYear(yourDate)# of #DaysinYear(yourDate)#). <br>
    <cfif IsLeapYear(Year(yourDate))>This is a leap year
        <cfelse>This is not a leap year
    </cfif>
</cfoutput>
</cfif>
```

Now

Description

Gets the current date and time of the computer running the ColdFusion server. The return value can be passed as a parameter to date functions such as `DaysInYear` or `FirstDayOfMonth`.

Returns

A date/time object; the current date and time of the computer running the ColdFusion server.

Category

Date and time functions

Function syntax

```
Now()
```

See also

`CreateDateTime`, `DatePart`

Example

```
<h3>Now Example</h3>
<p>Now returns the current date and time as a valid date/time object.

<p>The current date/time value is <cfoutput>#Now()#</cfoutput>
<p>You can also represent this as <cfoutput>#DateFormat(Now())#,
 #TimeFormat(Now())#</cfoutput>
```

NumberFormat

Description

Creates a custom-formatted number value. Supports the numeric formatting used in the U.S. For international number formatting, see LSNumberFormat.

Returns

A formatted number value:

- If no mask is specified, returns the value as an integer with a thousands separator.
- If the parameter value is "" (an empty string), returns 0.

Category

Display and formatting functions

Function syntax

NumberFormat(*number* [, *mask*])

See also

DecimalFormat, DollarFormat, IsNumeric, LSNumberFormat

History

ColdFusion MX: Changed behavior: if the mask format cannot correctly mask a number, this function returns the number unchanged. (It does not truncate the number nor throw an error.) (If no mask is selected, ColdFusion MX rounds the decimal part as ColdFusion 5 does. For example, it rounds 34.567 to 35.)

Parameters

Parameter	Description
number	A number.
mask	A string or a variable that contains one. Set of characters that determine how ColdFusion displays the number

The following table explains mask characters:

Mask character	Meaning
_ (underscore)	Optional. Digit placeholder.
9	Optional. Digit placeholder. (Shows decimal places more clearly than _.)
.	Location of a mandatory decimal point.
0	Located to the left or right of a mandatory decimal point. Pads with zeros.
( )	If number is less than zero, puts parentheses around the mask.
+	Puts plus sign before positive number; minus sign before negative number.
-	Puts a space before positive number; minus sign before negative number.
,	Separates every third decimal place with a comma.

Mask character	Meaning
L,C	Left-justifies or center-justifies number within width of mask column. First character of mask must be L or C. The default value is right-justified.
$	Puts a dollar sign before formatted number. First character of mask must be the dollar sign ($).
^	Separates left and right formatting.

Note: *If you do not specify a sign for the mask, positive and negative numbers do not align in columns. To put a plus sign or space before positive numbers and a minus sign before negative numbers, use the plus or minus sign, respectively.*

Usage

This function uses Java standard locale formatting rules on all platforms.

The position of symbols in format masks determines where the codes take effect. For example, if you put a dollar sign at the far left of a format mask, ColdFusion displays a dollar sign at the left edge of the formatted number. If you separate the dollar sign on the left edge of the format mask by at least one underscore, ColdFusion displays the dollar sign just to the left of the digits in the formatted number.

These examples show how symbols determine formats:

Number	Mask	Result
4.37	$____.__	"$ 4.37"
4.37	_$___.__	" $4.37"

The positioning can also show where to place the minus sign for negative numbers:

Number	Mask	Result
-4.37	-____.__	"- 4.37"
-4.37	_-___.__	" -4.37"

The positions for a symbol are: far left, near left, near right, and far right. The left and right positions are determined by the side of the decimal point on which the code character is shown. For formats that do not have a fixed number of decimal places, you can use a caret (^) to separate the left fields from the right.

An underscore determines whether the code is placed in the far or near position. Most code characters' effect is determined by the field in which they are located. This example shows how to specify where to put parentheses to display negative numbers:

Number	Mask	Result
3.21	C(__^__)	"(3.21)"
3.21	C__(^__)	" (3.21)"
3.21	C(__^)__	"(3.21) "
3.21	C__(^)__	" (3.21) "

When converting from string to double, to prevent rounding errors, this function adds a rounding factor of 1.5543122344752E-014 to the converted number. For example, without adding the rounding factor, converting the string value 1.275 to double with two digits of precision results in a value of 1.27499999999999999, which would be rounded up to 1.27. By adding the rounding factor, the conversion correctly results in a value of 1.28.

If you round off a double such as 1.9949999999999999999999999999, where the last decimal is 10E-14, the rounding factor can cause an incorrect result.

To set the default display format of date, time, number, and currency values, use the SetLocale function.

Example

```
<h3>NumberFormat Example</h3>

<cfloop FROM = 1000 TO = 1020 INDEX = "counter">
<cfset CounterRoot2 = counter * sqr(2)>

<!--- Show result in default format, adding comma for thousands place;
    and in custom format, displaying to two decimal places --->
<cfoutput>
<pre>#counter# * Square Root of 2: #NumberFormat(CounterRoot2,
'_____.__')#</pre>
<pre>#counter# * Square Root of 2: #NumberFormat(CounterRoot2)#</pre>
</cfoutput>
</cfloop>
```

ParagraphFormat

Description

Replaces characters in a string:

- Single newline characters (CR/LF sequences) with spaces
- Double newline characters with HTML paragraph tags (<p>)

Returns

A copy of the string, with characters converted.

Category

Display and formatting functions, String functions

Function syntax

`ParagraphFormat(`*`string`*`)`

See also

`StripCR`

Parameters

Parameter	Description
`string`	A string or a variable that contains one

Usage

This function is useful for displaying data entered in `textarea` fields.

Example

```
<h3>ParagraphFormat Example</h3>
<p>Enter text into this textarea, and see it returned as HTML.
<cfif IsDefined("FORM.myTextArea")>
    <p>Your text area, formatted
    <p><cfoutput>#ParagraphFormat(FORM.myTextArea)#</cfoutput>
</cfif>
<!--- use #Chr(10)##Chr(13)# to simulate a line feed/carriage
return combination; i.e, a return --->
<form action = "paragraphformat.cfm">
<textArea name = "MyTextArea" cols = "35" ROWS = 8>
This is sample text and you see how it scrolls
    <cfoutput>#Chr(10)##Chr(13)#</cfoutput>
    From one line
    <cfoutput>#Chr(10)##Chr(13)##Chr(10)##Chr(13)#</cfoutput>
    to the next
</textArea>
<input type = "Submit" name = "Show me the HTML version">
</form>
```

ParameterExists

Description

This function is deprecated. Use the `IsDefined` function.

Determines whether a parameter exists. ColdFusion does not evaluate the argument.

History

ColdFusion MX: Deprecated this function. It might not work, and might cause an error, in later releases.

ParseDateTime

Description

Parses a date/time string according to the English (U.S.) locale conventions. (To format a date/time string for other locales, use the LSParseDateTime function.)

Returns

A date/time object

Category

Date and time functions, Display and formatting functions

Function syntax

```
ParseDateTime(date/time-string [, pop-conversion ])
```

See also

IsDate, IsNumericDate, SetLocale

Parameters

Parameter	Description
date/time string	A string containing a date/time value formatted according to U.S. locale conventions. Can represent a date/time in the range 100 AD–9999 AD. Years 0-29 are interpreted as 2000-2029; years 30-99 are interpreted as 1930-1999.
pop-conversion	• pop: specifies that the date/time string is in POP format, which includes the local time of the sender and a time-zone offset from UTC. ColdFusion applies the offset and returns a value with the UTC time. • standard: (the default) function does no conversion.

Usage

This function is similar to CreateDateTime, but it takes a string instead of enumerated date/time values. These functions are provided primarily to increase the readability of code in compound expressions.

To calculate a difference between time zones, use the GetTimeZoneInfo function.

To set the default display format of date, time, number, and currency values, use the SetLocale function.

Example

```
<h3>ParseDateTime Example</h3>
<cfif IsDefined("form.theTestValue")>
    <cfif IsDate(form.theTestValue)>
    <h3>The expression <cfoutput>#DE(form.theTestValue)#</cfoutput> is a valid date</h3>
    <p>The parsed date/time is:
 <cfoutput>#ParseDateTime(form.theTestValue)#</cfoutput>
    <cfelse>
    <h3>The expression <cfoutput>#DE(form.theTestValue)#</cfoutput> is not a valid date</h3>
    </cfif>
</cfif>

<form action="#CGI.ScriptName#" method="POST">
<p>Enter an expression, and discover if it can be evaluated to a date value.
<input type="Text" name="TheTestValue" value="<CFOUTPUT>#DateFormat(Now())#
 #TimeFormat(Now())#</CFOUTPUT>">
<input type="Submit" value="Parse the Date" name="">
</form>
```

Pi

Description

Gets the mathematical constant π, accurate to 15 digits.

Returns

The number 3.14159265358979.

Category

Mathematical functions

Function syntax

```
Pi()
```

See also

ASin, Cos, Sin, Tan

Example

```
<h3>Pi Example</h3>
<!--- By default, ColdFusion displays only 11 significant digits.
 Use NumberFormat to display all 15. --->
The Pi function Returns the number
<cfoutput>
#NumberFormat(Pi(), "_._______________")#,
</cfoutput> the mathematical constant pi, accurate to 15 digits.
```

PrecisionEvaluate

Description

Evaluates one or more string expressions, dynamically, from left to right, using BigDecimal precision arithmetic to calculate the values of arbitrary precision arithmetic expressions.

Returns

An object; the result of the evaluation(s).

Category

Mathematical functions, Dynamic evaluation functions

Function syntax

```
PrecisionEvaluate(string_expression1 [, string_expression2 , ... ])
```

See also

`Evaluate`, "Using Expressions and Number Signs" on page 50 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`string_expression1, string_expression2...`	Expressions to evaluate

Usage

The `PrecisionEvaluate` function lets you calculate arbitrarily long decimal (BigDecimal precision) values. BigDecimal precision arithmetic accepts and generates decimal numbers of any length, and does not use exponential notation.

The `PrecisionEvaluate` function calculates arbitrary precision results only for addition, subtraction, multiplication and division. If you use any of the following operations, ColdFusion performs normal integer or floating point arithmetic and does not return BigDecimal values.

- exponentiation (^)
- modulus (MOD or %)
- integer division (\)

This function differs from the `Evaluate` function only in its use of BigDecimal precision arithmetic to calculate numeric values; otherwise the two functions are identical. The results of an evaluation on the left can have meaning in an expression to the right, and the function returns the result of evaluating the rightmost expression. If a string expression contains a single- or double-quotation mark, the mark must be escaped.

If an expression, such as 1/3, results in an infinitely repeating decimal value, ColdFusion limits the decimal part to 20 digits.

Note: *To increase processing efficiency, do not put the arithmetic expressions to evaluate in quotation marks ("). ColdFusion compiles* `PrecisionEvaluate(a*b)` *more efficiently than it compiles* `PrecisionEvaluate("a*b")`*, although both formats produce the same results.*

Example

```
<h3>PrecisionEvaluate Example</h3>
<cfif IsDefined("FORM.myExpression")>
    <cftry>
```

```
        <!--- Evaluate the expression and display the result. --->
        <cfset theExpression = PrecisionEvaluate(Form.myExpression)>
        <cfoutput>
            The value of the expression #FORM.MyExpression#
            is #theExpression#.<br>
        </cfoutput>

        <cfcatch type="any">
            <cfoutput>Could not evaluate the expression #Form.myExpression#.
            </cfoutput>
        </cfcatch>
    </cftry>
</cfif>

<cfform preservedata="yes">
    <h3>Enter a ColdFusion expression for evaluation.</h3>
    <p>Try using some really big decimal numbers.</p>
    <cfinput type="text" name="myExpression" size="60"><br>
    <br>
    <cfinput type="submit" name="submit">
</cfform>
```

PreserveSingleQuotes

Description

Prevents ColdFusion from automatically escaping single-quotation mark characters that are contained in a variable. ColdFusion does not evaluate the argument.

Returns

(None)

Category

Other functions

Function syntax

`PreserveSingleQuotes(variable)`

History

ColdFusion MX: Changed behavior: ColdFusion automatically escapes simple-variable, array-variable, and structure-variable references within a `cfquery` tag body or block. (Earlier releases did not automatically escape array-variable references.)

Parameters

Parameter	Description
`variable`	Variable that contains a string in which to preserve single-quotation marks.

Usage

This function is useful in SQL statements to defer evaluation of a variable reference until runtime. This prevents errors that result from the evaluation of a single-quote or apostrophe data character (for example, "Joe's Diner") as a delimiter.

Example A: Consider this code:

```
<cfset mystring = "'Newton's Law', 'Fermat's Theorem'">

PreserveSingleQuotes(#mystring#) is
<cfoutput>
    #PreserveSingleQuotes(mystring)#
</cfoutput>
```

The output is as follows:

```
PreserveSingleQuotes(#mystring#) is 'Newton's Law', 'Fermat's Theorem'
```

Example B: Consider this code:

```
<cfset list0 = " '1','2', '3' ">
<cfquery sql = "select * from foo where bar in (#list0#)">
```

ColdFusion escapes the single-quote characters in the list as follows:

```
""1"", ""2"", ""3""
```

The `cfquery` tag throws an error.

You code this function correctly as follows:

```
<cfquery sql = "select * from foo where bar in (#preserveSingleQuotes(list0)#)"> **tharwood
11/16
```

This function ensures that ColdFusion evaluates the code as follows:

```
'1', '2', '3'
```

Example

```
<h3>PreserveSingleQuotes Example</h3><p>This is a useful function for
    creating lists of information to return from a query. In this example,
    we pick the list of Centers in Suisun, San Francisco, and San Diego,
    using the SQL grammar IN to modify a WHERE clause, rather than looping
    through the result set after the query is run.
<cfset List = "'Suisun', 'San Francisco', 'San Diego'">
<cfquery name = "GetCenters" datasource = "cfdocexamples">
    SELECT Name, Address1, Address2, City, Phone
    FROM Centers
    WHERE City IN (#PreserveSingleQuotes(List)#)
</cfquery>
<p>We found <cfoutput>#GetCenters.RecordCount#</cfoutput> records.
<cfoutput query = "GetCenters">
<p>#Name#<br>
#Address1#<br>
<cfif Address2 is not "">#Address2#
    </cfif>
#City#<br>
#Phone#<br>
</cfoutput>
```

Quarter

Description
Calculates the quarter of the year in which a date falls.

Returns
An integer, 1–4.

Category
Date and time functions

Function syntax
`Quarter(date)`

See also
`DatePart`, `Month`

Parameters

Parameter	Description
`date`	A date/time object in the range 100 AD–9999 AD.

Usage
When passing a date/time value as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Example
```
<h3>Quarter Example</h3>

Today, <cfoutput>#DateFormat(Now())#</cfoutput>,
is in Quarter <cfoutput>#Quarter(Now())#</cfoutput>.
```

QueryAddColumn

Description

Adds a column to a query and populates its rows with the contents of a one-dimensional array. Pads query columns, if necessary, to ensure that all columns have the same number of rows.

Returns

The number of the column that was added.

Category

Query functions

Function syntax

`QueryAddColumn(query, column-name [, datatype], array-name)`

See also

`QueryNew`, `QueryAddRow`, `QuerySetCell`; "Managing data types for columns" on page 426 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added the `datatype` parameter.

ColdFusion MX: Changed behavior: if a user attempts to add a column whose name is invalid, ColdFusion throws an error. (In earlier releases, ColdFusion permitted the add operation, but the user could not reference the column after adding it.)

Parameters

Parameter	Description
`query`	Name of a query object.
`column-name`	Name of the new column.
`datatype`	(Optional) Column data type. ColdFusion generates an error if data you add to the column is not of this type, or if it cannot convert the data to this type. The following data types are valid: • Integer: 32-bit integer • BigInt: 64-bit integer • Double: 64-bit decimal number • Decimal: Variable length decimal, as specified by java.math.BigDecimal • VarChar: String • Binary: Byte array • Bit: Boolean (1=True, 0=False) • Time: Time • Data: Date (can include time information)
`array-name`	Name of an array whose elements populate the new column.

Usage

You can add columns to query objects, such as queries retrieved with the `cfquery` tag or queries created with the `QueryNew` function. You cannot use the `QueryAddColumn` function on a cached query. This function is useful for generating a query object from the arrays of output parameters that Oracle stored procedures can generate.

Adobe recommends that you use the optional *datatype* parameter. Without this parameter, ColdFusion must try to determine the column's data type when it uses the query object in a query of queries. Determining the data type requires additional processing, and can result in errors if ColdFusion does not guess the type correctly.

Example

The following example creates a new query object, uses the QueryAddColumn function to add three columns to the object, and displays the results. Because two of the arrays that provide the data are shorter than the third, QueryAddColumn adds padding to the corresponding columns in the query.

```
<!--- Make a query. --->
<cfset myQuery = QueryNew("")>

<!--- Create an array. --->
<cfset FastFoodArray = ArrayNew(1)>
<cfset FastFoodArray[1] = "French Fries">
<cfset FastFoodArray[2] = "Hot Dogs">
<cfset FastFoodArray[3] = "Fried Clams">
<cfset FastFoodArray[4] = "Thick Shakes">
<!--- Use the array to add a column to the query. --->
<cfset nColumnNumber = QueryAddColumn(myQuery, "FastFood", "VarChar",
           FastFoodArray)>

<!--- Create a second array. --->
<cfset FineCuisineArray = ArrayNew(1)>
<cfset FineCuisineArray[1] = "Lobster">
<cfset FineCuisineArray[2] = "Flambe">
<!--- Use the array to add a second column to the query. --->
<cfset nColumnNumber2 = QueryAddColumn(myQuery, "FineCuisine", "VarChar",
            FineCuisineArray)>

<!--- Create a third array. --->
<cfset HealthFoodArray = ArrayNew(1)>
<cfset HealthFoodArray[1] = "Bean Curd">
<cfset HealthFoodArray[2] = "Yogurt">
<cfset HealthFoodArray[3] = "Tofu">
<!--- Use the array to add a third column to the query. --->
<cfset nColumnNumber3 = QueryAddColumn(myQuery, "HealthFood", "VarChar",
            HealthFoodArray)>

<!--- Display the results. --->
<table cellspacing = "2" cellpadding = "2" border = "0">
<tr>
<th align = "left">Fast Food</th>
<th align = "left">Fine Cuisine</th>
<th align = "left">Health Food</th>
</tr>
<cfoutput query = "myQuery">
<tr>
<td>#FastFood#</td>
<td>#FineCuisine#</td>
<td>#HealthFood#</td>
</tr>
</cfoutput>
</table>
```

QueryAddRow

Description
Adds a specified number of empty rows to a query.

Returns
The number of rows in the query

Category
Query functions

Function syntax
`QueryAddRow(query [, number])`

See also
`QueryAddColumn`, `QueryAddRow`, `QuerySetCell`, `QueryNew`; "Creating a record set with the QueryNew() function" on page 415 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`query`	Name of an executed query.
`number`	Number of rows to add to the query. The default value is 1.

Example
```
<h3>QueryAddRow Example</h3>

<!--- start by making a query --->
<cfquery name = "GetCourses" datasource = "cfdocexamples">
    SELECT Course_ID, Descript
    FROM Courses
</cfquery>
<p>The Query "GetCourses" has <cfoutput>#GetCourses.RecordCount#</cfoutput> rows.

<cfset CountVar = 0>
<cfloop CONDITION = "CountVar LT 15">
    <cfset temp = QueryAddRow(GetCourses)>
    <cfset CountVar = CountVar + 1>
    <cfset Temp = QuerySetCell(GetCourses, "Number", 100*CountVar)>
    <cfset CountVar = CountVar + 1>
    <cfset Temp = QuerySetCell(GetCourses, "Descript",
    "Description of variable #Countvar#")>
</cfloop>

<P>After the QueryAddRow action, the query has
    <CFOUTPUT>#GetCourses.RecordCount#</CFOUTPUT>
    records.
    <CFOUTPUT query="GetCourses">
    <PRE>#Course_ID# #Course_Number# #Descript#</pre> </cfoutput>
```

QueryConvertForGrid

Description

Converts query data to a structure that contains a paged subset of the query. Used in CFC functions that return data to Ajax format `cfgrid` controls in response to a bind expression.

Returns

A structure that contains one page of data from the query.

Category

Query functions

Function syntax

`QueryConvertForGrid(query, page, pageSize)`

See also

`cfgrid`, "Dynamically filling form data" on page 631 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function.

Parameters

Parameter	Description
`query`	Name of the query whose data is returned.
`page`	The specific page of query data to be returned. Pages are numbered starting with 1.
`pageSize`	Number of rows of query data on a page.

Usage

You can also create the return value for a `cfgrid` bind CFC without using this function if your query returns only a single grid page of data at a time. For more information see "Using Ajax UI Components and Features" on page 614 in the *ColdFusion Developer's Guide*.

Example

The following example shows how a CFC function that is called by an Ajax format `cfgrid` tag `bind` attribute. uses the `QueryConvertForGrid` function to prepare query data to return to the grid. The CFML page with the `cfgrid` tag has the following code:

```
<cfform>
    <cfgrid format="html" name="grid01" pagesize=5 sort=true
            bind="cfc:places.getData({cfgridpage},{cfgridpagesize},
                {cfgridsortcolumn},{cfgridsortdirection})" selectMode="row">
        <cfgridcolumn name="Emp_ID" display=true header="Employee ID"/>
        <cfgridcolumn name="FirstName" display=true header="Name"/>
        <cfgridcolumn name="Email" display=true header="Email"/>
    </cfgrid>
</cfform>
```

The `getData` function in the places.cfc page has the following code:

```
<cffunction name="getData" access="remote" output="false">
    <cfargument name="page">
    <cfargument name="pageSize">
```

```
    <cfargument name="gridsortcolumn">
    <cfargument name="gridstartdirection">
    <cfset query = "SELECT Emp_ID, FirstName, EMail
         FROM Employees" >
    <cfif gridsortcolumn neq "" or gridstartdirection neq "">
     <cfset query=query & " order by #gridsortcolumn#
            #gridstartdirection#">
    </cfif>
    <cfquery name="team" datasource="cfdocexamples">
         <cfoutput>#query#</cfoutput>
    </cfquery>
    <cfreturn QueryConvertForGrid(team, page, pageSize)>
</cffunction>
```

QueryNew

Description
Creates an empty query (query object).

Returns
An empty query with a set of named columns, or an empty query.

Category
Query functions

Function syntax
`QueryNew(columnlist [, columntypelist])`

History
ColdFusion MX 7: Added `columntypelist` parameter.

See also
`QueryAddColumn`, `QueryAddRow`, `QuerySetCell`; "Managing data types for columns" on page 426 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`columnlist`	Comma-delimited list of column names, or an empty string.
`columntypelist`	(Optional) Comma-delimited list specifying column data types. ColdFusion generates an error if the data you add to the column is not of this type, or if it cannot convert the data to this type. The following data types are valid: • Integer: 32-bit integer • BigInt: 64-bit integer • Double: 64-bit decimal number • Decimal: Variable length decimal, as specified by java.math.BigDecimal • VarChar: String • Binary: Byte array • Bit: Boolean (1=True, 0=False) • Time: Time • Date: Date (can include time information)

Usage
If you specify an empty string in the *columnlist* parameter, you must use the `QueryAddColumn` function to add columns to the query.

Adobe recommends that you use the optional *columntypelist* parameter. Without this parameter, ColdFusion must try to determine data types when it uses the query object in a query of queries. Determining data types requires additional processing, and can result in errors if ColdFusion does not guess a type correctly.

Example
The following example uses the `QueryNew` function to create an empty query with three columns. It populates two rows of the query and displays the contents of the query object and its metadata.

```
<!--- Create a new three-column query, specifying the column data types --->
<cfset myQuery = QueryNew("Name, Time, Advanced", "VarChar, Time, Bit")>

<!--- Make two rows in the query --->
<cfset newRow = QueryAddRow(MyQuery, 2)>

<!--- Set the values of the cells in the query --->
<cfset temp = QuerySetCell(myQuery, "Name", "The Wonderful World of CMFL", 1)>
<cfset temp = QuerySetCell(myQuery, "Time", "9:15 AM", 1)>
<cfset temp = QuerySetCell(myQuery, "Advanced", False, 1)>
<cfset temp = QuerySetCell(myQuery, "Name", "CFCs for Enterprise
        Applications", 2)>
<cfset temp = QuerySetCell(myQuery, "Time", "12:15 PM", 2)>
<cfset temp = QuerySetCell(myQuery, "Advanced", True, 2)>

<h4>The query object contents</h4>
<cfoutput query = "myQuery">
    #Name# #Time# #Advanced#<br>
</cfoutput><br>
<br>
<h4>Using individual query data values</h4>
<cfoutput>
    #MyQuery.name[2]# is at #MyQuery.Time[2]#<br>
</cfoutput><br>
<br>
<h4>The query metadata</h4>
<cfset querymetadata=getMetaData(myQuery)>
<cfdump var="#querymetadata#">
```

QuerySetCell

Description

Sets a cell to a value. If no row number is specified, the cell on the last row is set.

Starting with ColdFusion MX 7, you cannot add a string literal (for example, "All") to a column that is of type numeric, although this was allowed in previous versions of ColdFusion.

Returns

True, if successful; False, otherwise.

Category

Query functions

Function syntax

```
QuerySetCell(query, column_name, value [, row_number ])
```

See also

QueryAddColumn, QueryAddRow, QueryNew; “Creating a record set with the QueryNew() function” on page 415 in the *ColdFusion Developer’s Guide*

History

ColdFusion MX 7: Changed the behavior of the function so that it does type validation.

Parameters

Parameter	Description
query	Name of an executed query.
column_name	Name of a column in the query.
value	Value to set in the cell.
row_number	Row number. The default value is last row.

Example

```
<!--- This example shows the use of QueryAddRow and QuerySetCell --->

<!--- start by making a query --->
<cfquery name = "GetCourses" datasource = "cfdocexamples">
    SELECT Course_ID, Descript
    FROM Courses
</cfquery>
<p>The Query "GetCourses" has <cfoutput>#GetCourses.RecordCount#</cfoutput> rows.

<cfset CountVar = 0>
<cfloop CONDITION = "CountVar LT 15">
    <cfset temp = QueryAddRow(GetCourses)>
    <cfset CountVar = CountVar + 1>
    <cfset Temp = QuerySetCell(GetCourses, "Number", 100*CountVar)>
    <cfset CountVar = CountVar + 1>
    <cfset Temp = QuerySetCell(GetCourses, "Descript",
    "Description of variable #Countvar#")>
</cfloop>

<P>After the QueryAddRow action, the query has
    <CFOUTPUT>#GetCourses.RecordCount#</CFOUTPUT>
```

```
records.
<CFOUTPUT query="GetCourses">
<PRE>#Course_ID# #Course_Number# #Descript#</pre> </cfoutput>
```

QuotedValueList

Description

Gets the values of each record returned from an executed query. ColdFusion does not evaluate the arguments.

Returns

A delimited list of the values of each record returned from an executed query. Each value is enclosed in single-quotation marks.

Category

Query functions, List functions

Function syntax

`QuotedValueList(query.column [, delimiter ])`

See also

`ValueList`

Parameters

Parameter	Description
`query.column`	Name of an executed query and column. Separate query name and column name with a period.
`delimiter`	A string or a variable that contains one. Character(s) that separate column data.

Example

```
<!--- use the contents of one query to create another dynamically --->
<cfset List = "'BIOL', 'CHEM'">
<!--- first, get the department IDs in our list --->
<cfquery name = "GetDepartments" datasource = "cfdocexamples">
    SELECT Dept_ID FROM Departments
    WHERE Dept_ID IN (#PreserveSingleQuotes(List)#)
</cfquery>

<!--- now, select the courses for that department based on the
quotedValueList produced from our previous query --->
<cfquery name = "GetCourseList" datasource = "cfdocexamples">
    SELECT *
    FROM CourseList
    WHERE Dept_ID IN ('#GetDepartments.Dept_ID#')
</cfquery>

<!--- now, output the results --->

List the course numbers that are in BIOL and CHEM (uses semicolon (;) as the delimiter):<br>
<cfoutput>
#QuotedValueList(GetCourseList.CorNumber,";")#<br>
</cfoutput>
```

Rand

Description
Generates a pseudo-random number.

Returns
A pseudo-random decimal number, in the range 0 – 1.

Category
Mathematical functions, Security functions

Function syntax
`Rand([algorithm])`

History
ColdFusion MX 7: Added the *algorithm* parameter.

See also
Randomize, RandRange

Parameters

Parameter	Description
algorithm	(Optional) The algorithm to use to generated the random number. ColdFusion installs a cryptography library with the following algorithms: • CFMX_COMPAT: the algorithm used in ColdFusion (default). • SHA1PRNG: generates a number using the Sun Java SHA1PRNG algorithm. This algorithm provides greater randomness than the default algorithm • IBMSecureRandom: for IBM WebSphere (IBM JVM does not support the SHA1PRNG algorithm).

Usage
Call the Randomize function before calling this function to seed the random number generator. Seeding the generator ensures that the Rand function always generates the same sequence of pseudo-random numbers. This behavior is useful if you must reproduce a pattern consistently.

ColdFusion MX 7 uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section (except the default algorithm). The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

Example
The following example uses the SHA1PRNG algorithm to generate a single random number:

```
<h3>Rand Example</h3>
<cfoutput>
    <p>Rand("SHA1PRNG") returned: #Rand("SHA1PRNG")#</p>
    <p><A HREF = "#CGI.SCRIPT_NAME#">Try again</A>
</cfoutput>
```

Randomize

Description

Seeds the pseudo-random number generator with an integer number, ensuring repeatable number patterns.

Returns

A pseudo-random decimal number, in the range 0–1.

Category

Mathematical functions, Security functions

Function syntax

```
Randomize(number[, algorithm])
```

History

ColdFusion MX 7: Added the *algorithm* parameter.

See also

Rand, RandRange

Parameters

Parameter	Description
number	Integer number. If the number is not in the range -2,147,483,648 — 2,147,483,647, ColdFusion generates an error.
algorithm	(Optional) The algorithm to use to generate the seed number. ColdFusion installs a cryptography library with the following algorithms: • CFMX_COMPAT: the algorithm used in ColdFusion (default). • SHA1PRNG: generates a number using the Sun Java SHA1PRNG algorithm. This algorithm provides greater randomness than the default algorithm. • IBMSecureRandom: for IBM WebSphere (IBM JVM does not support the SHA1PRNG algorithm).

Usage

Call this function before calling Rand to seed the random number generator. Seeding the generator ensures that the Rand function always generates the same sequence of pseudo-random numbers. This behavior is useful if you must reproduce a pattern consistently.

ColdFusion MX 7 uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section (except the default algorithm). The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

Example

The following example calls the Randomize function to seed the random number generator and generates 10 random numbers. To show the effect of the seed, submit the form with the same value multiple times.

```
<h3>Randomize Example</h3>

    <!--- Do the following only if the form has been submitted. --->
<cfif IsDefined("Form.myRandomInt")>

    <!--- Make sure submitted value is a number and display its value. --->
    <cfif IsNumeric(FORM.myRandomInt)>
```

```
        <cfoutput>
            <b>Seed value is #FORM.myRandomInt#</b><br>
        </cfoutput><br>

        <!--- Call Randomize to seed the random number generator. --->
        <cfset r = Randomize(FORM.myRandomInt, "SHA1PRNG")>

        <cfoutput>
            <b>Random number returned by Randomize(#Form.myRandomInt#,
                    "SHA1PRNG"):</b><br>
            #r#<br>
            <br>
            <b>10 random numbers generated using the SHA1PRNG algorithm:</b><br>
            <cfloop index = "i" from = "1" to = "10" step = "1">
                #Rand("SHA1PRNG")#<br>
            </cfloop><br>
        </cfoutput>

    <cfelse>
        <p>Please enter a number.
    </cfif>
</cfif>

<!--- Form to specify the seed value. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
<p>Enter a number to seed the randomizer:
<input type = "Text" name = "MyRandomInt" value="12345">
<p><input type = "Submit" name = "">
</form>
```

RandRange

Description
Generates a pseudo-random integer in the range between two specified numbers.

Returns
A pseudo-random integer.

Category
Mathematical functions, Security functions

Function syntax
`RandRange(number1, number2[, algorithm])`

History
ColdFusion MX 7: Added the *algorithm* parameter.

See also
Rand, Randomize

Parameters

Parameter	Description
number1, number2	Integer numbers. If the numbers are not in the range -2,147,483,648 — 2,147,483,647, ColdFusion generates an error.
algorithm	(Optional) The algorithm to use to generated the random number. ColdFusion installs a cryptography library with the following algorithms: • CFMX_COMPAT: the algorithm used in ColdFusion (default). • SHA1PRNG: generates a number using the Sun Java SHA1PRNG algorithm. This algorithm provides greater randomness than the default algorithm • IBMSecureRandom: for IBM WebSphere (IBM JVM does not support the SHA1PRNG algorithm.)

Usage
Very large positive or negative values for the *number1* and *number2* parameters might result in poor randomness in the results. To prevent this problem, do not specify numbers outside the range -1,000,000,000 – 1,000,000,000.

ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section (except the default algorithm). The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

Example
The following example contains a form that requires random number range values, and lets you optionally specify a random number seed value. It uses `cfform` controls and attributes to specify a default range, ensure that the range fields have values, and validate that the field values are in a specified integer range. When you submit the form, it checks whether the seed field has an empty string; if the field has a value, the code uses the number to seed the random number generator. It then generates and displays the random number.

```
<h3>RandRange Example</h3>

<!--- Do the following only if the form has been submitted. --->
```

```
<cfif IsDefined("Form.mySeed")>

    <!--- Do the following only if the seed field has a non-empty string. --->
    <cfif Form.mySeed NEQ "">
        <cfoutput>
            <b>Seed value is #FORM.mySeed#</b><br>
        </cfoutput>
        <br>

        <!--- Call Randomize to seed the random number generator. --->
        <cfset r = Randomize(FORM.mySeed, "SHA1PRNG")>
    <cfelse>
        <b>No Seed value submitted</b><br>
    </cfif>

    <!--- Generate and display the random number. --->
    <cfoutput><p><b>
        RandRange returned: #RandRange(FORM.myInt, FORM.myInt2, "SHA1PRNG")#
    </cfoutput></b></p>
</cfif>

<!--- This form uses cfform input validation to check the input range. --->
<cfform action = "#CGI.SCRIPT_NAME#">
<p>Enter the random number Range: From
<cfinput type = "Text" name = "MyInt" value = "1"
    RANGE = "-1000000000,1000000000"
    message = "Please enter a value between -1,000,000,000 and 1,000,000,000"
    validate = "integer" required = "Yes">
To
<cfinput type = "Text" name = "MyInt2" value = "9999"
    RANGE = "-1000000000,1000000000"
    message = "Please enter a value between --1,000,000,000and 1,000,000,000"
    validate = "integer" required = "Yes"></p>
<p>Enter a number to seed the randomizer:
<cfinput type = "Text" name = "mySeed" RANGE = "-1000000000,1000000000"
    message = "Please enter a value between -1,000,000,000 and 1,000,000,000"
    validate = "integer" required = "No"></p>
<p><input type = "Submit" name = "">
</cfform>
```

REFind

Description

Uses a regular expression (RE) to search a string for a pattern. The search is case sensitive.

For more information on regular expressions, including escape sequences, anchors, and modifiers, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.

Returns

Depends on the value of the `returnsubexpressions` parameter:

- If `returnsubexpressions = "False"`:
  - The position in the string where the match begins
  - 0, if the regular expression is not matched in the string
- If `returnsubexpressions = "True"`: a structure that contains two arrays, `len` and `pos`. The array elements are as follows:
  - If the regular expression is found in the string, the first element of the `len` and `pos` arrays contains the length and position, respectively, of the first match of the entire regular expression.
  - If the regular expression contains parentheses that group subexpressions, each subsequent array element contains the length and position, respectively, of the first occurrence of each group.
  - If the regular expression is not found in the string, the first element of the `len` and `pos` arrays contains 0.

Category

String functions

Function syntax

`REFind(`*`reg_expression, string [, start, returnsubexpressions`*` ] )`

See also

`Find`, `FindNoCase`, `REFindNoCase`, `REReplace`, `REReplaceNoCase`

Parameters

Parameter	Description
`reg_expression`	Regular expression for which to search. Case-sensitive.
`string`	A string, or a variable that contains one, in which to search.
`start`	Optional. A positive integer, or a variable that contains one. Position in the string at which to start search. The default value is 1.
`returnsubexpressions`	Optional. Boolean. Whether to return substrings of reg_expression, in arrays named `len` and `pos`: • True: if the regular expression is found, the first array element contains the length and position, respectively, of the first match. If the regular expression contains parentheses that group subexpressions, each subsequent array element contains the length and position, respectively, of the first occurrence of each group. If the regular expression is not found, the arrays each contain one element with the value 0. • False: the function returns the position in the string where the match begins. Default.

Usage

This function finds the first occurrence of a regular expression in a string. To find the second and subsequent instances of the expression or of subexpressions in it, you call this function more than once, each time with a different start position. To determine the next start position, use the `returnsubexpressions` parameter, and add the value returned in the first element of the length array to the value in the first element of the position array.

Example

```
<h3>REFind Example</h3>
<p>This example shows the use of the REFind function with and without the
    <i>returnsubexpressions</i> parameter set to True.
    If you do not use the <i>returnsubexpressions</i> parameter,
    REFind returns the position of the first occurrence of a regular
    expression in a string starting from the specified position.
    Returns 0 if no occurrences are found.</p>

<p>REFind("a+c+", "abcaaccdd"):
<cfoutput>#REFind("a+c+", "abcaaccdd")#</cfoutput></p>
<p>REFind("a+c*", "abcaaccdd"):
<cfoutput>#REFind("a+c*", "abcaaccdd")#</cfoutput></p>
<p>REFind("[[:upper:]]", "abcaacCDD"):
<cfoutput>#REFind("[[:upper:]]", "abcaacCDD")#</cfoutput></p>
<p>REFind("[\?&]rep = ", "report.cfm?rep = 1234&u = 5"):
    <cfoutput>#REFind("[\?&]rep = ", "report.cfm?rep = 1234&u = 5")#
    </cfoutput>
</p>
<!--- Set startPos to one; returnMatchedSubexpressions = TRUE --->
<hr size = "2" color = "#0000A0">
<p>If you use the <i>returnssubexpression</i> parameter, REFind returns the
    position and length of the first occurrence of a regular expression
    in a string starting from the specified position. The position and
    length variables are stored in a structure. To access position and length
    information, use the keys <i>pos</i> and <i>len</i>, respectively.</p>
<cfset teststring = "The cat in the hat hat came back!">
<p>The string in which the function is to search is:
<cfoutput><b>#teststring#</b></cfoutput>.</p>
<p>The first call to REFind to search this string is:
        <b>REFind("[A-Za-z]+",testString,1,"TRUE")</b></p>
<p>This function returns a structure that contains two arrays: pos and len.</p>
<p>To create this structure you can use a CFSET statement, for example: </p>
&lt;CFSET st = REFind("[[:alpha:]]",testString,1,"TRUE")&gt;
<cfset st = REFind("[[:alpha:]]",testString,1,"TRUE")>
<p>
    <cfoutput>
    The number of elements in each array: #ArrayLen(st.pos)#.
    </cfoutput></p>
<p><b>The number of elements in the pos and len arrays is always one
    if you do not use parentheses in the regular expression.</b></p>
<p>The value of st.pos[1] is: <cfoutput>#st.pos[1]#.</cfoutput></p>
<p>The value of st.len[1] is: <cfoutput>#st.len[1]#.</cfoutput></p>
<p>
<cfoutput>
    Substring is <b>[#Mid(testString,st.pos[1],st.len[1])#]</B>
    </cfoutput></p>
<hr size = "2" color = "#0000A0">
<p>However, if you use parentheses in the regular expression, the first
    element contains the position and length of the first instance
    of the whole expression. The position and length of the first instance
    of each parenthesized subexpression within is included in additional
    array elements.</p>
```

```
<p>For example:
&lt;CFSET st1 = REFind("([[:alpha:]])[ ]+(\1)",testString,1,"TRUE")&gt;</p>
<cfset st1 = REFind("([[:alpha:]]+)[ ]+(\1)",testString,1,"TRUE")>
<p>The number of elements in each array is <cfoutput>#ArrayLen(st1.pos)#
    </cfoutput>.</p>
<p>First whole expression match; position is
    <cfoutput>#st1.pos[1]#;
        length is #st1.len[1]#; whole expression match is
        <B>[#Mid(testString,st1.pos[1],st1.len[1])#]</B>
    </cfoutput></p>
<p>Subsequent elements of the arrays provide the position and length of
    the first instance of each parenthesized subexpression therein.</p>
 <cfloop index = "i" from = "2" to = "#ArrayLen(st1.pos)#">
    <p><cfoutput>Position is #st1.pos[i]#; Length is #st1.len[i]#;
    Substring is <B>[#Mid(testString,st1.pos[i],st1.len[i])#]
    </B></cfoutput></p>
</cfloop><br>
```

REFindNoCase

Description

Uses a regular expression (RE) to search a string for a pattern, starting from a specified position. The search is case-insensitive.

For more information on regular expressions, including escape sequences, anchors, and modifiers, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.

Returns

Depends on the value of the `returnsubexpressions` parameter:

- If `returnsubexpressions = "False"`:
  - The position in the string where the match begins
  - 0, if the regular expression is not matched in the string
- If `returnsubexpressions = "True"`: a structure that contains two arrays, `len` and `pos`. The array elements are as follows:
  - If the regular expression is found in the string, the first element of the `len` and `pos` arrays contains the length and position, respectively, of the first match of the entire regular expression.
  - If the regular expression contains parentheses that group subexpressions, each subsequent array element contains the length and position, respectively, of the first occurrence of each group.
  - If the regular expression is not found in the string, the first element of the `len` and `pos` arrays contains 0.

Category

String functions

Function syntax

`REFindNoCase(`*`reg_expression, string [, start, returnsubexpressions`*`]`)`

See also

`Find`, `FindNoCase`, `REFind`, `REReplace`, `REReplaceNoCase`

Parameters

Parameter	Description
`reg_expression`	Regular expression for which to search. Case-insensitive. For more information, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.
`string`	A string or a variable that contains one. String in which to search.
`start`	Optional. A positive integer or a variable that contains one. Position at which to start search. The default value is 1.
`returnsubexpressions`	Optional. Boolean. Whether to return substrings of reg_expression, in arrays named `len` and `pos`: • True: if the regular expression is found, the first array element contains the length and position, respectively, of the first match. If the regular expression contains parentheses that group subexpressions, each subsequent array element contains the length and position, respectively, of the first occurrence of each group. If the regular expression is not found, the arrays each contain one element with the value 0. • False: the function returns the position in the string where the match begins. Default.

Usage

This function finds the first occurrence of a regular expression in a string. To find the second and subsequent instances of the expression or of subexpressions in it, you call this function more than once, each time with a different start position. To determine the next start position, use the `returnsubexpressions` parameter, and add the value returned in the first element of the length array to the value in the first element of the position array.

Example

```
<h3>REFindNoCase Example</h3>
<p>This example demonstrates the use of the REFindNoCase function with and
    without the <i>returnsubexpressions</i> parameter set to True.</p>
<p>If you do not use the <i>returnsubexpressions</i> parameter, REFindNoCase
    returns the position of the first occurrence of a regular expression
    in a string starting from the specified position. Returns 0 if no
    occurrences are found. </p>
<p>REFindNoCase("a+c+", "abcaaccdd"):
<cfoutput>#REFindNoCase("a+c+", "abcaaccdd")#</cfoutput></p>
<p>REFindNoCase("a+c*", "abcaaccdd"):
<cfoutput>#REFindNoCase("a+c*", "abcaaccdd")#</cfoutput></p>
<p>REFindNoCase("[[:alpha:]]+", "abcaacCDD"):
<cfoutput>#REFindNoCase("[[:alpha:]]+", "abcaacCDD")#</cfoutput></p>
<p>REFindNoCase("[\?&]rep = ", "report.cfm?rep = 1234&u = 5"):
<cfoutput>#REFindNoCase("[\?&]rep = ", "report.cfm?rep = 1234&u = 5")#
    </cfoutput></p>
<!--- Set startPos to one; returnMatchedSubexpressions = True --->
<hr size = "2" color = "#0000A0">
<p>If you do use the <i>returnssubexpression</i> parameter, REFindNoCase returns
    the position and length of the first occurrence of a regular expression
    in a string starting from the specified position. The position and length
    variables are stored in a structure. To access position and length
    information, use the keys <i>pos</i> and <i>len</i>, respectively.</p>

<cfset teststring = "The cat in the hat hat came back!">
<p>The string in which the function is to search is:
    <cfoutput><b>#teststring#</b></cfoutput>.</p>
<p>The first call to REFindNoCase to search this string is:
<b>REFindNoCase("[[:alpha:]]+",testString,1,"True")</b></p>
<p>This function returns a structure that contains two arrays: pos and len.</p>
<p>To create this structure you can use a CFSET statement,
    for example:</p>
&lt;CFSET st = REFindNoCase("[[:alpha:]]+",testString,1,"True")&gt;
<cfset st = REFindNoCase("[[:alpha:]]+",testString,1,"True")>
<p>
    <cfoutput>
    The number of elements in each array: #ArrayLen(st.pos)#.
    </cfoutput></p>
<p><b>The number of elements in the pos and len arrays will always be one,
    if you do not use parentheses to denote subexpressions in the regular
    expression.</b></p>
<p>The value of st.pos[1] is: <cfoutput>#st.pos[1]#.</cfoutput></p>
<p>The value of st.len[1] is: <cfoutput>#st.len[1]#.</cfoutput></p>
<p>
    <cfoutput>
    Substring is <b>[#Mid(testString,st.pos[1],st.len[1])#]</B>
    </cfoutput></p>
<hr size = "2" color = "#0000A0">
<p>However, if you use parentheses to denote subexpressions in the regular
    expression, the first element contains the position and length of
    the first instance of the whole expression. The position and length
    of the first instance of each subexpression within will be included
```

```
    in additional array elements.</p>
<p>For example:
&lt;CFSET st1 = REFindNoCase("([[:alpha:]]+)[ ]+(\1)",testString,1,"True")&gt;</p>
<cfset st1 = REFindNoCase("([[:alpha:]]+)[ ]+(\1)",testString,1,"True")>

<p>The number of elements in each array is
<cfoutput>
    #ArrayLen(st1.pos)#
</cfoutput>.</p>

<p>First whole expression match; position is
<cfoutput>
    #st1.pos[1]#; length is #st1.len[1]#;
    whole expression match is <B>[#Mid(testString,st1.pos[1],st1.len[1])#]</B>
</cfoutput></p>

<p>Subsequent elements of the arrays provide the position and length of the
    first instance of each parenthesized subexpression therein.</p>
 <cfloop index = "i" from = "2" to = "#ArrayLen(st1.pos)#">
    <p><cfoutput>Position is #st1.pos[i]#; Length is #st1.len[i]#;
    Substring is <B>[#Mid(testString,st1.pos[i],st1.len[i])#]</B>
    </cfoutput></p>
</cfloop><br>
```

REMatch

Description

Uses a regular expression (RE) to search a string for a pattern, starting from a specified position. The search is case sensitive.

For more information on regular expressions, including escape sequences, anchors, and modifiers, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.

Returns

An array of strings that match the expression.

Category

String functions

Function syntax

```
REMatch(reg_expression, string)
```

See also

Find, FindNoCase, REFind, REReplace, REReplaceNoCase, REMatchNoCase

Parameters

Parameter	Description
reg_expression	Regular expression for which to search. Case sensitive. For more information, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.
string	A string or a variable that contains one. String in which to search.

Usage

This function finds all occurrence of a regular expression in a string.

Example

```
<!--- Find all the URLs in a web page retrieved via cfhttp:. --->
<!--- The search is case sensitive. --->
result = REMatch("https?://([-\w\.]+)+(:\d+)?(/([\w/_\.]*(\?\S+)?)?)?",
cfhttp.filecontent);
```

REMatchNoCase

Description

Uses a regular expression (RE) to search a string for a pattern, starting from a specified position. The search is case-insensitive.

For more information on regular expressions, including escape sequences, anchors, and modifiers, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.

Returns

An array of strings that match the expression.

Category

String functions

Function syntax

`REMatchNoCase(reg_expression, string)`

See also

`Find`, `FindNoCase`, `REFind`, `REReplace`, `REReplaceNoCase`, `REMatch`

Parameters

Parameter	Description
`reg_expression`	Regular expression for which to search. Case-insensitive. For more information, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.
`string`	A string or a variable that contains one. String in which to search.

Example

```
<!--- Find all the URLs in a web page retrieved via cfhttp:. --->
result = REMatch("https?://([-\w\.]+)+(:\d+)?(/([\w/_\.]*(\?\S+)?)?)?",
cfhttp.filecontent);
```

ReleaseComObject

Description
Releases a COM Object and frees up resources that it used.

Returns
Nothing.

Category
Extensibility functions

Function syntax
`ReleaseComObject(objectName)`

See also
`CreateObject`, `cfobject`

History
ColdFusion MX 6.1: Added this function.

Parameters

Parameter	Description
`objectName`	Variable name of a COM object that was created using the `CreateObject` function or `cfobject` tag.

Usage
This function forcefully terminates and releases the specified COM object and all COM objects that it created. Use this function when the object is no longer in use, to quickly free up resources. If the COM object has a method, such as a `quit` method, that terminates the program, call this method before you call the `ReleaseComObject` function.

This function can improve processing efficiency, but is not required for an application to work. If you do not use this function, the Java garbage collection mechanism eventually frees the resources. If you use this function on an object that is in use, the object is prematurely released and your application will get exceptions.

Example
```
<h3>ReleaseComObject Example</h3>
    <cfscript>
    obj = CreateObject("Com", "excel.application.9");
    //code that uses the object goes here???I'd like to fill this in with something???
    obj.quit();
    ReleaseComObject(obj);
    </cfscript>
```

RemoveChars

Description

Removes characters from a string.

Returns

A copy of the string, with *count* characters removed from the specified start position. If no characters are found, returns zero.

Category

String functions

Function syntax

`RemoveChars(string, start, count)`

See also

`Insert`, `Len`

Parameters

Parameter	Description
`string`	A string or a variable that contains one. String in which to search.
`start`	A positive integer or a variable that contains one. Position at which to start search.
`count`	Number of characters to remove.

Example

```
<h3>RemoveChars Example</h3>
Returns a string with <I>count</I> characters removed from the
    start position. Returns 0 if no characters are found.

<cfif IsDefined("FORM.myString")>
    <cfif (FORM.numChars + FORM.start) GT Len(FORM.myString)>
        <p>Your string is only <cfoutput>#Len(FORM.myString)#
         </cfoutput> characters long.
        Please enter a longer string, select fewer characters to remove or
        begin earlier in the string.
    <cfelse>
        <cfoutput>
        <p>Your original string: #FORM.myString#
        <p>Your modified string:#RemoveChars(FORM.myString,
    FORM.start, FORM.numChars)#
        </cfoutput>
    </cfif>
</cfif>
```

RepeatString

Description

Creates a string that contains a specified number of repetitions of the specified string.

Returns

A string.

Category

String functions

Function syntax

`RepeatString(string, count)`

See also

`CJustify`, `LJustify`, `RJustify`

Parameters

Parameter	Description
`string`	A string or a variable that contains one.
`count`	Number of repeats.

Example

```
<h3>RepeatString Example</h3>
<p>RepeatString returns a string created from <I>string</I>, repeated
    a specified number of times.
<ul>
    <li>RepeatString("-", 10): <cfoutput>#RepeatString("-", 10)#</cfoutput>
    <li>RepeatString("&lt;BR&gt;", 3): <cfoutput>#RepeatString("<br>", 3)#
    </cfoutput>
    <li>RepeatString("", 5): <cfoutput>#RepeatString("", 5)#</cfoutput>
    <li>RepeatString("abc", 0): <cfoutput>#RepeatString("abc", 0)#</cfoutput>
    <li>RepeatString("Lorem Ipsum", 2):
     <cfoutput>#RepeatString("Lorem Ipsum", 2)#</cfoutput>
</ul>
```

Replace

Description

Replaces occurrences of *substring1* in a string with *substring2*, in a specified scope. The search is case-sensitive.

Returns

The string, after making replacements.

Category

String functions

Function syntax

```
Replace(string, substring1, substring2 [, scope ])
```

See also

Find, REFind, ReplaceNoCase, ReplaceList, REReplace

Parameters

Parameter	Description
string	A string or a variable that contains one. String in which to search.
substring1	A string or a variable that contains one. String for which to search
substring2	String that replaces substring1
scope	• one: replaces the first occurrence (default) • all: replaces all occurrences

Usage

To remove a string, specify the empty string ("") as *substring2*.

You do not need to escape comma characters in strings. For example, the following code deletes the commas from the sentence:

```
replace("The quick brown fox jumped over the lazy cow, dog, and cat.",",","","","All")
```

Example

```
<h3>Replace Example</h3>

<p>The Replace function returns <I>string</I> with <I>substring1</I>
    replaced by <I>substring2</I> in the specified scope. This
    is a case-sensitive search.

<cfif IsDefined("FORM.MyString")>
<p>Your original string, <cfoutput>#FORM.MyString#</cfoutput>
<p>You wanted to replace the substring <cfoutput>#FORM.MySubstring1#
    </cfoutput>
    with the substring <cfoutput>#FORM.MySubstring2#</cfoutput>.
<p>The result: <cfoutput>#Replace(FORM.myString,
    FORM.MySubstring1, FORM.mySubString2)#</cfoutput>
</cfif>
```

ReplaceList

Description

Replaces occurrences of the elements from a delimited list in a string with corresponding elements from another delimited list. The search is case-sensitive.

Returns

A copy of the string, after making replacements.

Category

List functions, String functions

Function syntax

`ReplaceList(string, list1, list2)`

See also

`Find`, `REFind`, `Replace`, `REReplace`

Parameters

Parameter	Description
`string`	A string, or a variable that contains one, within which to replace substring
`list1`	Comma-delimited list of substrings for which to search
`list2`	Comma-delimited list of replacement substrings

Usage

The list of substrings to replace is processed sequentially. If a *list1* element is contained in *list2* elements, recursive replacement might occur. The second example shows this.

Example

```
<p>The ReplaceList function returns <I>string</I> with
<I>substringlist1</I> (e.g. "a,b") replaced by <I>substringlist2</I>
    (e.g. "c,d") in the specified scope.
<cfif IsDefined("FORM.MyString")>
<p>Your original string, <cfoutput>#FORM.MyString#</cfoutput>
<p>You wanted to replace the substring <cfoutput>#FORM.MySubstring1#
 </cfoutput>
with the substring <cfoutput>#FORM.MySubstring2#</cfoutput>.
<p>The result: <cfoutput>#Replacelist(FORM.myString,
FORM.MySubstring1, FORM.mySubString2)#</cfoutput>
</cfif>
<form action = "replacelist.cfm" method="post">
<p>String 1
<br><input type = "Text" value = "My Test String" name = "MyString">
<p>Substring 1 (find this list of substrings)
<br><input type = "Text" value = "Test, String" name = "MySubstring1">
<p>Substring 2 (replace with this list of substrings)
<br><input type = "Text" value = "Replaced, Sentence" name = "MySubstring2">
<p><input type = "Submit" value = "Replace and display" name = "">
</form>

<h3>Replacelist Example Two</h3>
<cfset stringtoreplace = "The quick brown fox jumped over the lazy dog.">
<cfoutput>
```

```
    #ReplaceList(stringtoreplace,"dog,brown,fox,black", "cow,black,ferret,white")#
</cfoutput>
```

ReplaceNoCase

Description

Replaces occurrences of *substring1* with *substring2*, in the specified scope. The search is case-insensitive.

Returns

A copy of the string, after making replacements.

Category

String functions

Function syntax

```
ReplaceNoCase(string, substring1, substring2 [, scope ])
```

See also

Find, REFind, Replace, ReplaceList, REReplace

Parameters

Parameter	Description
string	A string (or variable that contains one) within which to replace substring.
substring1	String (or variable that contains one) to replace, if found.
substring2	String (or variable that contains one) that replaces substring1.
scope	• one: replaces the first occurrence (default). • all: replaces all occurrences.

Example

```
<h3>ReplaceNoCase Example</h3>
<p>The ReplaceNoCase function returns <I>string</I> with <I>substring1</I>
    replaced by <I>substring2</I> in the specified scope.
    The search/replace is case-insensitive.

<cfif IsDefined("FORM.MyString")>
<p>Your original string, <cfoutput>#FORM.MyString#</cfoutput>
<p>You wanted to replace the substring <cfoutput>#FORM.MySubstring1#
 </cfoutput>
with the substring <cfoutput>#FORM.MySubstring2#</cfoutput>.
<p>The result: <cfoutput>#ReplaceNoCase(FORM.myString,
FORM.MySubstring1, FORM.mySubString2)#</cfoutput>
</cfif>
```

REReplace

Description

Uses a regular expression (RE) to search a string for a string pattern and replace it with another. The search is case-sensitive.

Returns

If the *scope* parameter is set to `one`, returns a string with the first occurrence of the regular expression replaced by the value of *substring*.

If the *scope* parameter is set to `all`, returns a string with all occurrences of the regular expression replaced by the value of *substring*.

If the function finds no matches, it returns a copy of the string unchanged.

Category

String functions

Function syntax

```
REReplace(string, reg_expression, substring [, scope ])
```

See also

`REFind`, `Replace`, `ReplaceList`, `REReplaceNoCase`

History

ColdFusion MX: Added supports for the following special codes in a replacement substring, to control case conversion:

- \u - uppercase the next character
- \l - lowercase the next character
- \U - uppercase until \E
- \L - lowercase until \E
- \E - end \U or \L

For more information on new features, see `REFind`.

Parameters

Parameter	Description
`string`	A string or a variable that contains one. String within which to search.
`reg_expression`	Regular expression to replace. The search is case-sensitive.
`substring`	A string or a variable that contains one. Replaces `reg_expression`.
`scope`	• one: replaces the first occurrence (default). • all: replaces all occurrences.

Usage

For details on using regular expressions, see “Using Regular Expressions in Functions” on page 107 in the *ColdFusion Developer’s Guide*.

Example

```
<p>The REReplace function returns <i>string</i> with a regular expression replaced
    with <i>substring</i> in the specified scope. Case-sensitive search.
<p>REReplace("CABARET","C|B","G","ALL"):
<cfoutput>#REReplace("CABARET","C|B","G","ALL")#</cfoutput>
<p>REReplace("CABARET","[A-Z]","G","ALL"):
<cfoutput>#REReplace("CABARET","[A-Z]","G","ALL")#</cfoutput>
<p>REReplace("I love jellies","jell(y|ies)","cookies"):
<cfoutput>#REReplace("I love jellies","jell(y|ies)","cookies")#
 </cfoutput>
<p>REReplace("I love jelly","jell(y|ies)","cookies"):
<cfoutput>#REReplace("I love jelly","jell(y|ies)","cookies")#</cfoutput>
```

REReplaceNoCase

Description

Uses a regular expression to search a string for a string pattern and replace it with another. The search is case-insensitive.

Returns

- If `scope = "one"`: returns a string with the first occurrence of the regular expression replaced by the value of *substring*.
- If `scope = "all"`: returns a string with all occurrences of the regular expression replaced by the value of *substring*.
- If the function finds no matches: returns a copy of the string, unchanged.

Category

String functions

Function syntax

`REReplaceNoCase(`*`string, reg_expression, substring`* `[, `*`scope`*` ])`

See also

`REFind`, `REFindNoCase`, `Replace`, `ReplaceList`

History

ColdFusion MX: Changed behavior: this function inserts the following special characters in regular expression replacement strings, to control case conversion: \u, \U, \l, \L, and \E. If any of these strings is present in a ColdFusion 5 application, you must insert a backslash before it (for example, change "\u" to "\\u").

Parameters

Parameter	Description
`string`	A string or a variable that contains one.
`reg_expression`	Regular expression to replace. For more information, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.
`substring`	A string or a variable that contains one. Replaces `reg_expression`.
`scope`	• one: replaces the first occurrence of the regular expression. Default. • all: replaces all occurrences of the regular expression.

Usage

For details on using regular expressions, see "Using Regular Expressions in Functions" on page 107 in the *ColdFusion Developer's Guide*.

Example

```
<p>The REReplaceNoCase function returns <i>string</i> with a regular
    expression replaced with <i>substring</i> in the specified scope.
    This is a case-insensitive search.
<p>REReplaceNoCase("cabaret","C|B","G","ALL"):
<cfoutput>#REReplaceNoCase("cabaret","C|B","G","ALL")#</cfoutput>
<p>REReplaceNoCase("cabaret","[A-Z]","G","ALL"):
<cfoutput>#REReplaceNoCase("cabaret","[A-Z]","G","ALL")#</cfoutput>
<p>REReplaceNoCase("I LOVE JELLIES","jell(y|ies)","cookies"):
```

```
<cfoutput>#REReplaceNoCase("I LOVE JELLIES","jell(y|ies)","cookies")#
</cfoutput>
<p>REReplaceNoCase("I LOVE JELLY","jell(y|ies)","cookies"):
<cfoutput>#REReplaceNoCase("I LOVE JELLY","jell(y|ies)","cookies")#
    </cfoutput>
```

Reverse

Description
Reverses the order of items, such as the characters in a string or the digits in a number.

Returns
A copy of *string,* with the characters in reverse order.

Category
String functions

Function syntax
`Reverse(string)`

See also
`Left`, `Mid`, `Right`

Parameters

Parameter	Description
`string`	A string or a variable that contains one

Usage
You can call this function on a number with code such as the following:

```
<cfoutput>reverse(6*2) equals #reverse(6*2)#<cfoutput>
```

This code outputs the following:

```
reverse(6*2) equals 21
```

Example
```
<h3>Reverse Example</h3>

<p>Reverse returns your string with the positions of the characters reversed.
<cfif IsDefined("FORM.myString")>
    <cfif FORM.myString is not "">
        <p>Reverse returned:
        <cfoutput>#Reverse(FORM.myString)#</cfoutput>
    <cfelse>
        <p>Please enter a string to be reversed.
    </cfif>
</cfif>

<form action = "reverse.cfm">
<p>Enter a string to be reversed:
<input type = "Text" name = "MyString">
<p><input type = "Submit" name = "">
</form>
```

Right

Description

Gets a specified number of characters from a string, beginning at the right.

Returns the specified number of characters from the end (or *right* side) of the specified string.

Returns

- If the length of the string is greater than or equal to *count*, the rightmost *count* characters of the string
- If *count* is greater than the length of the string, the whole string
- If *count* is greater than 1, and the string is empty, an empty string

Category

String functions

Function syntax

`Right(string, count)`

See also

`Left`, `Mid`, `Reverse`

Parameters

Parameter	Description
`string`	A string or a variable that contains one.
`count`	A positive integer that specifies the number of characters to return.

Example

```
<!--- Simple Right Example--->
<cfoutput>
#Right("See the quick red fox jump over the fence", 9)#
<br>
#Right("ColdFusion", 6)#
</cfoutput>

<!--- Right Example using form input --->
<h3>Right Example</h3>
<cfif IsDefined("Form.MyText")>
<!--- If len returns 0 (zero), then show error message. --->
    <cfif Len(FORM.myText)>
        <cfif Len(FORM.myText) LTE FORM.RemoveChars>
            <cfoutput><p style="color: red; font-weight: bold">Your string
            #FORM.myText# only has #Len(FORM.myText)# characters. You cannot output
            the #FORM.removeChars# rightmost characters of this string because it
            is not long enough.</p></cfoutput>
        <cfelse>
            <cfoutput><p>Your original string: <strong>#FORM.myText#</strong>
            <p>Your changed string, showing only the <strong>#FORM.removeChars#
            </strong> rightmost characters:
            <strong>#right(Form.myText, FORM.removeChars)#</strong></p>
            </cfoutput>
        </cfif>
    <cfelse>
        <p style="color: red; font-weight: bold">Please enter a string of more
```

```
        than 0 (zero) characters.</p>
    </cfif>
</cfif>

<form action="<cfoutput>#CGI.ScriptName#</cfoutput>" method="POST">
<p>Type in some text<br />
<input type="Text" name="myText"></p>
<p>How many characters from the right do you want to show?
<select name="RemoveChars">
<option value="1">1
<option value="3" selected>3
<option value="5">5
<option value="7">7
<option value="9">9</select>
<input type="Submit" name="Submit" value="Remove characters"></p>
</form>
```

RJustify

Description

Right justifies characters of a string.

Returns

A copy of a string, right-justified in the specified field length.

Category

Display and formatting functions, String functions

Function syntax

`RJustify(string, length)`

See also

`CJustify`, `LJustify`

Parameters

Parameter	Description
`string`	A string enclosed in quotation marks, or a variable that contains one.
`length`	A positive integer or a variable that contains one. Length of field in which to justify string.

Example

```
<!--- This example shows how to use RJustify --->
<cfparam name = "jstring" default = "">

<cfif IsDefined("FORM.justifyString")>
    <cfset jstring = rjustify(FORM.justifyString, 35)>
</cfif>
<html>
<head>
<title>RJustify Example</title>
</head>
<body>
<h3>RJustify Function</h3>
<p>Enter a string. It will be right justified within the sample field

<form action = "rjustify.cfm">
<p><input type = "Text" value = "<cfoutput>#jString#</cfoutput>"
    size = 35 name = "justifyString">

<p><input type = "Submit" name = ""> <input type = "reset">
</form>
```

Round

Description
Rounds a number to the closest integer that is larger than the input parameter.

Returns
An integer.

Category
Mathematical functions

Function syntax
`Round(number)`

See also
`Ceiling`, `Fix`, `Int`

Parameters

Parameter	Description
`number`	Number to round

Usage
Use this function to round a number. This function rounds numbers that end with .5 up to the nearest integer. It rounds 3.5 to 4 and -3.5 to -3.

Example
```
<h3>Round Example</h3>
<p>This function rounds a number to the closest integer.
<ul>
 <li>Round(7.49) : <cfoutput>#Round(7.49)#</cfoutput>
 <li>Round(7.5) : <cfoutput>#Round(7.5)#</cfoutput>
 <li>Round(-10.775) : <cfoutput>#Round(-10.775)#</cfoutput>
 <li>Round(-35.5) : <cfoutput>#Round(-35.5)#</cfoutput>
 <li>Round(35.5) : <cfoutput>#Round(35.5)#</cfoutput>
 <li>Round(1.2345*100)/100 : <cfoutput>#Round(1.2345*100)/100#</cfoutput>
</ul>
```

RTrim

Description

Removes spaces from the end of a string.

Returns

A copy of *string*, after removing trailing spaces.

Category

String functions

Function syntax

`RTrim(string)`

See also

`LTrim`, `Trim`

Parameters

Parameter	Description
`string`	A string or a variable that contains one

Example

```
<h3>RTrim Example</h3>

<cfif IsDefined("FORM.myText")>
<cfoutput>
<pre>
Your string:"#FORM.myText#"
Your string:"#RTrim(FORM.myText)#"
(right trimmed)
</pre>
</cfoutput>
</cfif>

<form action = "Rtrim.cfm" method="post">
<p>Enter some text. It will be modified by Rtrim to remove spaces from the right.
<p><input type = "Text" name = "myText" value = "TEST ">

<p><input type = "Submit" name = "">
</form>
```

Second

Description

Extracts the ordinal for the second from a date/time object.

Returns

An integer in the range 0–59.

Category

Date and time functions

Function syntax

`Second(date)`

See also

`DatePart`, `Hash`, `Minute`

Parameters

Parameter	Description
`date`	A date/time object

Usage

When passing a date/time object as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Example

```
<!--- This example shows the use of Hour, Minute, and Second --->
<h3>Second Example</h3>
<cfoutput>
The time is currently #TimeFormat(Now())#.
We are in hour #Hour(Now())#, Minute #Minute(Now())#
and Second #Second(Now())# of the day.
</cfoutput>
```

SendGatewayMessage

Description

Sends an outgoing message through a ColdFusion event gateway.

Returns

String. The value returned depends on the gateway type.

Category

Extensibility functions

Function syntax

`SendGatewayMessage(gatewayID, data)`

See also

`GetGatewayHelper`; "IM gateway message sending commands" on page 1377, "SMS Gateway CFEvent structure and commands" on page 1403, "CFML event gateway SendGatewayMessage data parameter" on page 1413, and "Sending a message using the SendGatewayMessage function" on page 1075 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`gatewayID`	Identifier of the gateway to send the message. Must be the Gateway ID of one of the ColdFusion event gateway instances configured on the ColdFusion Administrator Event Gateways section's Gateways page.
`data`	A ColdFusion structure. The contents of the structure depend on the event gateway type, but typically include a MESSAGE field that contains the message to send and a field that contains the destination address.

Usage

The `SendGatewayMessage` function calls the specified gateway's `outgoingMessage` method. The value returned by the function depends on the gateway type. The following table describes the return values for standard ColdFusion gateway types:

Gateway type	Return values
Asynchronous CFML	If the message was queued for delivery to the CFC, returns True; False, otherwise.
Lotus SameTime	If the message or command was successful, returns OK.
	If an error occurred, returns a string indicating the cause.
SMS	If the gateway is in asynchronous mode, returns the empty string immediately.
	If the gateway is in synchronous mode, the function waits for the gateway to return a response. If the message was succesfully sent to the short message service center (SMSC), returns the message ID from the SMSC. If an error occured, returns a string indicating the cause.
XMPP	If the message or command was successful, returns OK
	If an error occured, returns a string indicating the cause.

Example

The following example uses an instance of the CFML gateway to log messages asynchronously to a file. To use this example, you must configure an instance of the CFML gateway with the name "Asynch Logger" in the ColdFusion Administrator. This gateway instance must use a CFC that takes the message and logs it. For sample CFC code, see "Using the CFML event gateway for asynchronous CFCs" on page 1077 in the *ColdFusion Developer's Guide*.

```
Sending an event to the CFML event gateway that is registered in the
    ColdFusion Administrator as Asynch Logger.<br>
<cfscript>
    status = false;
    props = structNew();
    props.message = "Replace me with a variable with data to log";
    status = SendGatewayMessage("Asynch Logger", props);
    if (status IS True) WriteOutput('Event Message "#props.message#" has been sent.');
</cfscript>
```

SerializeJSON

Description
Converts ColdFusion data into a JSON (JavaScript Object Notation) representation of the data.

Returns
A string that contains a JSON representation of the parameter value.

Category
Conversion functions

Syntax
SerializeJSON(*var*[, *serializeQueryByColumns*])

See also
DeserializeJSON, IsJSON, cfajaxproxy, "Using data interchange formats" on page 668 in the *ColdFusion Developer's Guide*, http://www.json.org

History
ColdFusion 8: Added function

Parameters

Parameter	Description
var	A ColdFusion data value or variable that represents one.
serializeQueryByColumns	A Boolean value that specifies how to serialize ColdFusion queries. • false (the default): Creates an object with two entries: an array of column names and an array of row arrays. This format is required by the HTML format cfgrid tag. • true: Creates an object that corresponds to WDDX query format. For more information, see the Usage section.

Usage
This function is useful for generating JSON format data to be consumed by an Ajax application.

The SerializeJSON function converts ColdFusion dates and times into strings that can be easily parsed by the JavaScript Date object. The strings have the following format:

```
MonthName, DayNumber Year Hours:Minutes:Seconds
```

The SerializeJSON function converts the ColdFusion date time object for October 3, 2007 at 3:01 PM, for example, into the JSON string "October, 03 2007 15:01:00".

The SerializeJSON function with a false serializeQueryByColumns parameter (the default) converts a ColdFusion query into a row-oriented JSON Object with the following elements:

Element	Description
COLUMNS	An array of the names of the columns.
DATA	A two-dimensional array, where: • Each entry in the outer array corresponds to a row of query data. • Each entry in the inner arrays is a column field value in the row, in the same order as the `COLUMNS` array entries.

For example, the `SerializeJSON` function with a `serializeQueryByColumns` parameter value of `false` converts a ColdFusion query with two columns, City, and State, and two rows of data into following format:

```
{"COLUMNS":["CITY","STATE"],"DATA":[["Newton","MA"],["San Jose","CA"]]}
```

The `SerializeJSON` function with a `serializeQueryByColumns` parameter value of `true` converts a ColdFusion query into a column-oriented JSON Object that is equivalent to the WDDX query representation. The JSON Object has three elements:

Element	Description
ROWCOUNT	The number of rows in the query.
COLUMNS	An array of the names of the columns.
DATA	An Object with the following: • The keys are the query column names • The values are arrays that contain the column data

The `SerializeJSON` function with a `serializeQueryByColumns` parameter value of `true` converts a ColdFusion query with two columns, City, and State, and two rows of data into following format:

```
{"ROWCOUNT":2, "COLUMNS":["CITY","STATE"],"DATA":{"City":["Newton","San
Jose"],"State":["MA","CA"]}}
```

Note: *The `SerializeJSON` function generates an error if you try to convert binary data into JSON format.*

The `SerializeJSON` function converts all other ColdFusion data types to the corresponding JSON types. It converts structures to JSON Objects, arrays to JSON Arrays, numbers to JSON Numbers, and strings to JSON Strings.

Note: *ColdFusion internally represents structure key names using all-uppercase characters, and, therefore, serializes the key names to all-uppercase JSON representations. Any JavaScript that handles JSON representations of ColdFusion structures must use all-uppercase structure key names, such as CITY or STATE. You also use the all-uppercase names `COLUMNS` and `DATA` as the keys for the two arrays that represent ColdFusion queries in JSON format.*

Example

This example creates a JSON-format data feed with simple weather data for two cities. The data feed is in the form of a JavaScript application that consists of a single function call that has a JSON Object as its parameter. The example code does the following:

1 Creates a query object with two rows of weather data. Each row has a city, the current temperature, and an array of forecast structures, with each with the high, low, and weather prediction for one day. This data would normally be provided by a data source; to keep the example simple, the example uses the same prediction for all cites and days.

2 Converts the query to a JSON format string and surrounds it in a JavaScript function call.

3 Writes the result to the output.

If you view this page in your browser, you see the resulting JavaScript function and JSON parameter. To use the results of this page in an application, put this file and the example for the DeserializeJSON function in an appropriate location under your ColdFusion web root, replace the URL in the DeserializeJSON example code with the correct URL for this page, and run the DeserializeJSON example.

```
<!--- Generate a clean feed by suppressing white space and debugging
         information. --->
<cfprocessingdirective suppresswhitespace="yes">
<cfsetting showdebugoutput="no">
<!--- Generate the JSON feed as a JavaScript function. --->
<cfcontent type="application/x-javascript">

<cfscript>
    // Construct a weather query with information on cities.
    // To simplify the code, we use the same weather for all cities and days.
    // Normally this information would come from a data source.
    weatherQuery = QueryNew("City, Temp, Forecasts");
    QueryAddRow(weatherQuery, 2);
    theWeather=StructNew();
    theWeather.High=73;
    theWeather.Low=53;
    theWeather.Weather="Partly Cloudy";
    weatherArray=ArrayNew(1);
    for (i=1; i<=5; i++) weatherArray[i]=theWeather;
    querySetCell(weatherQuery, "City", "Newton", 1);
    querySetCell(weatherQuery, "Temp", "65", 1);
    querySetCell(weatherQuery, "ForeCasts", weatherArray, 1);
    querySetCell(weatherQuery, "City", "San Jose", 2);
    querySetCell(weatherQuery, "Temp", 75, 2);
    querySetCell(weatherQuery, "ForeCasts", weatherArray, 2);

    // Convert the query to JSON.
    // The SerializeJSON function serializes a ColdFusion query into a JSON
    // structure.
    theJSON = SerializeJSON(weatherQuery);

    // Wrap the JSON object in a JavaScript function call.
    // This makes it easy to use it directly in JavaScript.
    writeOutput("onLoad( "&theJSON&" )");
</cfscript>
</cfprocessingdirective>
```

SetEncoding

Description

Sets the character encoding (character set) of Form and URL scope variable values; used when the character encoding of the input to a form, or the character encoding of a URL, is not in UTF-8 encoding.

Returns

None

Category

International functions, System functions

Function syntax

`SetEncoding(scope_name,charset)`

See also

`GetEncoding`, `cfcontent`, `cfprocessingdirective`, `URLDecode`, `URLEncodedFormat`; "Locales" on page 341 in the *ColdFusion Developer's Guide*

History

ColdFusion MX: Added this function.

Parameters

Parameter	Description
scope_name	• url • form
charset	The character encoding in which text in the scope variables is encoded. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16

Usage

Use this function when the character encoding of the input to a form or the character encoding of a URL is not in UTF-8 encoding. For example, Traditional Chinese characters are often in Big5 encoding. This function resets URL and Form variables, so you should call it before using these variables (typically, in the Application.cfm page or Application.cfc file). Calling this function first also avoids interpreting the characters of the variables incorrectly.

For more information on character encoding, see the following web pages:

- www.w3.org/International/O-charset.html provides general information on character encoding and the web, and has several useful links.
- www.iana.org/assignments/character-sets is a complete list of character sets names used on the Internet, maintained by the Internet Assigned Numbers Authority.
- java.sun.com/j2se/1.4.1/docs/guide/intl/encoding.doc.html lists the character encoding that Java 1.4.1, and therefore the default ColdFusion configuration, can interpret. If you use a JVM that does not conform to the Sun Java 2 Platform, Standard Edition, v 1.4.1, the supported locales may differ. The list uses Java internal names, not the IANA character encoding names that you normally use in the `SetEncoding charset` parameter and other ColdFusion attributes and parameters. Java automatically converts standard IANA names to its internal names as needed.

Example

```
<!--- This example sends and interprets the contents of two fields as
        big5 encoded text. Note that the form fields are received as URL variables
        because the form uses the GET method. --->
<cfcontent type="text/html; charset=big5">
<form action='#cgi.script_name#' method='get'>
<input name='xxx' type='text'>
<input name='yyy' type='text'>
<input type="Submit" value="Submit">
</form>

<cfif IsDefined("URL.xxx")>
<cfscript>
    SetEncoding("url", "big5");
    WriteOutput("URL.XXX is " & URL.xxx & "<br>");
    WriteOutput("URL.YYY is " & URL.yyy & "<br>");
theEncoding = GetEncoding("URL");
     WriteOutput("The URL variables were decoded using '" &
    theEncoding & "' encoding.");

WriteOutput("The encoding is " & theEncoding);
</cfscript>
</cfif>
```

SetLocale

Description

Sets the country/language locale for ColdFusion processing and the page returned to the client. The locale value determines the default format of date, time, number, and currency values, according to language and regional conventions.

Returns

The locale value prior to setting the new locale, as a string.

Category

International functions, System functions

Function syntax

SetLocale(new_locale)

See also

GetHttpTimeString, GetLocale, GetLocaleDisplayName; "Locales" on page 341 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added support for all locales supported by the ColdFusion Java runtime.

ColdFusion MX:

- Changed formatting behavior: this function might return a different value than in earlier releases. This function uses Java standard locale determination and formatting rules on all platforms.
- Deprecated the Spanish (Mexican) locale option. It might not work, and it might cause an error, in later releases.
- Changed the Spanish (Modern) option: it now sets the locale to Spanish (Standard).

Parameters

Parameter	Description
new_locale	The name of a locale; for example, "English (US)"

Usage

You can specify any locale name that is listed in the Server.Coldfusion.SupportedLocales variable. This variable is a comma-delimited list of all locale names supported by the JVM, plus the locale names that were required by ColdFusion prior to ColdFusion MX 7.

The following locale names were used in ColdFusion releases through ColdFusion MX 6.1, and continue to be supported. If you use any of these values in the SetLocale function, the GetLocale function returns the name you set, not the corresponding Java locale name.

Chinese (China)	French (Belgian)	Korean
Chinese (Hong Kong)	French (Canadian)	Norwegian (Bokmal)
Chinese (Taiwan)	French (Standard)	Norwegian (Nynorsk)
Dutch (Belgian)	French (Swiss)	Portuguese (Brazilian)
Dutch (Standard)	German (Austrian)	Portuguese (Standard)

English (Australian)	German (Standard)	Spanish (Modern)
English (Canadian)	German (Swiss)	Spanish (Standard)
English (New Zealand)	Italian (Standard)	Swedish
English (UK)	Italian (Swiss)	
English (US)	Japanese	

ColdFusion determines the locale value as follows:

1 By default, ColdFusion uses the JVM locale, and the default JVM locale is the operating system locale. You can set JVM locale value explicitly in ColdFusion in the ColdFusion Administrator Java and JVM Settings page JVM Arguments field; for example:

```
-Duser.language=de -Duser.region=DE.
```

• A locale set using the `SetLocale` function persists for the current request or until it is reset by another `SetLocale` function in the request.

• If a request has multiple `SetLocale` functions, the current locale setting affects how locale-sensitive ColdFusion tags and functions, such as the functions that start with LS format data. The last `SetLocale` function that ColdFusion processes before sending a response to the requestor (typically the client browser) determines the value of the response `Content-Language` HTTP header. The browser that requested the page displays the response according to the rules for the language specified by the `Content-Language` header.

• ColdFusion ignores any `SetLocale` functions that follow a `cfflush` tag.

Because this function returns the previous locale setting, you can save the original locale value. You can restore the original locale by calling `SetLocale` again with the saved variable. For example, the following line saves the original locale ins a Session variable:

```
<cfset Session.oldlocale = SetLocale(newLocale)>
```

The variable `server.ColdFusion.SupportedLocales` is initialized at startup with a comma-delimited list of the locales that ColdFusion and the operating system support. If you call `SetLocale` with a locale that is not in the list, the call generates an error.

Note: *ColdFusion uses the Spanish (Standard) formats for Spanish (Modern) and Spanish (Standard).*

Example

```
<h3>SetLocale Example</h3>
<p>SetLocale sets the locale to the specified new locale for the current session.
<p>A locale encapsulates the set of attributes that govern the display and
    formatting of date, time, number, and currency values.
<p>The locale for this system is <cfoutput>#GetLocale()#</cfoutput>
<p><cfoutput><I>the old locale was #SetLocale("English (UK)")#</I>
<p>The locale is now #GetLocale()#</cfoutput>
```

SetProfileString

Description

Sets the value of a profile entry in an initialization file.

Returns

An empty string, upon successful execution; otherwise, an error message.

Category

System functions

Function syntax

`SetProfileString(iniPath, section, entry, value)`

See also

GetProfileSections, GetProfileString, SetProfileString

Parameters

Parameter	Description
iniPath	Absolute path of initialization file
section	Section of the initialization file in which the entry is to be set
entry	Name of the entry to set
value	Value to which to set the entry

Example

```
<h3>SetProfileString Example</h3>
This example uses SetProfileString to set the time-out value in an
    initialization file. Enter the full path of your initialization
    file, specify the time-out value, and submit the form.
<!--- This section checks whether the form was submitted. If so, this
    section sets the initialization path and time-out value to the
    path and time-out value specified in the form --->
<cfif Isdefined("Form.Submit")>

    <cfset IniPath = FORM.iniPath>
    <cfset Section = "boot loader">
    <cfset MyTimeout = FORM.MyTimeout>
    <cfset timeout = GetProfileString(IniPath, Section, "timeout")>

    <cfif timeout Is Not MyTimeout>
     <cfif MyTimeout Greater Than 0>
         <hr size = "2" color = "#0000A0">
        <p>Setting the time-out value to <cfoutput>#MyTimeout#</cfoutput>
        </p>
            <cfset code = SetProfileString(IniPath,
                Section, "timeout", MyTimeout)>
        <p>Value returned from SetProfileString:
                <cfoutput>#code#</cfoutput></p>
        <cfelse>
            <hr size = "2" color = "red">
            <p>The time-out value should be greater than zero in order to
                provide time for user response.</p>
            <hr size = "2" color = "red">
```

```
        </cfif>
    <cfelse>
        <p>The time-out value in your initialization file is already
            <cfoutput>#MyTimeout#</cfoutput>.</p>
    </cfif>
    <cfset timeout = GetProfileString(IniPath, Section, "timeout")>
    <cfset default = GetProfileString(IniPath, Section, "default")>

    <h4>Boot Loader</h4>
    <p>The time-out is set to: <cfoutput>#timeout#</cfoutput>.</p>
    <p>Default directory is: <cfoutput>#default#</cfoutput>.</p>

</cfif>

<form action = "setprofilestring.cfm">
<hr size = "2" color = "#0000A0">
<table cellspacing = "2" cellpadding = "2" border = "0">
<tr>
<td>Full Path of Init File</td>
<td><input type = "Text" name = "IniPath"
               value = "C:\myboot.ini"></td>
</tr>
<tr>
<td>Time-out</td>
<td><input type = "Text" name = "MyTimeout" value = "30"></td>
</tr>
<tr>
<td><input type = "Submit" name = "Submit" value = "Submit"></td>
<td></td>
</tr>
</table>
</form>
```

SetVariable

Description

This function is no longer required in well-formed ColdFusion pages.

Sets a variable in the `name` parameter to the value of the `value` parameter.

Returns

The new value of the variable.

Category

Dynamic evaluation functions

Function syntax

```
SetVariable(name, value)
```

See also

DE, Evaluate, IIf

Parameters

Parameter	Description
name	Variable name
value	A string, the name of a string, or a number

Usage

You can use direct assignment statements in place of this function to set values of dynamically named variables. To do so, put the dynamically named variable in quotation marks and number signs (#); for example:

```
<cfset DynamicVar2 = "ABD">
<cfset "#DynamicVar2#" = "Test Value2">
```

Also, the following lines are equivalent:

```
<cfset "myVar#i#" = myVal>
SetVariable("myVar" & i, myVal)
```

For more information, see “Using Expressions and Number Signs” on page 50 in the *ColdFusion Developer’s Guide*.

Example

```
<h3>SetVariable Example</h3>

<cfif IsDefined("FORM.myVariable")>
<!--- strip out url, client., cgi., session., caller. --->
<!--- This example only lets you set form variables --->
<cfset myName = ReplaceList(FORM.myVariable,
 "url,client,cgi,session,caller", "FORM,FORM,FORM,FORM,FORM")>

<cfset temp = SetVariable(myName, FORM.myValue)>
<cfset varName = myName>
<cfset varNameValue = Evaluate(myName)>
<cfoutput>
    <p>Your variable, #varName#
    <p>The value of #varName# is #varNameValue#
</cfoutput>
</cfif>
```

Sgn

Description

Determines the sign of a number.

Returns

- 1, if *number* is positive.
- 0, if *number* is 0.
- -1, if *number* is negative.

Category

Mathematical functions

Function syntax

`Sgn(number)`

See also

`Abs`

Parameters

Parameter	Description
`number`	A number

Example

```
<h3>Sgn Example</h3>
<p>Sgn determines the sign of a number. Returns 1 if number is positive;
    0 if number is 0; -1 if number is negative.
<p>Sgn(14): <cfoutput>#Sgn(14)#</cfoutput>
<p>Sgn(21-21): <cfoutput>#Sgn(21-21)#</cfoutput>
<p>Sgn(-0.007): <cfoutput>#Sgn(-0.007)#</cfoutput>
```

Sin

Description

Calculates the sine of an angle that is entered in radians.

Returns

A number; the sine of the angle.

Category

Mathematical functions

Function syntax

`Sin(number)`

See also

ASin, Cos, ACos, Tan, Atn, Pi

Parameters

Parameter	Description
number	Angle, in radians for which to calculate the sine.

Usage

The range of the result is -1 to 1.

To convert degrees to radians, multiply degrees by π/180. To convert radians to degrees, multiply radians by 180/π.

Note: *Because the function uses floating point arithmetic, it returns a very small number (such as 6.12323399574E-017) for angles that should produce 0. To test for a 0 value, check whether the value is less than 0.0000000000001.*

Example

```
<h3>Sin Example</h3>
<!--- Calculate sine if form has been submitted --->
<cfif IsDefined("FORM.sinNum")>
<!--- Make sure input is a number --->
    <cfif IsNumeric(#FORM.sinNum#)>
<!--- Convert degrees to radians, call the Sin function. --->
<cfset sinValue=#Sin((Form.sinNum * PI()) / 180)#>
<!--- 0.0000000000001 is the function's precision limit.
 If absolute value of returned sine value is
 less, set result to 0 --->
<cfif Abs(sinValue) LT 0.0000000000001>
 <cfset sinValue=0>
</cfif>
<cfoutput>
        Sin(#FORM.sinNum#) = #sinValue#<br><br>
</cfoutput>
    <cfelse>
<!--- If input is not a number, show an error message --->
        <h4>You must enter a numeric angle in degrees.</h4>
    </cfif>
</cfif>
<form action = "#CGI.script_name#" method="post">
Enter an angle in degrees to get its sine:
<br><input type = "Text" name = "sinNum" size = "15">
<br><br>
```

```
<input type = "Submit" name = "">  
<input type = "RESET"
</form>
```

Sleep

Description
Causes the current thread to stop processing for a specified period of time.

Returns
Does not return a value.

Category
System functions

Syntax
Sleep(duration)

See also
cfthread, "Using ColdFusion Threads" on page 301 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Added function

Parameters

Parameter	Description
duration	Time, in milliseconds, to stop processing the thread.

Description
The Sleep function is useful when one thread must wait until another thread performs some action. The thread that must wait uses the Sleep function to stop processing for a time, and, when it awakens, checks to see if the other thread is ready. If it is not, the thread can sleep again. This type of action is useful, for example, when one thread must wait for another thread to complete initialization operations that apply to both threads.

The Sleep function behaves identically to the cfthread tag with an action attribute value of sleep.

Example
The following example has two threads. The second thread (threadB) uses the sleep function to ensure that the first thread (threadA) has completed before it starts processing.

```
<!--- ThreadA loops to simulate an initialization activity that might take time. --->
<cfthread name="threadA" action="run">
    <cfset thread.j=1>
    <cfloop index="i" from="1" to="99999">
        <cfset thread.j=thread.j+1>
    </cfloop>
</cfthread>

<!--- ThreadB loops while threadA is not finished, sleeping for
        1/2 second each time. --->
<cfthread name="threadB" action="run">
    <cfscript>
        thread.sleepTimes=0;
        thread.initialized=false;
        while ((threadA.Status != "COMPLETED") && (threadA.Status
                != "TERMINATED")) {
            sleep(500);
```

```
            thread.sleeptimes++;
        }
        // Only do the post-initilization code if the threadA completed.
        If (threadA.Status == "COMPLETED") {
            thread.initialized=true;
            // Post-initialization code would go here.
        }
    </cfscript>
</cfthread>

<!Join the threads. --->
<cfthread action="join" name="threadA,threadB" timeout="10000"/>

<!--- Display the thread information. --->
<!--- Different actions might be taken based on the thread status information. --->
<cfoutput>
    threadA index value: #threadA.j#<br />
    threadA status: #threadA.Status#<br>
    threadB status: #threadB.Status#<br>
    threadB sleepTimes: #threadB.sleepTimes#<br>
    threadB initialized: #threadB.initialized#<br>
</cfoutput>
```

SpanExcluding

Description

Gets characters from a string, from the beginning to a character that is in a specified set of characters. The search is case-sensitive.

Returns

A string; characters from *string*, from the beginning to a character that is in *set*.

Category

String functions

Function syntax

`SpanExcluding(string, set)`

See also

`GetToken`, `SpanIncluding`; "Caching parts of ColdFusion pages" on page 241 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`string`	A string or a variable that contains one
`set`	A string or a variable that contains one. Must contain one or more characters

Example

```
<h3>SpanExcluding Example</h3>

<cfif IsDefined("FORM.myString")>
<p>Your string was <cfoutput>#FORM.myString#</cfoutput>
<p>Your set of characters was <cfoutput>#FORM.mySet#</cfoutput>
<p>Your string up until one of the characters in the set is:
<cfoutput>#SpanExcluding(FORM.myString, FORM.mySet)#</cfoutput>
</cfif>

<p>Returns all characters from string from beginning to a character
    from the set of characters. The search is case-sensitive.

<form method = post action = "spanexcluding.cfm">
<p>Enter a string:
<br><input type = "Text" name = "myString" value = "Hey, you!">
<p>And a set of characters:
<br><input type = "Text" name = "mySet" value = "Ey">
<br><input type = "Submit" name = "">
</form>
```

SpanIncluding

Description

Gets characters from a string, from the beginning to a character that is not in a specified set of characters. The search is case-sensitive.

Returns

A string; characters from *string*, from the beginning to a character that is not in *set*.

Category

String functions

Function syntax

`SpanIncluding(string, set)`

See also

`GetToken`, `SpanExcluding`; "Caching parts of ColdFusion pages" on page 241in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`string`	A string or a variable that contains the search string.
`set`	A string or a variable that contains a set of characters. Must contain one or more characters.

Example

```
<h3>SpanIncluding Example</h3>
<cfif IsDefined("FORM.myString")>
<p>Your string was <cfoutput>#FORM.myString#</cfoutput>
<p>Your set of characters was <cfoutput>#FORM.mySet#</cfoutput>
<p>Your string, until the characters in the set have been found, is:
<cfoutput>#SpanIncluding(FORM.myString, FORM.mySet)#</cfoutput>
</cfif>

<p>Returns characters of a string, from beginning to a character
    that is not in set. The search is case-sensitive.

<form action = "spanincluding.cfm" method="post">
<p>Enter a string:
<br><input type = "Text" name = "myString" value = "Hey, you!">
<p>And a set of characters:
<br><input type = "Text" name = "mySet" value = "ey,H">
<br><input type = "Submit" name = "">
</form>
```

Sqr

Description

Calculates the square root of a number.

Returns

Number; square root of *number*.

Category

Mathematical functions

Function syntax

`Sqr(number)`

See also

`Abs`

Parameters

Parameter	Description
`number`	A positive integer or a variable that contains one. Number whose square root to get.

Usage

The value in `number` must be greater than or equal to 0.

Example

```
<h3>Sqr Example</h3>

<p>Returns the square root of a positive number.

<p>Sqr(2): <cfoutput>#Sqr(2)#</cfoutput>
<p>Sqr(Abs(-144)): <cfoutput>#Sqr(Abs(-144))#</cfoutput>
<p>Sqr(25^2): <cfoutput>#Sqr(25^2)#</cfoutput>
```

StripCR

Description
Deletes return characters from a string.

Returns
A copy of *string*, after removing return characters.

Category
Display and formatting functions, String functions

Function syntax
`StripCR(string)`

See also
`ParagraphFormat`

Parameters

Parameter	Description
`string`	A string or a variable that contains one

Usage
Useful for preformatted (between `<pre>` and `</pre>` tags) HTML display of data entered in `textarea` fields.

Example
```
<h3>StripCR Example</h3>

<p>Function StripCR is useful for preformatted HTML display of data
    (PRE) entered in textarea fields.
<cfif isdefined("Form.myTextArea")>

<pre>
<cfoutput>#StripCR(Form.myTextArea)#</cfoutput>
</pre>
</cfif>
<!--- use #Chr(10)##Chr(13)# to simulate line feed/carriage return combination --->
<form action = "stripcr.cfm">
<textarea name = "MyTextArea" cols = "35" rows = 8>
This is sample text and you see how it scrolls
    <cfoutput>#Chr(10)##Chr(13)#</cfoutput>
From one line
    <cfoutput>#Chr(10)##Chr(13)##Chr(10)##Chr(13)#</cfoutput>
    to the next
</textarea>
<input type = "Submit" name = "Show me the HTML version">
</form>
```

StructAppend

Description
Appends one structure to another.

Returns
True, upon successful completion; False, otherwise.

Category
Structure functions

Function syntax
`StructAppend(struct1, struct2, overwriteFlag)`

See also
Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
`struct1`	Structure to which struct2 is appended.
`struct2`	Structure that contains the data to append to `struct1`
`overwriteFlag`	• True or Yes: values in `struct2` overwrite corresponding values in `struct1`. Default. • False or No: values in `struct2` do not overwrite corresponding values in `struct1`.

Usage
This function appends the fields and values of `struct2` to `struct1`; `struct2` is not modified. If `struct1` already contains a field of `struct2`, `overwriteFlag` determines whether the value in `struct2` overwrites it.

A structure's keys are unordered.

Example
```
<html>
<body>
<!---- Create a Name structure --->
<cfset nameCLK=StructNew()>
<cfset nameCLK.first="Chris">
<cfset nameCLK.middle="Lloyd">
<cfset nameCLK.last="Gilson">
<!--- Create an address struct --->
<cfset addrCLK=StructNew()>
<cfset addrCLK.street="17 Gigantic Rd">
<cfset addrCLK.city="Watertown">
<cfset addrCLK.state="MA">
<cfset addrCLK.zip="02472">
<!---- Create a Person structure --->
<cfset personCLK=StructNew()>
<cfset personCLK.name=#nameCLK#>
<cfset personCLK.addr=#addrCLK#>
<!--- Display the contents of the person struct before the Append --->
```

```
<p>
The person struct <b>before</b> the Append call:<br>
<cfloop collection=#personCLK# item="myItem">
<cfoutput>
#myItem#<br>
</cfoutput>
</cfloop>
<!--- Merge the address struct into the top-level person struct --->
<cfset bSuccess = StructAppend( personCLK, addrCLK )>

<!--- Display the contents of the person struct, after the Append --->
<p>
The person struct <b>after</b> the Append call:<br>
<cfloop collection=#personCLK# item="myItem">
    <cfoutput>
        #myItem#<br>
    </cfoutput>
</cfloop>
```

StructClear

Description
Removes all data from a structure.

Returns
True, on successful execution; False, otherwise.

Category
Structure functions

Function syntax
`StructClear(structure)`

See also
Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
structure	Structure to clear

Usage
Do not call this function on a session variable. For more information, see TechNote 14143, "*ColdFusion 4.5 and the StructClear(Session) function*," at www.coldfusion.com/Support/KnowledgeBase/SearchForm.cfm. (The article applies to ColdFusion 4.5, 5.x, and ColdFusion MX.)

Example
```
<!--- Shows StructClear function. Calls cf_addemployee custom tag which
    uses the addemployee.cfm file. --->
<body>
<h1>Add New Employees</h1>
<!--- Establish params for first time through --->
<cfparam name = "Form.firstname" default = "">
<cfparam name = "Form.lastname" default = "">
<cfparam name = "Form.email" default = "">
<cfparam name = "Form.phone" default = "">
<cfparam name = "Form.department" default = "">
<cfif form.firstname eq "">
 <p>Please fill out the form.
<cfelse>
 <cfoutput>
<cfscript>
 employee = StructNew();
 StructInsert(employee, "firstname", Form.firstname);
 StructInsert(employee, "lastname", Form.lastname);
 StructInsert(employee, "email", Form.email);
 StructInsert(employee, "phone", Form.phone);
 StructInsert(employee, "department", Form.department);
</cfscript>
 </cfoutput>
```

```
<!--- Call the custom tag that adds employees --->
 <cf_addemployee empinfo = "#employee#">
 <cfscript>StructClear(employee);</cfscript>
 </cfif>
```

StructCopy

Description

Copies a structure. Copies top-level keys, values, and arrays in the structure by value; copies nested structures by reference. **Returns**
A copy of a structure, with the same keys and values; if *structure* does not exist, throws an exception.

Category

Structure functions

Function syntax

`StructCopy(structure)`

See also

Structure functions; "Structure functions" on page 90 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`structure`	Structure to copy

Usage

The following code shows how this function copies a structure that contains a string field, a number field, and a two-dimensional array at the top level:

```
<cfoutput>
    <cfset assignedCopy = StructNew()>
<cfset assignedCopy.string = #struct.string#>
    <cfset assignedCopy.number = #struct.number#>
    <cfset assignedCopy.array = ArrayNew(2)>
    <cfset assignedCopy.array[1][1] = #struct.array[1][1]#>
    <cfset assignedCopy.array[1][2] = #sruct.array[1][2]#>
</cfoutput>
```

The following code shows how `StructCopy` copies a nested structure:

```
<cfoutput>
<cfset assignedCopy.nestedStruct = struct.nestedStruct>
</cfoutput>
```

To copy a structure entirely by value, use "Duplicate" on page 778.

The following table shows how variables are assigned:

Variable type	Assigned by
structure.any_simple_value Boolean Binary Base64	Value
structure.array	Value

Variable type	Assigned by
structure.nested_structure	Reference
structure.object	Reference
structure.query	Reference

Example

```
<!--- This code shows assignment by-value and by-reference. --->
// This script creates a structure that StructCopy copies by value. <br>
<cfscript>
    // Create elements.
    s = StructNew();
    s.array = ArrayNew(2);

    // Assign simple values to original top-level structure fields.
    s.number = 99;
    s.string = "hello tommy";

    // Assign values to original top-level array.
    s.array[1][1] = "one one";
    s.array[1][2] = "one two";
</cfscript>

<!--- Output original structure --->
<hr>
<b>Original Values</b><br>
<cfoutput>
    // Simple values <br>
    s.number = #s.number#<br>
    s.string = #s.string#<br>
    // Array value <br>
    s.array[1][1] = #s.array[1][1]#<br>
    s.array[1][2] = #s.array[1][2]#<br>
</cfoutput>

// Copy this structure to a new structure. <br>
<cfset copied = StructCopy(s)>

<cfscript>
// Change the values of the original structure. <br>
    s.number = 100;
    s.string = "hello tommy (modified)";
    s.array[1][1] = "one one (modified)";
    s.array[1][2] = "one two (modified)";
</cfscript>
<hr>
<b>Modified Original Values</b><br>
<cfoutput>
    // Simple values <br>
    s.number = #s.number#<br>
    s.string = #s.string#<br>
    // Array value <br>
    s.array[1][1] = #s.array[1][1]#<br>
    s.array[1][2] = #s.array[1][2]#<br>
</cfoutput>
<hr>
<b>Copied structure values should be the same as the original.</b><br>
<cfoutput>
    // Simple values <br>
```

```
    copied.number = #copied.number#<br>
    copied.string = #copied.string#<br>
    // Array value <br>
    copied.array[1][1] = #copied.array[1][1]#<br>
    copied.array[1][2] = #copied.array[1][2]#<br>
</cfoutput>

// This script creates a structure that StructCopy copies by reference.
<cfscript>
    // Create elements.
    s = StructNew();
    s.nested = StructNew();
    s.nested.array = ArrayNew(2);
    // Assign simple values to nested structure fields.
    s.nested.number = 99;
    s.nested.string = "hello tommy";
    // Assign values to nested array.
    s.nested.array[1][1] = "one one";
    s.nested.array[1][2] = "one two";
</cfscript>

<!--- Output original structure --->
<hr>
<b>Original Values</b><br>
<cfoutput>
    // Simple values <br>
    s.nested.number = #s.nested.number#<br>
    s.nested.string = #s.nested.string#<br>

    // Array values <br>
    s.nested.array[1][1] = #s.nested.array[1][1]#<br>
    s.nested.array[1][2] = #s.nested.array[1][2]#<br>
</cfoutput>

// Use StructCopy to copy this structure to a new structure. <br>
<cfset copied = StructCopy(s)>
// Use Duplicate to clone this structure to a new structure. <br>
<cfset duplicated = Duplicate(s)>

<cfscript>
// Change the values of the original structure.
    s.nested.number = 100;
    s.nested.string = "hello tommy (modified)";
    s.nested.array[1][1] = "one one (modified)";
    s.nested.array[1][2] = "one two (modified)";
</cfscript>
<hr>
<b>Modified Original Values</b><br>
<cfoutput>
    // Simple values <br>
    s.nested.number = #s.nested.number#<br>
    s.nested.string = #s.nested.string#<br>

    // Array value <br>
    s.nested.array[1][1] = #s.nested.array[1][1]#<br>
    s.nested.array[1][2] = #s.nested.array[1][2]#<br>
</cfoutput>

<hr>
<b>Copied structure values should reflect changes to original.</b><br>
<cfoutput>
```

```
    // Simple values <br>
    copied.nested.number = #copied.nested.number#<br>
    copied.nested.string = #copied.nested.string#<br>
    // Array values <br>
    copied.nested.array[1][1] = #copied.nested.array[1][1]#<br>
    copied.nested.array[1][2] = #copied.nested.array[1][2]#<br>
</cfoutput>

<hr>
<b>Duplicated structure values should remain unchanged.</b><br>
<cfoutput>
    // Simple values <br>
    duplicated.nested.number = #duplicated.nested.number#<br>
    duplicated.nested.string = #duplicated.nested.string#<br>
    // Array value <br>
    duplicated.nested.array[1][1] = #duplicated.nested.array[1][1]#<br>
    duplicated.nested.array[1][2] = #duplicated.nested.array[1][2]#<br>
</cfoutput>
```

StructCount

Description
Counts the keys in a structure.

Returns
A number; if *structure* does not exist, throws an exception.

Category
Structure functions

Function syntax
`StructCount(structure)`

See also
Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
structure	Structure to access

Example
```
<!--- This view-only example shows use of StructCount. --->
<p>This file is similar to addemployee.cfm, which is called by
    StructNew, StructClear, and StructDelete. To test, copy
    StructCount function to appropriate place in addemployee.cfm.
<!---
<cfswitch expression = "#ThisTag.ExecutionMode#">
<cfcase value = "start">
 <cfif StructIsEmpty(attributes.EMPINFO)>
<cfoutput>Error. No employee data was passed.</cfoutput>
 <cfexit method = "ExitTag">
 <cfelse>
<cfquery name = "AddEmployee" datasource = "cfdocexamples">
 INSERT INTO Employees
 (FirstName, LastName, Email, Phone, Department)
VALUES
 <cfoutput>
(
                '#StructFind(attributes.EMPINFO, "firstname")#' ,
                '#StructFind(attributes.EMPINFO, "lastname")#' ,
                '#StructFind(attributes.EMPINFO, "email")#' ,
                '#StructFind(attributes.EMPINFO, "phone")#' ,
                '#StructFind(attributes.EMPINFO, "department")#'
 )
 </cfoutput>
 </cfquery>
 </cfif>
 <cfoutput><hr>Employee Add Complete
 <p>#StructCount(attributes.EMPINFO)# columns added.</cfoutput>
</cfcase>
</cfswitch> --->
```

StructDelete

Description
Removes an element from a structure.

Returns
Boolean value. The value depends on the `indicatenotexisting` parameter value.

Category
Structure functions

Function syntax
`StructDelete(structure, key [, indicatenotexisting ])`

See also
Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
`structure`	Structure or a variable that contains one. Contains element to remove.
`key`	Element to remove.
`indicatenotexisting`	• True: returns Yes if *key* exists; No if it does not. • False: returns Yes regardless of whether *key* exists. Default.

Example
```
<h3>StructDelete Function</h3>
<!--- Delete the surrounding comments to make this page work
<p>This example uses the StructInsert and StructDelete functions.
<!--- Establish params for first time through --->
<cfparam name = "firstname" default = "Mary">
<cfparam name = "lastname" default = "Sante">
<cfparam name = "email" default = "msante@allaire.com">
<cfparam name = "phone" default = "777-777-7777">
<cfparam name = "department" default = "Documentation">

 <cfif IsDefined("FORM.Delete")>
 <cfoutput>
 Field to be deleted: #form.field#
</cfoutput>
<p>
<CFScript>
 employee = StructNew();
 StructInsert(employee, "firstname", firstname);
 StructInsert(employee, "lastname", lastname);
 StructInsert(employee, "email", email);
 StructInsert(employee, "phone", phone);
 StructInsert(employee, "department", department);
</CFScript>
    Before deletion, employee structure looks like this:
<cfdump var="#employee#">
```

```
<br>
    <cfset rc = StructDelete(employee, "#form.field#", "True")>
    <cfoutput>
    Did I delete the field "#form.field#"? The code indicates: #rc#<br>
    The structure now looks like this:<br>
<cfdump var="#employee#">
<br>
    </cfoutput>
</cfif>
<br><br>
<form method="post" action = "#CGI.Script_Name#">
    <p>Select the field to be deleted: 
    <select name = "field">
    <option value = "firstname">first name
    <option value = "lastname">last name
    <option value = "email">email
    <option value = "phone">phone
    <option value = "department">department
    </select>
    <input type = "submit" name = "Delete" value = "Delete">
</form>
Delete this comment to make this page work --->
```

StructFind

Description
Determines the value associated with a key in a structure.

Returns
The value associated with a key in a structure; if *structure* does not exist, throws an exception.

Category
Structure functions

Function syntax
`StructFind(structure, key)`

See also
Structure functions; “Structure functions” on page 90 in the *ColdFusion Developer’s Guide*

Parameters

Parameter	Description
`structure`	Structure that contains the value to return
`key`	Key whose value to return

Usage
A structure’s keys are unordered.

Example
```
<!--- This view-only example shows the use of StructFind. --->
<p>This file is identical to addemployee.cfm, which is called by StructNew,
    StructClear, and StructDelete. It adds employees. Employee information
    is passed through the employee structure (EMPINFO attribute). In UNIX,
    you must also add the Emp_ID.
<!--- <cfswitch expression = "#ThisTag.ExecutionMode#">
<cfcase value = "start">
 <cfif StructIsEmpty(attributes.EMPINFO)>
<cfoutput>Error. No employee data was passed.</cfoutput>
 <cfexit method = "ExitTag">
 <cfelse>
 <cfquery name = "AddEmployee" datasource = "cfdocexamples">
 INSERT INTO Employees (FirstName, LastName, Email, Phone, Department)
VALUES
 <cfoutput>
(
                '#StructFind(attributes.EMPINFO, "firstname")#' ,
                '#StructFind(attributes.EMPINFO, "lastname")#' ,
                '#StructFind(attributes.EMPINFO, "email")#' ,
                '#StructFind(attributes.EMPINFO, "phone")#' ,
                '#StructFind(attributes.EMPINFO, "department")#' )
 </cfoutput>
</cfquery>
 </cfif>
 <cfoutput><hr>Employee Add Complete</cfoutput>
</cfcase>
</cfswitch> --->
```

StructFindKey

Description

Searches recursively through a substructure of nested arrays, structures, and other elements, for structures whose values match the search key in the *value* parameter.

Returns

An array that contains structures with values that match *value*.

Category

Structure functions

Function syntax

```
StructFindKey(top, value, scope)
```

See also

Structure functions; "Structure functions" on page 90 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
top	ColdFusion object (structure or array) from which to start search. This parameter requires an object, not a name of an object.
value	String or a variable that contains one for which to search.
scope	• one: returns one matching key. Default. • all: returns all matching keys.

Usage

Returns an array that includes one structure for each of the specified values it finds. The fields of each of these structures are:

- `Value`: value held in the found key
- `Path`: string that can be used to reach the found key
- `Owner`: parent object that contains the found key

A structure's keys are unordered.

Example

```
<cfset aResults = StructFindKey( #request#, "bass" )>
```

StructFindValue

Description

Searches recursively through a substructure of nested arrays, structures, and other elements for structures with values that match the search key in the `value` parameter.

Returns

An array that contains structures with values that match the search key *value*. If none are found, returns an array of size 0.

Category

Structure functions

Function syntax

```
StructFindValue( top, value [, scope])
```

See also

Structure functions; "Structure functions" on page 90 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`top`	ColdFusion structure from which to start search. This parameter requires an object, not a name of an object.
`value`	String or a variable that contains one for which to search. The type must be a simple object. Arrays and structures are not supported.
`scope`	• one: function returns one matching key (default). • all: function returns all matching keys.

Usage

The fields of each structure in the returned array are:

- `Key`: name of the key in which the value was found
- `Path`: string which could be used to reach the found key
- `Owner`: parent object that contains the found key

A structure's keys are unordered.

Example

```
<cfset aResults = StructFindValue( #request#, "235" )>
```

StructGet

Description
Gets a structure(s) from a specified path.

Returns
An alias to the variable in the *pathDesired* parameter. If necessary, `StructGet` creates structures or arrays to make *pathDesired* a valid variable "path."

Category
Structure functions

Function syntax
`StructGet(pathDesired)`

See also
Structure functions; “Modifying a ColdFusion XML object” on page 878 in the *ColdFusion Developer’s Guide*

History
ColdFusion MX:

- Changed behavior: this function can be used on XML objects.
- Changed behavior: if there is no structure or array present in *pathDesired*, this function creates structures or arrays to make *pathDesired* a valid variable "path."

Parameters

Parameter	Description
`pathDesired`	Pathname of variable that contains structure or array from which ColdFusion retrieves structure.

Usage
You can inadvertently create invalid structures using this function. For example, if array notation is used to expand an existing array, the specified new element is created, regardless of the type currently held in the array.

Example
```
<!--- GetStruct() test --->
<cfset test = StructGet( "dog.myscope.test" )>
<cfset test.foo = 1>
<cfif NOT IsDefined("dog")>
    Dog is not defined<br>
</cfif>
<cfif NOT IsDefined("dog.myscope")>
    Dog.Myscope is not defined<br>
</cfif>
<cfif NOT Isdefined("dog.myscope.test")>
    Dog.Myscope.Test is not defined<br>
</cfif>
<cfif NOT Isdefined("dog.myscope.test.foo")>
    Dog.Myscope.Test.Foo is not defined<br>
</cfif>
<cfoutput>
    #dog.myscope.test.foo#<br>
</cfoutput>
<cfset test = StructGet( "request.myscope[1].test" )>
```

```
<cfset test.foo = 2>
<cfoutput>
    #request.myscope[1].test.foo#<br>
</cfoutput>
<cfset test = StructGet( "request.myscope[1].test[2]" )>
<cfset test.foo = 3>
<cfoutput>
    #request.myscope[1].test[2].foo#<br>
</cfoutput>
```

StructInsert

Description
Inserts a key-value pair into a structure.

Returns
True, upon successful completion. If `structure` does not exist, or if `key` exists and `allowoverwrite = "False"`, ColdFusion throws an exception.

Category
Structure functions

Function syntax
`StructInsert(structure, key, value [, allowoverwrite ])`

See also
Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
`structure`	Structure to contain the new key-value pair.
`key`	Key that contains the inserted value.
`value`	Value to add.
`allowoverwrite`	Optional. Whether to allow overwriting a key. The default value is False.

Usage
A structure's keys are unordered.

Example
```
<h1>Add New Employees</h1>
<!--- Establish params for first time through --->
<cfparam name = "FORM.firstname" default = "">
<cfparam name = "FORM.lastname" default = "">
<cfparam name = "FORM.email" default = "">
<cfparam name = "FORM.phone" default = "">
<cfparam name = "FORM.department" default = "">

<cfif FORM.firstname EQ "">
 <p>Please fill out the form.
<cfelse>
 <cfoutput>
<CFScript>
 employee = StructNew();
 StructInsert(employee, "firstname", FORM.firstname);
 StructInsert(employee, "lastname", FORM.lastname);
 StructInsert(employee, "email", FORM.email);
 StructInsert(employee, "phone", FORM.phone);
 StructInsert(employee, "department", FORM.department);
 </CFScript>
```

```
 <p>First name is #StructFind(employee, "firstname")#</p>
 <p>Last name is #StructFind(employee, "lastname")#</p>
 <p>EMail is #StructFind(employee, "email")#</p>
 <p>Phone is #StructFind(employee, "phone")#</p>
 <p>Department is #StructFind(employee, "department")#</p>
 </cfoutput>

 <!--- Call the custom tag that adds employees --->
 <CF_ADDEMPLOYEE EMPINFO = "#employee#">
</cfif>

<Hr>
<form action = "structinsert.cfm">
    <p>First Name: 
    <input name = "firstname" type = "text" hspace = "30" maxlength = "30">
    <p>Last Name: 
    <input name = "lastname" type = "text" hspace = "30" maxlength = "30">
    <p>EMail: 
    <input name = "email" type = "text" hspace = "30" maxlength = "30">
    <p>Phone: 
    <input name = "phone" type = "text" hspace = "20" maxlength = "20">
    <p>Department: 
    <input name = "department" type = "text" hspace = "30" maxlength = "30">
    <p>
    <input type = "submit" value = "OK">
</form>
```

StructIsEmpty

Description
Determines whether a structure contains data.

Returns
True, if *structure* is empty; if *structure* does not exist, ColdFusion throws an exception.

Category
Decision functions, Structure functions

Function syntax
`StructIsEmpty(structure)`

See also
Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
structure	Structure to test

Example
```
<!--- This example illustrates use of StructIsEmpty. --->
<p>This file is identical to addemployee.cfm, which is called by StructNew,
    StructClear, and StructDelete. It adds employees. Employee information
    is passed through employee structure (EMPINFO attribute). In UNIX, you
    must also add the Emp_ID.
<cfswitch expression = "#ThisTag.ExecutionMode#">
<cfcase value = "start">
 <cfif StructIsEmpty(attributes.EMPINFO)>
<cfoutput>Error. No employee data was passed.</cfoutput>
 <cfexit method = "ExitTag">
 <cfelse>
<!--- Add the employee; In UNIX, you must also add the Emp_ID --->
 <cfquery name = "AddEmployee" datasource = "cfdocexamples">
 INSERT INTO Employees
 (FirstName, LastName, Email, Phone, Department)
VALUES
 <cfoutput>
(
                '#StructFind(attributes.EMPINFO, "firstname")#' ,
                '#StructFind(attributes.EMPINFO, "lastname")#' ,
                '#StructFind(attributes.EMPINFO, "email")#' ,
                '#StructFind(attributes.EMPINFO, "phone")#' ,
                '#StructFind(attributes.EMPINFO, "department")#'
 )
 </cfoutput>
</cfquery>
 </cfif>
 <cfoutput><hr>Employee Add Complete</cfoutput>
</cfcase>
</cfswitch>
```

StructKeyArray

Description
Finds the keys in a ColdFusion structure.

Returns
An array of keys; if *structure* does not exist, ColdFusion throws an exception.

Category
Structure functions

Function syntax
`StructKeyArray(structure)`

See also
Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`structure`	Structure from which to extract a list of keys

Usage
A structure's keys are unordered.

Example
```
<!--- Shows StructKeyArray function to copy keys from a structure to an array.
    Uses StructNew to create structure and fills its fields with the
    information the user enters in the form fields. --->
<h3>StructKeyArray Example</h3>
<h3>Extracting the Keys from the Employee Structure</h3>
<!-- Create structure. Check whether Submit was pressed. If so, define fields
    in employee structure with user entries on form. ----->
<cfset employee = StructNew()>
<cfif Isdefined("Form.Submit")>
    <cfif Form.Submit is "OK">
        <cfset employee.firstname = FORM.firstname>
        <cfset employee.lastname = FORM.lastname>
        <cfset employee.email = FORM.email>
        <cfset employee.phone = FORM.phone>
        <cfset employee.company = FORM.company>
    <cfelseIf Form.Submit is "Clear">
        <cfset rc = StructClear(employee)>
    </cfif>
</cfif>
<p> This example uses the StructNew function to create a structure called
    "employee" that supplies employee info. Its fields are filled by
    the form. After you enter employee information in structure, the
    example uses StructKeyArray function to copy all of the keys from
    the structure into an array. </p>
<hr size = "2" color = "#0000A0">
<form action = "structkeyarray.cfm">
<table cellspacing = "2" cellpadding = "2" border = "0">
    <tr>
    <td>First Name:</td>
    <td><input name = "firstname" type = "text"
```

```
        value = "" hspace = "30" maxlength = "30"></td>
    </tr>
    <tr>
    <td>Last Name:</td>
    <td><input name = "lastname" type = "text"
        value = "" hspace = "30" maxlength = "30"></td>
    </tr>
    <tr>
    <td>EMail</td>
    <td><input name = "email" type = "text"
        value = "" hspace = "30" maxlength = "30"></td>
    </tr>
    <tr>
    <td>Phone:</td>
    <td><input name = "phone" type = "text"
        value = "" hspace = "20" maxlength = "20"></td>
    </tr>
    <tr>
    <td>Company:</td>
    <td><input name = "company" type = "text"
        value = "" hspace = "30" maxlength = "30"></td>
    </tr>
    <tr>
    <td><input type = "submit" name = "submit"
        value = "OK"></td>
    <td><b>After you submit the FORM, scroll down to see the array.</b>
    </td>
    </tr>
</table>
</form>
<cfif NOT StructISEmpty(employee)>
    <hr size = "2" color = "#0000A0">
    <cfset keysToStruct = StructKeyArray(employee)>
    <cfloop index = "i" from = "1" to = "#ArrayLen(keysToStruct)#">
        <p><cfoutput>Key#i# is #keysToStruct[i]#</cfoutput></p>
        <p><cfoutput>Value#i# is #employee[keysToStruct[i]]#</cfoutput>
        </p>
    </cfloop>
</cfif>
```

StructKeyExists

Description

Determines whether a specific key is present in a structure.

Returns

True, if *key* is in *structure*; if *structure* does not exist, ColdFusion throws an exception.

Category

Decision functions, Structure functions

Function syntax

```
StructKeyExists(structure, "key")
```

See also

Structure functions; "Structure functions" on page 90 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
structure	Name of structure to test
key	Key to test

Usage

This function can sometimes be used in place of the `IsDefined` function, when working with the URL and Form scopes, which are structures. The following pieces of code are equivalent:

```
cfif IsDefined("Form.JediMaster")>
<cfif StructKeyExists(Form,"JediMaster")>
```

A structure's keys are unordered.

Example

```
<!--- This example shows the use of StructKeyExists. --->
<p>This file is similar to addemployee.cfm, which is called by StructNew,
    StructClear, and StructDelete. To test, copy the &LT;CFELSEif&GT;
    statement to the appropriate place in addemployee.cfm. It is a custom tag
    to add employees. Employee information is passed through the employee
    structure (the EMPINFO attribute). In UNIX, you must also add the Emp_ID.

<cfswitch expression = "#ThisTag.ExecutionMode#">
<cfcase value = "start">
 <cfif StructIsEmpty(attributes.EMPINFO)>
<cfoutput>Error. No employee data was passed.</cfoutput>
 <cfexit method = "ExitTag">
 <cfelseIf NOT StructKeyExists(attributes.EMPINFO, "department")>
<cfscript>StructUpdate(attributes.EMPINFO, "department",
"Unassigned");
        </cfscript>
<cfexit method = "ExitTag">
         <cfelse>
```

StructKeyList

Description
Extracts keys from a ColdFusion structure.

Returns
A list of keys; if *structure* does not exist, ColdFusion throws an exception.

Category
Structure functions

Function syntax
`StructKeyList(structure [, delimiter])`

See also
Structure functions; "Modifying a ColdFusion XML object" on page 878 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`structure`	Structure from which to extract a list of keys.
`delimiter`	Optional. Character that separates keys in list. The default value is comma.

Usage
A structure's keys are unordered.

Example
```
<!--- This example shows how to use StructKeyList to list the keys
    in a structure. It uses StructNew function to create structure
    and fills it with information user enters in form fields. --->
<!--- This section creates structure and checks whether Submit has been pressed.
    If so, code defines fields in the employee structure with what the
    user entered in the form. --->
<cfset employee = StructNew()>
<cfif Isdefined("Form.Submit")>
    <cfif Form.Submit is "OK">
        <cfset employee.firstname = FORM.firstname>
        <cfset employee.lastname = FORM.lastname>
        <cfset employee.email = FORM.email>
        <cfset employee.phone = FORM.phone>
        <cfset employee.company = FORM.company>
    <cfelseIf Form.Submit is "Clear">
        <cfset rc = StructClear(employee)>
    </cfif>
</cfif>
<html>
<head>
    <title>StructKeyList Function</title>
</head>
<body>
<h3>StructKeyList Function</h3>
<h3>Listing the Keys in the Employees Structure</h3>
<p>This example uses StructNew function to create structure "employee" that
    supplies employee information. The fields are filled with the
    contents of the following form.</p>
```

```
<p>After you enter employee information into structure, example uses
    <b>StructKeyList</b> function to list keys in structure.</p>
<p>This code does not show how to insert information into a database.
    See cfquery for more information about database insertion.
<hr size = "2" color = "#0000A0">
<form action = "structkeylist.cfm">
<table cellspacing = "2" cellpadding = "2" border = "0">
    <tr>
    <td>First Name:</td>
    <td><input name = "firstname" type = "text"
        value = "" hspace = "30" maxlength = "30"></td>
    </tr>
    <tr>
    <td>Last Name:</td>
    <td><input name = "lastname" type = "text"
        value = "" hspace = "30" maxlength = "30"></td>
    </tr>
    <tr>
    <td>EMail</td>
    <td><input name = "email" type = "text"
        value = "" hspace = "30" maxlength = "30"></td>
    </tr>
    <tr>
    <td>Phone:</td>
    <td><input name = "phone" type = "text"
        value = "" hspace = "20" maxlength = "20"></td>
    </tr>
    <tr>
    <td>Company:</td>
    <td><input name = "company" type = "text"
        value = "" hspace = "30" maxlength = "30"></td>
    </tr>
    <tr>
    <td><input type = "submit" name = "submit" value = "OK"></td>
    <td><b>After you submit form, scroll down to see the list.</b></td>
    </tr>
</table>
</form>
<cfif NOT StructISEmpty(employee)>
    <hr size = "2" color = "#0000A0">
    <cfset keysToStruct = StructKeyList(employee,"<li>")>
    <p>Here are the keys to the structure:</p>
    <ul>
    <li><cfoutput>#keysToStruct#</cfoutput>
    </ul>
    <p>If fields are correct, we can process new employee information.
    If they are not correct, consider rewriting application.</p>
</cfif>
```

StructNew

Description
Creates a structure.

Returns
A structure.

Category
Structure functions

Function syntax
```
StructNew()
```

See also
Structure functions; "Structure functions" on page 90 in the *ColdFusion Developer's Guide*

Parameters
None

Example
```
<!--- Shows StructNew. Calls CF_ADDEMPLOYEE, which uses the |
    addemployee.cfm file to add employee record to database. --->
<h1>Add New Employees</h1>
<cfparam name = "FORM.firstname" default = "">
<cfparam name = "FORM.lastname" default = "">
<cfparam name = "FORM.email" default = "">
<cfparam name = "FORM.phone" default = "">
<cfparam name = "FORM.department" default = "">
<cfif FORM.firstname EQ "">
 <p>Please fill out the form.
<cfelse>
    <cfoutput>
<cfscript>
 employee = StructNew();
 StructInsert(employee, "firstname", FORM.firstname);
 StructInsert(employee, "lastname", FORM.lastname);
 StructInsert(employee, "email", FORM.email);
 StructInsert(employee, "phone", FORM.phone);
 StructInsert(employee, "department", FORM.department);
 </cfscript>
    <p>First name is #StructFind(employee, "firstname")#
    <p>Last name is #StructFind(employee, "lastname")#
    <p>EMail is #StructFind(employee, "email")#
    <p>Phone is #StructFind(employee, "phone")#
    <p>Department is #StructFind(employee, "department")#
    </cfoutput>
<!--- Call the custom tag that adds employees --->
 <CF_ADDEMPLOYEE EMPINFO = "#employee#">
</cfif>
```

StructSort

Description

Returns a sorted array of the top level keys in a structure. Sorts using alphabetic or numeric sorting, and can sort based on the values of any structure element.

Returns

An array of top-level key names (strings), sorted by the value of the specified subelement.

Category

Structure functions

Function syntax

`StructSort(base, sortType, sortOrder, pathToSubElement)`

See also

Structure functions; "Structure functions" on page 90 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`base`	A ColdFusion structure with one field (an associative array).
`sortType`	• numeric • text: case-sensitive (all lowercase letters precede the first uppercase letter). Default. • textnocase
`sortOrder`	• asc: ascending (a to z) sort order. Default. • desc: descending (z to a) sort order
`pathToSubElement`	String or a variable that contains one. Path to apply to each top-level key, to reach element value by which to sort. The default value is nothing (top-level entries sorted by their own values).

Usage

The `pathToSubElement` string does not support array notation, and only supports substructures of structures.

This function does not sort or change the structure.

Example

```
<cfscript>
    salaries = StructNew() ;
    employees = StructNew() ;
    departments = StructNew() ;
    for ( i=1; i lt 6; i=i+1 )
    {
        salary = 120000 - i*10000 ;
        salaries["employee#i#"] = salary ;

        employee = StructNew() ;
        employee["salary"] = salary ;
        // employee.salary = salary ;
        employees["employee#i#"] = employee ;

        departments["department#i#"] = StructNew() ;
```

```
        departments["department#i#"].boss = employee ;
    }
</cfscript>

<cfoutput>
<p>list of employees based on the salary (text search): <br>
1) #ArrayToList( StructSort( salaries ) )#<br>
2) #ArrayToList( StructSort( salaries, "text", "ASC" ) )#<br>
3) #ArrayToList( StructSort( salaries, "textnocase", "ASC" ) )#<br>
4) #ArrayToList( StructSort( salaries, "text", "DESC" ) )#<br>
<p>list of employees based on the salary (numeric search): <br>
5) #ArrayToList( StructSort( salaries, "numeric", "ASC" ) )#<br>
6) #ArrayToList( StructSort( salaries, "numeric", "DESC" ) )#<br>
<p>list of employees based on the salary (subfield search): <br>
7) #ArrayToList( StructSort( employees, "numeric", "DESC", "salary" ) )#<br>
8) #ArrayToList( StructSort( employees, "text", "ASC","salary" ) )#<br>
<p>list of departments based on the salary (sub-sub-field search): <br>
9) #ArrayToList( StructSort( departments, "text", "ASC", "boss.salary" ) )#<br>
</cfoutput>

<!--- add an invalid element and test that it throws an error --->
<p><p>
<cfset employees[ "employee4" ] = StructNew()>
<cftry>
    <cfset temp = StructSort( employees, "text", "ASC", "salary" )>
    <cfoutput>We have a problem - this was supposed to throw an exception!<br>
    </cfoutput>
<cfcatch type="any">
    <cfoutput>
        ERROR: <b>This error was expected!</b><br>
        #cfcatch.message# - #cfcatch.detail#<br>
    </cfoutput>
</cfcatch>
</cftry>
```

StructUpdate

Description
Updates a key with a value.

Returns
True, on successful execution; if the structure does not exist, ColdFusion throws an error.

Category
Structure functions

Function syntax
`StructUpdate(structure, key, value)`

See also
Structure functions; “Modifying a ColdFusion XML object” on page 878 in the *ColdFusion Developer’s Guide*

History
ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

Parameter	Description
`structure`	Structure to update
`key`	Key, the value of which to update
`value`	New value

Example
```
<!--- This example shows the use of StructUpdate. --->
<p>This file is similar to addemployee.cfm, which is called by StructNew,
    StructClear, and StructDelete. To test this file, copy the
    &LT;CFELSEIF&GT; statement to the appropriate place in
    addemployee.cfm. It is an example of a custom tag used to add
    employees. Employee information is passed through the employee
    structure (the EMPINFO attribute). In UNIX, you must also add the Emp_ID.

<cfswitch expression = "#ThisTag.ExecutionMode#">
<cfcase value = "start">
 <cfif StructIsEmpty(attributes.EMPINFO)>
<cfoutput>Error. No employee data was passed.</cfoutput>
        <cfexit method = "ExitTag">
 <cfelseIf StructFind(attributes.EMPINFO, "department") EQ "">
<cfscript>
 StructUpdate(attributes.EMPINFO, "department", "Unassigned");
</cfscript>
<cfexit method = "ExitTag">
           <cfelse>
```

Tan

Description
Calculates the tangent of an angle that is entered in radians.

Returns
A number; the tangent of an angle.

Category
Mathematical functions

Function syntax
`Tan(number)`

See also
Atn, Cos, ACos, Sin, ASin, Pi

Parameters

Parameter	Description
number	Angle, in radians, for which to calculate the tangent.

Usage
To convert degrees to radians, multiply degrees by π/180. To convert radians to degrees, multiply radians by 180/π.

Note: *Because the function uses floating point arithmetic, it can return a very small number (such as 6.12323399574E-017) for angles that should produce 0 and can return a very large number (such as 1.63312393532E+016) for infinity or not a number. To test for a 0 value, check whether the value is less than 0.0000000000001. To test for an infinite value, check whether the value is more than 1E15.*

Example
```
<h3>Tan Example</h3>
<!--- Calculate tangent if form has been submitted --->
<cfif IsDefined("FORM.tanNum")>
<!--- Make sure input is a number --->
    <cfif IsNumeric(#FORM.tanNum#)>
<!--- Convert degrees to radians, call the Tan function. --->
<cfset tanValue=#Tan((Form.tanNum * PI()) / 180)#>
<!--- 0.0000000000001 is the function's precision limit.
 If absolute value of returned value is
 less, set result to 0 --->
<cfif Abs(tanValue) LT 0.0000000000001>
 <cfset tanValue=0>
</cfif>
<cfoutput>
        Tan(#FORM.tanNum#) = #tanValue#<br><br>
</cfoutput>
    <cfelse>
<!--- If input is not a number, show an error message --->
        <h4>You must enter a numeric angle in degrees.</h4>
    </cfif>
</cfif>
<form action = "#CGI.script_name#" method="post">
Enter an angle in degrees to get its tangent:
<br><input type = "Text" name = "tanNum" size = "15">
```

```
<br><br>
<input type = "Submit" name = "">  
<input type = "RESET"
</form>
```

TimeFormat

Description

Formats a time value using U.S. English time formatting conventions.

Returns

A custom-formatted time value. If no mask is specified, returns a time value using the `hh:mm tt` format. For international time formatting, see `LSTimeFormat`.

Category

Date and time functions, Display and formatting functions

Function syntax

`TimeFormat(time [, mask ])`

See also

`CreateTime`, `Now`, `ParseDateTime`, `LSTimeFormat`, `DateFormat`

History

ColdFusion MX 6.1: Added the mask character L or l to represent milliseconds.

ColdFusion MX:

- Changed the way extra characters are processed: this function processes extra characters within the `mask` value differently than in earlier releases, as follows:
  - ColdFusion 5 and earlier: the function returns the time format and an apostrophe-delimited list of the extra characters. For example, `TimeFormat(Now(), "hh:mm:ss dog")` returns `8:17:23 d'o'g`.
  - ColdFusion MX: the function returns the time format and the extra characters. For example, for the call above, it returns `8:17:23 dog`.

  If the extra characters are single-quoted (for example, `hh:mm:ss 'dog'`), ColdFusion 5 and ColdFusion MX return the time format and the extra characters: `8:17:23 dog`.

1 Added support for the following *mask* parameter options: `short`, `medium`, `long`, and `full`.

Parameters

Parameter	Description
time	A date/time value or string to convert
mask	Masking characters that determine the format: • h: hours; no leading zero for single-digit hours (12-hour clock) • hh: hours; leading zero for single-digit hours (12-hour clock) • H: hours; no leading zero for single-digit hours (24-hour clock) • HH: hours; leading zero for single-digit hours (24-hour clock) • m: minutes; no leading zero for single-digit minutes • mm: minutes; a leading zero for single-digit minutes • s: seconds; no leading zero for single-digit seconds • ss: seconds; leading zero for single-digit seconds • l or L: milliseconds, with no leading zeros • t: one-character time marker string, such as A or P • tt: multiple-character time marker string, such as AM or PM • short: equivalent to h:mm tt • medium: equivalent to h:mm:ss tt • long: medium followed by three-letter time zone; as in, 2:34:55 PM EST • full: same as long

Usage

When passing a date/time value as a string, you must enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Database query results for date and time values can vary in sequence and formatting unless you use functions to format the results. To ensure that dates and times display with appropriate formatting, and that users of your ColdFusion application are not confused by dates and times displayed, Adobe recommends that you use the `DateFormat` and `TimeFormat` functions to format date and time values from queries. For more information and examples, see TechNote 22183, "*ColdFusion Server (5 and 4.5.x) with Oracle: Formatting Date and Time Query Results*," at www.coldfusion.com/Support/KnowledgeBase/SearchForm.cfm.

Example

```
<cfset todayDate = #Now()#>
<body>
<h3>TimeFormat Example</h3>
<p>Today's date is <cfoutput>#todayDate#</cfoutput>.
<p>Using Timeformat, we can display the value in different ways:
<cfoutput>
<ul>
    <li>#TimeFormat(todayDate)#
    <li>#TimeFormat(todayDate, "hh:mm:ss")#
    <li>#TimeFormat(todayDate, "hh:mm:sst")#
    <li>#TimeFormat(todayDate, "hh:mm:sstt")#
    <li>#TimeFormat(todayDate, "HH:mm:ss")#
</ul>
</cfoutput>
<p>To generate a standard ISO 8601 W3C Date and Time string like
1997-07-16T19:20, concatenate a DateFormat function, the character T, and a
```

```
TimeFormat function.
For example: dateformat(now(), "yyyy-mm-dd")#T#TimeFormat(now(), "HH:mm:ss")
produces:</p>
<cfoutput>#dateformat(now(), "yyyy-mm-dd")#T#TimeFormat(now(), "HH:mm:ss")#</cfoutput>
</body>
```

ToBase64

Description

Calculates the Base64 representation of a string or binary object. The Base64 format uses printable characters, allowing binary data to be sent in forms and e-mail, and stored in a database or file.

Returns

The Base64 representation of a string or binary object.

Category

Conversion functions, String functions

Function syntax

`ToBase64(string or binary_object[, encoding])`

See also

- `BinaryEncode` for converstion of binary data to base64
- `cffile` for information about loading and reading binary data
- `cfwddx` for information about serializing and deserializing binary data
- `IsBinary` and `ToBinary` for checking for binary data and converting a Base64 object to binary format

History

ColdFusion MX: Added the *encoding* parameter.

Parameters

Parameter	Description
string or binary_object	A string, the name of a string, or a binary object.
encoding	For a string, defines how characters are represented in a byte array. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information on character encoding, see: www.w3.org/International/O-charset.html. The default value is the encoding of the page on which the function is called. See `cfcontent`. For a binary object, this parameter is ignored.

Usage

Adobe recommends that you use the BinaryEncode function to convert binary data to Base64-encoded data in all new applications.

```
<h3>ToBase64 Example</h3>
<!--- Initialize data. ---->
<cfset charData = "">
<!--- Create string of ASCII characters (32-255); concatenate them --->
<cfloop index = "data" from = "32" to = "255">
    <cfset ch = chr(data)>
    <cfset charData = charData & ch>
</cfloop>
<p>
The following string is the concatenation of all characters (32 to 255)
    from the ASCII table.<br>
<cfoutput>#charData#</cfoutput>
</p>
<!--- Create a Base64 representation of this string. ---->
<cfset data64 = toBase64(charData)>

<!----- Convert string to binary. ------->
<cfset binaryData = toBinary(data64)>
<!--- Convert binary back to Base64. ---->
<cfset another64 = toBase64(binaryData)>
<!---- Compare another64 with data64 to ensure that they are equal. ---->
<cfif another64 eq data64>
    <h3>Base64 representations are identical.</h3>
<cfelse>
    <h3>Conversion error.</h3>
</cfif>
```

ToBinary

Description

Calculates the binary representation of Base64-encoded data.

Returns

The binary representation of Base64-encoded data.

Category

Conversion functions, String functions

Function syntax

ToBinary(string_in_Base64 or binary_value)

See also

- BinaryDecode for conversion of binary-encoded data, including Base64, to binary data
- cffile for information about loading and reading binary data
- cfwddx for information about serializing and deserializing binary data
- IsBinary and ToBase64 for checking format and converting to Base64
- Len for determining the length of a binary object
- "Binary data type and binary encoding" on page 31 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
string_in_Base64	A string in Base64 format to convert to binary.

Usage

Adobe recommends that you use the BinaryDecode function to convert Base64 encoded data to binary data in all new applications.

If you pass a binary value to this function, it returns the input value.

Example

```
<h3>ToBinary Example</h3>
<!---- Initialize data. ---->
<cfset charData = "">
<!---- Create a string of ASCII characters (32-255); concatenate them. ---->
<cfloop index = "data" from = "32" to = "255">
    <cfset ch = chr(data)>
    <cfset charData = charData & ch>
</cfloop>
<p>The following string is the concatenation of all characters (32 to 255)
    from the ASCII table.<br>
<cfoutput>#charData#</cfoutput></p>
<!----- Create a Base64 representation of this string. ----->
<cfset data64 = toBase64(charData)>

<!--- Convert string to binary. ---->
<cfset binaryData = toBinary(data64)>
<!--- Convert binary back to Base64. --->
<cfset another64 = toBase64(binaryData)>
```

```
<!---- Compare another64 with data64 to ensure that they are equal. ---->
<cfif another64 eq data64>
    <h3>Base64 representation of binary data is identical to the Base64
    representation of string data.</h3>
<cfelse>
    <h3>Conversion error.</h3>
</cfif>
```

ToScript

Description

Creates a JavaScript or ActionScript expression that assigns the value of a ColdFusion variable to a JavaScript or ActionScript variable. This function can convert ColdFusion strings, numbers, arrays, structures, and queries to JavaScript or ActionScript syntax that defines equivalent variables and values.

Returns

A string that contains a JavaScript or ActionScript variable definition corresponding to the specified ColdFusion variable value.

Category

Conversion functions, Extensibility functions

Function syntax

`ToScript(cfvar, javascriptvar, outputformat, ASFormat)`

See also

`cfwddx`; "WDDX JavaScript Objects" on page 1453 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`cfvar`	A ColdFusion variable. This can contain one of the following: • String • Number • Array • Structure • Query
`javascriptvar`	A string that specifies the name of the JavaScript variable that the `ToScript` function creates.
`outputformat`	Optional. A Boolean value that determines whether to create WDDX (JavaScript) or ActionScript style output for structures and queries: • True: creates WDDX-style output (default). • False: creates ActionScript-style output.
`ASFormat`	Optional. A Boolean value that specifies whether to use ActionScript shortcuts in the script: • True: creates new Objects and Arrays with ActionScript shortcuts: [] for New Array(), and {} for New Object. Using ActionScript shortcuts allows you to pass ActionScript into `cfform` attributes without triggering ActionScript validation. • False: does not use ActionScript shortcuts to create new Objects and new Arrays when generating the script. Instead, generates New Object() and New Array() in the script (default).

Usage

To use a ColdFusion variable in JavaScript or ActionScript, the `ToScript` function must be in a `cfoutput` region and be surrounded by number signs (#). For example, the following code uses the `ToScript` function to convert a ColdFusion variable to a JavaScript variable:

```
<cfset thisString="hello world">
<script type="text/javascript" language="JavaScript">
    <cfoutput>
        var #toScript(thisString, "jsVar")#;
    </cfoutput>
</script>
```

When ColdFusion runs this code, it sends the following to the client:

```
<script type="text/javascript" language="JavaScript">
    var jsVar = "hello world";
</script>
```

An HTML script tag must enclose the JavaScript code. The `cfoutput` tag does not need to be inside the script block; it can also surround the block.

WDDX-style output generates JavaScript code that creates a WDDXRecordset object, where the key of each record set entry is a column name, and the value of the recordlist entry is an array of the corresponding query column entries, as follows:

```
WDDXQuery = new WddxRecordset();
col0 = new Array();
col0[0] = "John";
col0[1] = "John";
WDDXQuery["firstname"] = col0;
col0 = null;
col1 = new Array();
col1[0] = "Lund";
col1[1] = "Allen";
WDDXQuery["lastname"] = col1;
col1 = null;
```

To use WDDX-style output, you must first load the *cf_webroot*/CFIDE/scripts/wddx.js script, which defines JavaScript WDDX objects, as in the following line:

```
<script type="text/javascript" src="/CFIDE/scripts/wddx.js"> </script>
```

For more information on WDDX in JavaScript, see "WDDX JavaScript Objects" on page 1453.

ActionScript-style output generates code that creates an array of objects, where the array is indexed by row number, and the objects consist of column name - column value pairs, as follows:

```
    ActionScriptQuery = new Array();
ActionScriptQuery[0] = new Object();
ActionScriptQuery[0]['firstname'] = "John";
ActionScriptQuery[0]['lastname'] = "Lund";
ActionScriptQuery[1] = new Object();
ActionScriptQuery[1]['firstname'] = "John";
ActionScriptQuery[1]['lastname'] = "Allen";
```

An ActionScript-style array does not require you to include the wddx.js file, and creates a variable that you can use in ActionScript on a Flash format form, for example, in an `onChange` attribute.

If the `outputformat` parameter is False, setting `ASFormat` to True causes `ToScript` to use the ActionScript shortcut `[]` in place of `New Array()` and the shortcut `{}` in place of `New Object()`. Using these shortcuts allows you to pass ActionScript into `cfform` attributes without triggering ActionScript validation. If `ASFormat` is False, `ToScript` generates `New Array()` and `New Object()` in the script.

Example

The following example shows the results of converting a ColdFusion string, array, and query object to JavaScript variables. It also uses the string and array in JavaScript code.

```
<h2>ToScript</h2>

<h3>Converting a string variable</h3>
<cfset thisString = "This is a string">
<cfoutput>
    <b>The thisString variable in ColdFusion</b><br>
    #thisString#<br>
    <br>
    <strong>The output of ToScript(thisString, "jsVar")</strong><br>
    #ToScript(thisString, "jsVar")#<br>
    <br>
    <strong>In a JavaScript script, convert thisString Variable to JavaScript
     and output the resulting variable:</strong><br>
    <script type="text/javascript" language="JavaScript">
        var #ToScript(thisString, "jsVar")#;
        document.write("jsVar in JavaScript is: " + jsVar);
    </script>
</cfoutput>

<h3>Converting an array</h3>
<!--- Create and populate a one-dimensional array --->
<cfset myArray=ArrayNew(1)>
<cfloop index="i" from="1" to="4">
            <cfset myArray[i]="This is array element" & i>
</cfloop>

<cfoutput>
<b>The ColdFusion myArray Array</b><br>
<!--- Write the contents of the myArray variable in ColdFusion --->
    <cfloop index="i" from="1" to="#arrayLen(myArray)#">
        myArry[#i#]: #myArray[i]#<br>
    </cfloop>
    <br>
    <strong>The output of ToScript(myArray, "jsArray")</strong><br>
    #toScript(myArray, "jsArray")#<br>
    <br>
    <strong>In JavaScript, convert myArray to a JavaScript variable and write it's
contents</strong><br>
    <script type="text/javascript" language="JavaScript">
        var #ToScript(myArray, "jsArray")#;
        for (i in jsArray)
        {
            document.write("myArray[" + i + "]: " + jsArray[i] + "<br>");
        }
    </script>
<br>
<h3>Converting a query</h3>
This section converts the following query object to both WDDX format
and ActionScript type JavaScript objects.<br>

<!--- Query a database --->
<cfquery name="thisQuery" datasource="cfdocexamples">
    SELECT FirstName,LastName
    FROM employee
    WHERE FirstName = 'John'
</cfquery>
<br>
The Query in ColdFusion
<cftable query="thisQuery" headerlines="1" colheaders>
    <cfcol align="left" width="9" header="<b>FirstName</b>" text="#FirstName#">
    <cfcol align="left" width="9" header="<b>LastName</b>" text="#LastName#">
```

```
</cftable>

<strong>JavaScript generated by ToScript(thisQuery, "WDDXQuery"):</strong><br>
    #toScript(thisQuery, "WDDXQuery")#;<br>
    <br>
<strong>JavaScript generated by ToScript(thisQuery, "ActionScriptQuery",
            False):</strong><br>
    #toScript(thisQuery, "ActionScriptQuery", False)#<br>
    <br>
<!--- Convert to both WDDX format and ActionScript format --->
<script type="text/javascript" language="JavaScript">
    #ToScript(thisQuery, "WDDXQuery")#;
    #ToScript(thisQuery, "ActionScriptQuery", False)#;
</script>
<!--- For brevity, this example does not use JavaScript query variables --->
</cfoutput>
```

ToString

Description
Converts a value to a string.

Returns
A string.

Category
Conversion functions, String functions

Function syntax
`ToString(value[, encoding])`

See also
`ToBase64`, `ToBinary`, `CharsetEncode`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX:

- Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode character values 0–65535. (ColdFusion 5 and earlier releases supported ASCII values 1–255.)
- Added the *encoding* parameter.
- Added ability to convert an XML document object to a string.

Parameters

Parameter	Description
value	Value to convert to a string; can be a simple value such as an integer, a binary object, or an XML document object.
encoding	The character encoding (character set) of the string. Optional for binary data, Generates an error if used for a simple value or XML document object. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information on character encoding, see: www.w3.org/International/O-charset.html. The default value is the encoding of the page on which the function is called. See `cfcontent`.

Usage

This function can convert simple values and binary values that do not contain Byte zero. If this function cannot convert a value, it throws an exception. This function can also convert an XML document object to a string XML representation.

Adobe recommends that you use the `CharsetEncode` function to convert binary data to a string.

Example

```
<h3>ToString Example</h3>
<!---- Initialize data. ------>
<cfset charData = "">
<!----- Create string of ASCII characters (32-255) and concatenate them. ---->
<cfloop index = "data" from = "32" to = "255">
    <cfset ch = chr(data)>
    <cfset charData = charData & ch>
</cfloop>
<p>The following string is the concatenation of characters (32 to 255)
    from the ASCII table.<br>
<cfoutput>#charData#</cfoutput></p>

<!------ Create a Base64 representation of this string. ---->
<cfset data64 = toBase64(#charData#)>
<p>
The following string is the Base64 representation of the string.<br>
<cfoutput>#data64#</cfoutput></p>
<!---- Create a binary representation of Base64 data. --->
<cfset dataBinary = toBinary(data64)>

<!---- Create the string representation of the binary data. ----->
<cfset dataString = ToString(dataBinary)>
<p>The following is the string representation of the binary data.<br>
<cfoutput>#dataString#</cfoutput></p>
```

Trim

Description

Removes leading and trailing spaces and control characters from a string.

Returns

A copy of the *string* parameter, after removing leading and trailing spaces and control characters.

Category

String functions

Function syntax

```
Trim(string)
```

See also

LTrim, RTrim

Parameters

Parameter	Description
string	A string or a variable that contains a string.

Example

```
<h3>Trim Example</h3>
<cfif IsDefined("FORM.myText")>
    <cfoutput>
        <pre>
            Your string:"#FORM.myText#"
            Your string:"#Trim(FORM.myText)#"
            (trimmed on both sides)
        </pre>
    </cfoutput>
</cfif>
<form method = "post" action = "trim.cfm">
<p>Type in some text, and it will be modified by trim to remove leading
    spaces from the left and right
<p><input type = "Text" name = "myText" value = " TEST ">
<p><input type = "Submit" name = "">
</form>
```

UCase

Description

Converts the alphabetic characters in a string to uppercase.

Returns

A copy of a string, converted to uppercase.

Category

String functions

Function syntax

```
UCase(string)
```

See also

LCase

Parameters

Parameter	Description
string	A string or a variable that contains one

Example

```
<h3>UCase Example</h3>

<cfif IsDefined("FORM.sampleText")>
    <cfif FORM.sampleText is not "">
        <p>Your text, <cfoutput>#FORM.sampleText#</cfoutput>,
    returned in uppercase is <cfoutput>#UCase(FORM.sampleText)#</cfoutput>.
    <cfelse>
        <p>Please enter some text.
    </cfif>
</cfif>

<form action = "ucase.cfm">
<p>Enter your sample text, and press "submit" to see the text returned in
    uppercase:
<p><input type = "Text" name = "SampleText" value = "sample">

<input type = "Submit" name = "" value = "submit">
</form>
```

URLDecode

Description
Decodes a URL-encoded string.

Returns
A copy of a string, decoded.

Category
Conversion functions, Other functions, String functions

Function syntax
`URLDecode(urlEncodedString[, charset])`

See also
`URLEncodedFormat`; "Tags and functions for globalizing applications" on page 345 in the *ColdFusion Developer's Guide*

ColdFusion MX 6.1: Changed the default charset: the default charset is the character encoding of the URL scope.

ColdFusion MX:

- Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode character values 0–65535. (Earlier releases supported ASCII values.)
- Added the `charset` parameter.

Parameters

Parameter	Description
`urlEncodedString`	URL-encoded string or a variable that contains one.
`charset`	The character encoding in which the URL is encoded. Optional. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information on character encoding, see: www.w3.org/International/O-charset.html. The default value is the character encoding of the URL scope. See `SetEncoding`.

Usage

URL encoding formats some characters with a percent sign and the two-character hexadecimal representation of the character. For example, a character whose code is 129 is encoded as `%81`. A space is encoded with a plus sign.

Query strings in HTTP are always URL-encoded.

Example

This example creates, encodes, and decodes a string that contains ASCII character codes:

```
<cfscript>
    // Build string
    s = "";
    for (c = 1; c lte 256; c = c + 1)
    {
        s = s & chr(c);
    }
    // Encode string and display result
    enc = URLEncodedFormat(s);
    WriteOutput("Encoded string is: '#enc#'.<br>");
    // Decode and compare result with original
    dec = URLDecode(enc);
    if (dec neq s)
    {
        WriteOutput("Decoded is not the same as encoded.");
    }
    else
    {
        WriteOutput("All's quiet on the Western front.");
    }
</cfscript>
```

URLEncodedFormat

Description

Generates a URL-encoded string. For example, it replaces spaces with `%20`, and non-alphanumeric characters with equivalent hexadecimal escape sequences. Passes arbitrary strings within a URL (ColdFusion automatically decodes URL parameters that are passed to a page).

Returns

A copy of a string, URL-encoded.

Category

Conversion functions, Other functions, String functions

Function syntax

`URLEncodedFormat(string [, charset ])`

See also

`URLDecode`; "Tags and functions for globalizing applications" on page 345 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 6.1: Changed the default encoding to be the response character encoding.

ColdFusion MX: Added the `charset` parameter.

Parameters

Parameter	Description
`string`	A string or a variable that contains one
`charset`	The character encoding in which the string is encoded. Optional. The following list includes commonly used values: • utf-8 • iso-8859-1 • windows-1252 • us-ascii • shift_jis • iso-2022-jp • euc-jp • euc-kr • big5 • euc-cn • utf-16 For more information on character encoding, see: www.w3.org/International/O-charset.html. The default value is the character encoding of the response. See `cfcontent`.

Usage

URL encoding formats some characters with a percent sign and the two-character hexadecimal representation of the character. For example, a character whose code is 129 is encoded as `%81`. A space is encoded with a plus sign.

Query strings in HTTP are always URL-encoded.

Example

```
<h3>URLEncodedFormat Example</h3>
<cfif IsDefined("url.myExample")>
    <p>The url variable url.myExample was passed from the previous link ...
    its value is:
    <br><b>"<cfoutput>#url.myExample#</cfoutput>"</b>
</cfif>
<p>This function returns a URL encoded string.
<cfset s = "My url-encoded string has special characters & other stuff">
<p> <A HREF = "urlencodedformat.cfm?myExample=<cfoutput>#URLEncodedFormat(s)#
</cfoutput>">Click me</A>
```

URLSessionFormat

Description

Depending on whether a client computer accepts cookies, this function does the following:

- If the client does not accept cookies: automatically appends all required client identification information to a URL
- If the client accepts cookies: does not append information

This function automatically determines which identifiers are required, and sends only the required information. It provides a more secure and robust method for supporting client identification than manually encoding the information in each URL, because it sends only required information, when it is required, and it is easier to code.

Returns

A URL; if cookies are disabled for the browser, client and session data are appended.

Category

Other functions; "Maintaining client identity" on page 276 in the *ColdFusion Developer's Guide*

Function syntax

`URLSessionFormat(request_URL)`

Parameters

Parameter	Description
`request_URL`	URL of a ColdFusion page

Usage

In the following example, the `cfform` tag posts a request to another page and sends the client identification, if required. If cookie support is detected, the function returns the following:

```
myactionpage.cfm
```

If the detected cookie is not turned on, or cookie support cannot be reliably detected, the function return value is as follows:

```
myactionpage.cfm?jsessionid=xxxx;cfid=xxxx&cftoken=xxxxxxxx
```

Example

```
<cfform
    method="Post"
    action="#URLSessionFormat("MyActionPage.cfm")#">
</cfform>
```

Val

Description

Converts numeric characters that occur at the beginning of a string to a number.

Returns

A number. If conversion fails, returns zero.

Category

Conversion functions, String functions

Function syntax

`Val(string)`

See also

`IsNumeric`

Parameters

Parameter	Description
`string`	A string or a variable that contains one

Usage

This function works as follows:

- If `TestValue = "234A56?7'"`, `Val(TestValue)` returns 234.
- If `TestValue = "234'5678'9?'"`, `Val(TestValue)` returns 234.
- If `TestValue = "BG234"`, `Val(TestValue)` returns the value 0, (not an error).
- If `TestValue = "0"`, `Val(TestValue)` returns the value 0, (not an error).

Example

```
<h3>Val Example</h3>
<cfif IsDefined("FORM.theTestValue")>
    <cfif Val(FORM.theTestValue) is not 0>
        <h3>The string <cfoutput>#DE(FORM.theTestValue)#</cfoutput>
    can be converted to a number:
        <cfoutput>#Val(FORM.theTestValue)#</cfoutput></h3>
    <cfelse>
        <h3>The beginning of the string <cfoutput>#DE(FORM.theTestValue)#
         </cfoutput> cannot be converted to a number</h3>
    </cfif>
</cfif>
<form action = "val.cfm">
<p>Enter a string, and determine whether its beginning can be evaluated
    to a numeric value.
<p>
<input type = "Text"
    name = "TheTestValue"
    value = "123Boy">
<input type = "Submit"
    value = "Is the beginning numeric?"
    name = "">
</form>
```

ValueList

Description

Inserts a delimiter between each value in an executed query. ColdFusion does not evaluate the arguments.

Returns

A delimited list of the values of each record returned from an executed query.

Category

List functions, Query functions

Function syntax

`ValueList(query.column [, delimiter ])`

See also

`QuotedValueList`

Parameters

Parameter	Description
`query.column`	Name of an executed query and column. Separate query name and column name with a period.
`delimiter`	A delimiter character to separate column data items. The default value is comma (,).

Example

```
<h3>ValueList Example</h3>

<!--- use the contents of a query to create another dynamically --->
<cfquery name = "GetDepartments" datasource = "cfdocexamples">
    SELECT Dept_ID FROM Departments
    WHERE Dept_ID IN ('BIOL')
</cfquery>

<cfquery name = "GetCourseList" datasource = "cfdocexamples">
    SELECT *
    FROM CourseList
    WHERE Dept_ID IN ('#GetDepartments.Dept_ID#')
</cfquery>

Value list of all BIOL Course ID's using (--) as the delimiter:<br>
<cfoutput>
#ValueList(GetCourseList.Course_ID,"--")#<br>
</cfoutput>

Value list of all BIOL Course Numbers using (;) as the delimiter:<br>
<cfoutput>
#ValueList(GetCourseList.CorNumber,";")#<br>
</cfoutput>
```

VerifyClient

Description
Requires remote invocations of the page or calls to functions on the page to include an encrypted security token.

Returns
Does not return a value.

Category
Security functions

Function syntax
```
VerifyClient()
```

See also
`cffunction`, “Improving security” on page 674 in the *ColdFusion Developer’s Guide*

History
ColdFusion 8: Added this function.

Parameters
Does not take any parameters

Usage
Use this function to help prevent security attacks where an unauthorized party attempts to perform an action on the server, such as changing a password. As a general rule, you should use this feature for Ajax requests to the server to perform sensitive actions, such as updating passwords.

If you call this function, you must enable client management or session management in your application; otherwise, you do not get an error, but ColdFusion does not verify clients. Use this function only on pages that respond to client-side ColdFusion Ajax features, such as bind expressions. These features include code that correctly sends the security token when needed.

Week

Description

From a date/time object, determines the week number within the year.

Returns

An integer in the range 1–53; the ordinal of the week, within the year.

Category

Date and time functions

Function syntax

`Week(date)`

See also

`DatePart`

Parameters

Parameter	Description
`date`	A date/time object in the range 100 AD–9999 AD.

Usage

When passing date as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date.

Example

```
<h3>Week Example</h3>
<cfif IsDefined("FORM.year")>
More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
    <p>Your date, #DateFormat(yourDate)#.
    <br>It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day
         #DayOfWeek(yourDate)# in the week.
    <br>This is day #Day(YourDate)# in the month of
         #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days.
    <br>We are in week #Week(yourDate)# of #Year(yourDate)# (day
         #DayOfYear(yourDate)# of #DaysInYear(yourDate)#). <br>
    <cfif IsLeapYear(Year(yourDate))>This is a leap year
        <cfelse>This is not a leap year
    </cfif>
</cfoutput>
</cfif>
```

Wrap

Description

Wraps text so that each line has a specified maximum number of characters.

Note: *The wrap function does not insert line breaks by placing the* `<br>` *tag in HTML text. Instead, it wraps the text in the display without adding the* `<br>` *tag.*

Returns

String containing the wrapped text.

Category

String functions

Function syntax

`Wrap(string, limit[, strip])`

See also

`cfmail`

History

ColdFusion MX 6.1: Added this function.

Parameters

Parameter	Description
`string`	String or variable that contains one. The text to wrap.
`limit`	Positive integer maximum number of characters to allow on a line.
`strip`	Boolean value specifying whether to remove all existing newline and carriage return characters in the input string with spaces before wrapping the text. The default value is False.

Usage

Inserts line break at the location of the first white space character (such as a space, tab, or new line) before the specified limit on a line. If a line has no whitespace characters before the limit, inserts a line break at the limit. Uses the operating-system specific line break: newline for UNIX, carriage return and newline on Windows.

If you specify the `strip` parameter, all existing line breaks are removed, so any paragraph formatting is lost.

Use this function to limit the length of text lines, such as text to be included in a mail message. The `cfmail` and `cfmailpart` tag `wraptext` attributes use this function

Example

```
<h3>Wrap Example</h3>
<cfset inputText="This is an example of a text message that we want to wrap. It is rather
long and needs to be broken into shorter lines.">
<cfoutput>#Wrap(inputText, 59)#</cfoutput>
```

WriteOutput

Description

Appends text to the page-output stream.

This function writes to the page-output stream regardless of conditions established by the `cfsetting` tag.

Category

Other functions, System functions

Function syntax

`WriteOutput(`*`string`*`)`

Parameters

Parameter	Description
`string`	A string or a variable that contains one

Usage

Within the `cfquery` and `cfmail` tags, this function does not output to the current page; it writes to the current SQL statement or mail text. Do not use `WriteOutput` within `cfquery` and `cfmail`.

Although you can call this function anywhere within a page, it is most useful inside a `cfscript` block.

Example

```
...
<cfscript>
employee = StructNew();
StructInsert(employee, "firstname", FORM.firstname);
StructInsert(employee, "lastname", FORM.lastname);
StructInsert(employee, "email", FORM.email);
StructInsert(employee, "phone", FORM.phone);
StructInsert(employee, "department", FORM.department);
WriteOutput("About to add " & FORM.firstname & " " & FORM.lastname);
</cfscript>
```

XmlChildPos

Description
Gets the position of a child element within an XML document object.

Returns
The position, in an XmlChildren array, of the *N*th child that has the specified name.

Category
XML functions

Function syntax
`XmlChildPos(elem, childName, N)`

See also
`IsXmlElem`, `XmlElemNew`, `XmlSearch`, `XmlTransform`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Added this function.

Parameters

Parameter	Description
`elem`	XML element within which to search.
`childName`	XML child element for which to search. Must be an immediate child of the `elem` parameter.
`N`	Index of XMLchild element for which to search.

Usage
You can use the returned index in the `ArrayInsertAt` and `ArrayDeleteAt` functions to change XML document objects. If the specified child is not found, the function returns -1.

Example
The following example searches XML document element, `xmlobject.employee.name[1]`, for its second `Status` element child and uses the position in an `ArrayDeleteAt` function to remove the element:

```
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<employee>
    <!-- A list of employees -->
    <name EmpType="Regular">
        <first>Almanzo</first>
        <last>Wilder</last>
        <Status>Medical Absence</Status>
        <Status>Extended Leave</Status>
    </name>
    <name EmpType="Contract">
        <first>Laura</first>
        <last>Ingalls</last>
    </name>
</employee>
</cfxml>
```

```
<!--- Find the second Status child of the first employee.name element --->
<cfscript>
elempos=XMLChildPos(xmlobject.employee.name[1], "Status", 2);
ArrayDeleteAt(xmlobject.employee.name[1].XmlChildren, elempos);
</cfscript>

<!--- Dump the resulting document object to confirm the deletion --->
<cfdump var="#xmlobject#">
```

XmlElemNew

Description
Creates an XML document object element.

Returns
An XML document object element.

Category
XML functions

Function syntax
`XmlElemNew(xmlObj[, namespace], childName)`

See also
`cfxml`, `IsXmlElem`, `XmlChildPos`, `XmlFormat`, `XmlNew`, `XmlParse`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX 7: Added the `namespace` parameter.

ColdFusion MX: Added this function.

Parameters

Parameter	Description
`xmlObj`	Name of the XML document object in which you are creating the element
`namespace`	(Optional) URI of the namespace to which this element belongs
`childName`	Name of the element to create

Usage
The function's return variable specifies the location of the new element in the document object. It must specify a valid location in the document object identified by the *xmlObj* parameter. The following statements show this use:

```
MyDoc.MyRoot.XmlChildren[2] = XmlElemNew(MyDoc,"childNode");
ArrayAppend(MyDoc.MyRoot.XmlChildren, XmlElemNew(MyDoc,"childNode"));
```

If you do not specify a `namespace` URI and use a namespace prefix in the `childName` parameter, ColdFusion checks to see if a namespace URI has already been specified for the prefix, and if so, uses that namespace.

Example
The following example creates and displays a ColdFusion document object:

```
<cfscript>
    MyDoc = XmlNew();
    MyDoc.xmlRoot = XmlElemNew(MyDoc,"MyRoot");
    if (testVar IS TRUE)
        MyDoc.MyRoot.XmlText = "The value of testVar is True.";
    else
        MyDoc.MyRoot.XmlText = "The value of testVar is False.";
    for (i = 1; i LTE 4; i = i + 1)
        {
        MyDoc.MyRoot.XmlChildren[i] = XmlElemNew(MyDoc,"childNode");
        MyDoc.MyRoot.XmlChildren[i].XmlText = "This is Child node " & i &".";
```

```
        }
</cfscript>
<cfdump var=#MyDoc#>
```

XmlFormat

Description
Escapes special XML characters in a string so that the string can be used as text in XML.

Returns
A copy of the *string* parameter that is safe to use as text in XML.

Category
String functions, XML functions

Function syntax
`XmlFormat(string)`

See also
`cfxml`, `XmlNew`, `XmlParse`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Added this function.

Parameters

Parameter	Description
`string`	A string or a variable that contains one

Usage
This function escapes characters as follows:

Text character	Escaped representation
Greater than symbol (>)	>
Less than symbol (<)	<
Single-quotation mark (')	'
Double-quotation mark (")	"
Ampersand symbol (&)	&
Carriage return (but not line feed)	Removed from the text.
High ASCII characters in the range 128-255.	Replaced by unicode escape sequence; for example, É (capital E with an Acute symbol) is replaced by É.

Example
The following example shows how `XmlFormat` escapes special XML characters. Use the View Source command in the browser to see the results. ColdFusion interprets the "" in the second text string as representing a single-quotation mark in text before it applies the `XmlFormat` function.

```
<?xml version = "1.0"?>
<cfoutput>
<someXML>
    <someElement someAttribute="#XmlFormat("'a quoted value'")#">
        #XmlFormat("Body of element with <, >, "" and & goes here.")#
    </someElement>
```

```
</someXML>
</cfoutput>
```

XmlGetNodeType

Description

Determines the type of an XML document object node.

Returns

A string identifying the XML node type. The following values are valid:

ATTRIBUTE_NODE	CDATA_SECTION_NODE
COMMENT_NODE	DOCUMENT_FRAGMENT_NODE
DOCUMENT_NODE	DOCUMENT_TYPE_NODE
ELEMENT_NODE	ENTITY_NODE
ENTITY_REFERENCE_NODE	NOTATION_NODE
PROCESSING_INSTRUCTION_NODE	TEXT_NODE

If the argument is not a document object node, the function generates an error.

Category

XML functions

Function syntax

`XmlGetNodeType(`*`xmlNode`*`)`

See also

`IsXmlAttribute`, `IsXmlDoc`, `IsXmlElem`, `IsXmlNode`, `IsXmlRoot`, `XmlChildPos`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
`xmlNode`	An XML DOM object node

Usage

The `XmlGetNodeType` function can determine the types of the nodes returned by the `XmlSearch` function, or the types of the entries in an element's XmlNodes array.

Example

The following example checks the node types of various parts of an XML document object:

```
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<?xml version="1.0" encoding="UTF-8"?>
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <!-- This item is coded to show several node types -->
            <![CDATA["Our Best" hammer & chisel set!!!]]> Imported from France
```

```
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
    </items>
</order>
</cfxml>

<!--- Display the node types --->
<cfoutput>
<h3>Node Types</h3>
xmlobject: #XMLGetNodeType(xmlobject)#<br>
xmlobject.order: #XMLGetNodeType(xmlobject.order)#<br>
<br>
Now check the types of all the nodes in the xmlobject.order.items.item
element's XmlNodes array.<br>
    Note the many apparently empty Text nodes generated by whitespace characters in the XML
text source.<br><br>
<cfset descnodes=xmlobject.order.items.item.XmlNodes>
<cfloop from="1" to="#ArrayLen(descnodes)#" index="i">
    #i# Node type is: #XMLGetNodeType(descnodes[i])#<br>
    #i# Node name is: #descnodes[i].XmlName#<br>
    <cfif (descnodes[#i#].XmlValue NEQ "")>
        #i# Node value is: #descnodes[i].XmlValue#<br>
    </cfif>
    <br>
</cfloop>
</cfoutput>
```

XmlNew

Description
Creates an XML document object.

Returns
An empty XML document object.

Category
XML functions

Function syntax
`XmlNew([caseSensitive])`

See also
`cfxml`, `IsXmlDoc`, `ToString`, `XmlFormat`, `XmlParse`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX: Added this function.

Parameters

Parameter	Description
`caseSensitive`	Determines how ColdFusion processes the case of XML document object component identifiers: • True: maintains case • False: ColdFusion ignores case. Default.

Usage
An XML document object is represented in ColdFusion as a structure.

The *caseSensitive* parameter value determines whether identifiers whose characters are of varying case, but are otherwise the same, refer to different components; for example:

- If True, the element or attribute names "name" and "NAME" refer to different elements or attributes.
- If False, these names refer to the same elements or attributes.

If your XML object is case sensitive, you cannot use dot notation to reference an element or attribute name. Use the name in associative array (bracket) notation, or a reference that does not use the case-sensitive name (such as xmlChildren[1]) instead. In the following code, the first line will work with a case-sensitive XML object. The second and third lines cause errors:

```
MyDoc.xmlRoot.XmlAttributes["Version"] = "12b";
MyDoc.xmlRoot.XmlAttributes.Version = "12b";
MyDoc.MyRoot.XmlAttributes["Version"] = "12b";
```

To convert an XML document object into a string, use the `ToString` function.

Example
The following example creates and displays a ColdFusion document object:

```
<cfset testVar = True>
<cfscript>
```

```
    MyDoc = XmlNew();
    MyDoc.xmlRoot = XmlElemNew(MyDoc,"MyRoot");
    if (testVar IS TRUE)
        MyDoc.MyRoot.XmlText = "The value of testVar is True.";
    else
        MyDoc.MyRoot.XmlText = "The value of testVar is False.";
    for (i = 1; i LTE 4; i = i + 1){
        MyDoc.MyRoot.XmlChildren[i] = XmlElemNew(MyDoc,"childNode");
        MyDoc.MyRoot.XmlChildren[i].XmlText = "This is Child node " & i &".";
        }
</cfscript>
<cfdump var=#MyDoc#>
```

XmlParse

Description
Converts XML text into an XML document object.

Returns
An XML document object.

Category
Conversion functions, XML functions

Function syntax
`XmlParse(xmlText [, caseSensitive ], validator])`

See also
`cfxml`, `IsXML`, `ToString`, `XmlFormat`, `XmlNew`, `XmlSearch`, `XmlTransform`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion MX 7:

- Added the `validator` parameter.
- Added support for filenames and URLs in the *`xmlText`* parameter.
- Added support for relative URLs and pathnames.

ColdFusion MX: Added this function.

Parameters

Parameter	Description
`xmlText`	Any of the following: • A string containing XML text. • The name of an XML file. • The URL of an XML file; valid protocol identifiers include http, https, ftp, and file.
`caseSensitive`	• Yes: maintains the case of document elements and attributes. • No: Default
`validator`	Any of the following: • The name of a Document Type Definition (DTD) or XML Schema file. • The URL of a DTD or Schema file; valid protocol identifiers include http, https, ftp, and file. • A string representation of a DTD or Schema. • An empty string; in this case, the XML file must contain an embedded DTD or Schema identifier, which is used to validate the document.

Usage
If you specify a relative URL or pathname in a parameter, ColdFusion uses the directory (or, for URLs, the logical directory) that contains the current ColdFusion page as the path root.

The *caseSensitive* parameter value determines whether identifiers whose characters are of varying case, but are otherwise the same, refer to different components; for example:

- If true, the element or attribute names “name” and “NAME” refer to different elements or attributes.
- If false, these names refer to the same elements or attributes.

If your XML object is case sensitive, you cannot use dot notation to reference an element or attribute name. Use the name in associative array (bracket) notation, or a reference that does not use the case-sensitive name (such as xmlChildren[1]) instead. In the following code, the first line will work with a case-sensitive XML object. The second and third lines cause errors:

```
MyDoc.xmlRoot.XmlAttributes["Version"] = "12b";
MyDoc.xmlRoot.XmlAttributes.Version = "12b";
MyDoc.MyRoot.XmlAttributes["Version"] = "12b";
```

The optional *validator* parameter specifies a DTD or Schema to use to validate the document. If the parser encounters a validation error, ColdFusion generates an error and stops parsing the document. You must specify a *validator* parameter to make the `XmlParse` function validate your document. If you do not specify a *validator* parameter, and the XML file specifies a DTD or Schema, ColdFusion ignores the DTD or Schema. If you specify a *validator* parameter, you must also specify a *caseSensitive* parameter.

If you do not specify a *validator* parameter, the *xmlText* parameter can specify a well-formed XML fragment, and does not have to specify a complete document.

Note: *To convert an XML document object back into a string, use the `ToString` function.*

Example

The following example has three parts: an XML file, a DTD file, and a CFML page that parses the XML file and uses the DTD for validation. The CFML file displays the returned XML document object. To show the results of invalid XML, modify the bmenuD.xml.

Note: *The DTD used in the following example represents the same XML structure as the Schema used in the `XmlValidate` example*

The custorder.xml file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE order SYSTEM "C:\CFusionMX7\wwwroot\examples\custorder.dtd">
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <name>Deluxe Carpenter&apos;s Hammer</name>
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
        <item id="54">
            <name>36&quot; Plastic Rake</name>
            <quantity>2</quantity>
            <unitprice>6.95</unitprice>
        </item>
        <item id="68">
            <name>Standard paint thinner</name>
            <quantity>3</quantity>
            <unitprice>8.95</unitprice>
        </item>
    </items>
</order>
```

The custorder.dtd file is as follows:

```
<!ELEMENT order (customer, items)>
<!ATTLIST order
    id CDATA #REQUIRED>
<!ELEMENT customer EMPTY>
<!ATTLIST customer
    firstname CDATA #REQUIRED
    lastname CDATA #REQUIRED
    accountNum CDATA #REQUIRED>
<!ELEMENT items (item+)>
<!ELEMENT item (name, quantity, unitprice)>
<!ATTLIST item
    id CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>
<!ELEMENT unitprice (#PCDATA)>
```

The CFML file is as follows. It uses a filename for the XML file and a URL for the DTD. Note that the XML and URL paths must be absolute.

```
<cfset
myDoc=XMLParse("C:\CFusionMX7\wwwroot\examples\custorder.xml",
false, "http://localhost:8500/examples/custorder.dtd")>
Dump of myDoc XML document object<br>
<cfdump var="#myDoc#">
```

XmlSearch

Description
Uses an XPath language expression to search an XML document object.

Returns
The results of the XPath search. For details, see Usage.

Category
XML functions

Function syntax
`XmlSearch(xmlDoc, xPathString)`

See also
`cfxml`, `IsXML`, `XmlChildPos`, `XmlParse`, `XmlTransform`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History
ColdFusion 8: Added support for returning any valid XPath result, not just arrays of XML object nodes.

ColdFusion MX 7: Added support for attribute searches.

ColdFusion MX: Added this function.

Parameters

Parameter	Description
`xmlDoc`	XML document object
`xPathString`	XPath expression

Usage
The `XmlSearch` function attempts to return the values returned by the search whenever possible. For example, if the XPath expression returns a Boolean, the CFML variable is assigned a `true` or `false` value.

The following table lists XPath expression result data types and how they are represented in the CFML return value.

XPath return type	ColdFusion representation
Boolean	Boolean
Null	"" (empty string)
Number	Number
String	String
NodeSet	Array of XML nodes
Result Tree Fragment	Array of XML nodes

Results that are `Unknown` or have an unresolved variable in the expression throw an error.

XPath is specified by the World Wide Web Consortium (W3C). For detailed information on XPath, including XPath expression syntax, see the W3C website at www.w3.org/TR/xpath.

Example

The following example extracts the elements named *last*, which contain employee last names, from an XML file, and displays the names.

The employeesimple.xml file contains the following XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<employee>
<!-- A list of employees -->
 <name EmpType="Regular">
<first>Almanzo</first>
<last>Wilder</last>
 </name>
 <name EmpType="Contract">
<first>Laura</first>
<last>Ingalls</last>
 </name>
</employee>
```

The CFML file contains the following lines:

```
<cfscript>
    myxmldoc = XmlParse("C:\CFusionMX7\wwwroot\examples\employeesimple.xml");
    selectedElements = XmlSearch(myxmldoc, "/employee/name/last");
    for (i = 1; i LTE ArrayLen(selectedElements); i = i + 1)
        writeoutput(selectedElements[i].XmlText & "<br>");
</cfscript>
```

XmlTransform

Description

Applies an Extensible Stylesheet Language Transformation (XSLT) to XML. The XML can be in string format or an XML document object.

Returns

A string containing the results of applying the XSLT to the XML.

Category

Conversion functions, XML functions

Function syntax

`XmlTransform(xml, xsl[, parameters])`

See also

`cfxml`, `XmlFormat`, `XmlNew`, `XmlParse`, `XmlSearch`, `XmlValidate`; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added the `parameters` parameter and the ability to use a file for the XSL.

ColdFusion MX: Added this function.

Parameters

Parameter	Description
`xml`	An XML document in string format, or an XML document object
`xsl`	XSLT transformation to apply; can be any of the following: Any of the following: • A string containing XSL text. • The name of an XSTLT file. Relative paths start at the directory containing the current CFML page. • The URL of an XSLT file; valid protocol identifiers include http, https, ftp, and file. Relative paths start at the directory containing the current CFML page.
`parameters`	A structure containing XSL template parameter name-value pairs to use in transforming the document. The XSL transform defined in the *`xslString`* parameter uses these parameter values in processing the XML.

Usage

An XSLT converts an XML document to another format or representation by applying an Extensible Stylesheet Language (XSL) stylesheet to it. XSL, including XSLT syntax is specified by the World Wide Web Consortium (W3C). For detailed information on XSL and XSLT, see the W3C website at www.w3.org/Style/XSL/.

If the XSLT code contains include statements with relative paths, ColdFusion resolves them relative to the location of the XSLT file, or for an XSL string, the location of the current ColdFusion page.

Example

The following example converts an XML document that represents a customer order into an HTML document with the customer name and a table with the order items and quantities:

The custorder.xml file that represents a customer order has the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <name>Deluxe Carpenter&amp;apos;s Hammer</name>
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
        <item id="54">
            <name>36&amp;quot; Plastic Rake</name>
            <quantity>2</quantity>
            <unitprice>6.95</unitprice>
        </item>
        <item id="68">
            <name>Standard paint thinner</name>
            <quantity>3</quantity>
            <unitprice>8.95</unitprice>
        </item>
    </items>
</order>
```

The custorder.xsd XSLT file that transforms the XML to HTML that displays the customer's name, and the items and quantities ordered has the following lines:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" doctype-public="-//W3C//DTD HTML 4.0 Transitional//EN" />
    <xsl:template match="/">
        <html>
            <body>
                <table border="2" bgcolor="yellow">
                    <tr>
                        <th>Name</th>
                        <th>Price</th>
                    </tr>
                    <xsl:for-each select="breakfast_menu/food">
                        <tr>
                            <td>
                                <xsl:value-of select="name"/>
                            </td>
                            <td>
                                <xsl:value-of select="price"/>
                            </td>
                        </tr>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

The CFML file has the following lines:

```
<cffile action="read" file="C:\CFusionMX7\wwwroot\examples\custorder.xsl"
variable="xmltrans">
<cfset xmldoc = XmlParse("C:\CFusionMX7\wwwroot\examples\custorder.xml")>
<cfoutput>#XmlTransform(xmldoc, xmltrans)#</cfoutput>
```

XmlValidate

Description

Uses a Document Type Definition (DTD) or XML Schema to validate an XML text document or an XML document object.

Returns

The following validation structure:

Field	Description
Errors	An array containing any validator error messages. These messages indicate that the document does not conform to the DTD or Schema (is not valid).
FatalErrors	An array containing any validator fatal error messages. Fatal errors indicate that the document contains XML formatting errors (is not well-formed XML).
Status	A Boolean value: • True if the document is valid. • False if the validation check failed.
Warning	An array containing any validator warnings. A well-formed and valid document can produce warning messages.

Category

XML functions

Function syntax

```
XmlValidate(xmlDoc[, validator])
```

See also

cfxml, IsXmlDoc, IsXML, XmlFormat, XmlNew, XmlParse, XmlSearch, XmlTransform; "Using XML and WDDX" on page 867 in the *ColdFusion Developer's Guide*

History

ColdFusion MX 7: Added this function.

Parameters

Parameter	Description
xmlDoc	Any of the following: • A string containing an XML document. • The name of an XML file. • The URL of an XML file; valid protocol identifiers include http, https, ftp, and file. • An XML document object, such as one generated by the XmlParse function.
validator	Any of the following: • A string containing a DTD or Schema. • The name of a DTD or Schema file. • The URL of a DTD or Schema file; valid protocol identifiers include http, https, ftp, and file.

Usage

If you specify a relative URL or filename in a parameter, ColdFusion uses the directory (or, for URLs, the virtual directory) that contains the current ColdFusion page as the path root.

The *validator* parameter specifies a DTD or Schema to use to validate the document. If you omit the parameter, the XML document must contain one of the following:

- A `!DOCTYPE` tag to specify the DTD or its location
- An `xsi:schemaLocation` or `xsi:noNamespaceSchemaLocation` tag to specify the Schema location

If you use a *validator* parameter and the XML document specifies a DTD or Schema, the `XmlValidate` function uses the *validator* parameter, and ignores the specification in the XML document.

If you do not use a *validator* parameter, and the XML document does not specify a DTD or Schema, the function returns a structure with an error message in the Errors field.

This function attempts to process the complete XML document, and reports all errors found during the processing. As a result, the returned structure can have a combination of Warning, Error, and FatalError fields, and each field can contain multiple error messages.

Example

The following example has three parts: an XML file, an XSD Schema file, and a CFML page that parses the XML file and uses the Schema for validation. The CFML file displays the value of the returned structure's Status field and displays the returned structure. To show the results of invalid XML, modify the custorder.xml file.

Note: *The Schema used in the following example represents the same XML structure as the DTD used in the* `XmlParse` *example.*

The custorder.xml file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://localhost:8500/something.xsd" id="4323251" >
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <name>Deluxe Carpenter&apos;s Hammer</name>
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
        <item id="54">
            <name>36&quot; Plastic Rake</name>
            <quantity>2</quantity>
            <unitprice>6.95</unitprice>
        </item>
        <item id="68">
            <name>Standard paint thinner</name>
            <quantity>3</quantity>
            <unitprice>8.95</unitprice>
        </item>
    </items>
</order>
```

The custorder.xsd file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
    <xs:element name="customer">
```

```
        <xs:complexType>
            <xs:attribute name="firstname" type="xs:string" use="required"/>
            <xs:attribute name="lastname" type="xs:string" use="required"/>
            <xs:attribute name="accountNum" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="quantity" type="xs:string"/>
    <xs:element name="unitprice" type="xs:string"/>
    <xs:element name="item">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="name"/>
                <xs:element ref="quantity"/>
                <xs:element ref="unitprice"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:integer" use="required">
            </xs:attribute>
        </xs:complexType>
    </xs:element>
    <xs:element name="items">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="item" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="order">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="customer"/>
                <xs:element ref="items"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

The CFML file is as follows. It uses a filename for the XML file and a URL for the Schema. The XML and URL paths must be absolute.

```
<cfset
myResults=XMLValidate("C:\CFusionMX7\wwwroot\examples\custorder.xml",
"http://localhost:8500/examples/custorder.xsd")>
<cfoutput>
Did custorder.xml validate against custorder.xsd? #results.status#<br><br>
</cfoutput>
Dump of myResults structure returned by XMLValidate<br>
<cfdump var="#myResults#">
```

Year

Description

From a date/time object, gets the year value.

Returns

The year value of *date*.

Category

Date and time functions

Function syntax

`Year(date)`

See also

`DatePart`, `IsLeapYear`

Parameters

Parameter	Description
`date`	A date/time object in the range 100 AD–9999 AD.

Usage

When passing a date as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date.

Example

```
<h3>Year Example</h3>
<cfif IsDefined("FORM.year")>
    More information about your date:
    <cfset yourDate = CreateDate(FORM.year,FORM.month,FORM.day)>
    <cfoutput>
    <p>Your date, #DateFormat(yourDate)#.
    <br>It is #DayOfWeekAsString(DayOfWeek(yourDate))#,
    day #DayOfWeek(yourDate)# in the week.
    <br>This is day #Day(yourDate)#
    in the month of #MonthAsString(Month(yourDate))#,
    which has #DaysInMonth(yourDate)# days.
    <br>We are in week #Week(yourDate)# of #Year(YourDate)#
    (day #DayOfYear(yourDate)# of #DaysInYear(yourDate)#). <br>
    <cfif IsLeapYear(Year(yourDate))>
        This is a leap year
        <cfelse>This is not a leap year
    </cfif>
    </cfoutput>
</cfif>
```

YesNoFormat

Description

Evaluates a number or Boolean value.

Returns

Yes, for a nonzero value; No for zero, `false`, and `no` Boolean values, and an empty string (`""`).

Category

Decision functions, Display and formatting functions

Function syntax

`YesNoFormat(value)`

See also

`IsBinary`, `IsNumeric`

Parameters

Parameter	Description
`value`	A number or Boolean value

Example

```
<h3>YesNoFormat Example</h3>
<p>The YesNoFormat function returns non-zero values as "Yes"; zero, false and no Boolean
values, and empty strings ("") as "No".

<cfoutput>
<ul>
    <li>YesNoFormat(1):#YesNoFormat(1)#
    <li>YesNoFormat(0):#YesNoFormat(0)#
    <li>YesNoFormat("1123"):#YesNoFormat("1123")#
    <li>YesNoFormat("No"):#YesNoFormat("No")#
    <li>YesNoFormat(True):#YesNoFormat(True)#
</ul>
</cfoutput>
```

Chapter 5: AJAX JavaScript Functions

You can use the JavaScript functions listed below on pages that use ColdFusion AJAX features.

Contents

Function summary

The following table briefly describes the JavaScript functions that you can use in ColdFusion pages that use AJAX features:

Function	Description
ColdFusion.Ajax.submitForm	Submits form data without refreshing the entire page when the results are returned.
ColdFusion.getElementValue	Gets the value of an attribute of a bindable ColdFusion control.
ColdFusion.Grid.getGridObject	Gets the underlying Ext JS - JavaScript Library object for the specified HTML format `cfgrid` control.
ColdFusion.Grid.refresh	Manually refreshes a displayed grid.
ColdFusion.Grid.sort	Sorts the specified HTML format grid.
ColdFusion.Layout.collapseArea	Collapses an area of a border layout (`cflayout` tag with a `type` attribute of `border`).
ColdFusion.Layout.createTab	Creates a new tab in an existing tabbed layout (`cflayout` tag with a `type` attribute of `tab`).
ColdFusion.Layout.disableTab	Disables the specified tab so it cannot be selected.
ColdFusion.Layout.enableTab	Enables the specified tab so users can select it and display the area contents.
ColdFusion.Layout.expandArea	Expands a collapsed area of a border layout.
ColdFusion.Layout.getBorderLayout	Gets the underlying Ext JS - JavaScript Library object for the specified border type `cflayout` control.
ColdFusion.Layout.getTabLayout	Gets the underlying Ext JS - JavaScript Library object for the specified tab type `cflayout` control.
ColdFusion.Layout.hideArea	Hides a bordered layout area.
ColdFusion.Layout.hideTab	Hides a tab.
ColdFusion.Layout.selectTab	Selects a tab and displays the layout area contents.
ColdFusion.Layout.showArea	Shows an area of a border layout that was hidden using the `inithide` attribute or the `hideArea()` function.
ColdFusion.Layout.showTab	Shows a tab that was hidden using the `inithide` attribute or the `hideTab()` function.
ColdFusion.Log.debug	Displays a debug-level message in the log window.
ColdFusion.Log.dump	Displays information about a complex variable in the log window.
ColdFusion.Log.error	Displays an error-level message in the log window.
ColdFusion.Log.info	Displays an information-level message in the log window.
ColdFusion.navigate	Displays the output of a link URL in a specified `cfdiv`, `cflayoutarea`, `cfpod`, or `cfwindow` container.
ColdFusion.setGlobalErrorHandler	Replaces the global JavaScript error handler for displaying information about ColdFusion AJAX errors.
ColdFusion.Tree.getTreeObject	Gets the underlying Yahoo YUI Library object for the specified HTML format `cftree` control.
ColdFusion.Tree.refresh	Manually refreshes a displayed HTML format tree.
ColdFusion.Window.create	Creates a ColdFusion pop-up window. Equivalent to the `cfwindow` tag.

Function	Description
ColdFusion.Window.getWindowObject	Gets the underlying Ext JS - JavaScript Library object for the specified HTML format `cfwindow` control.
ColdFusion.Window.hide	Hides a window
ColdFusion.Window.onHide	Specifies a JavaScript function to run each time a specific window hides.
ColdFusion.Window.onShow	Specifies a JavaScript function to run each time a specific window shows.
ColdFusion.Window.show	Shows a hidden window.

ColdFusion.Ajax.submitForm

Description

Submits form data without refreshing the page when the results are returned.

Function syntax

`ColdFusion.Ajax.submitForm(formId, URL[, callbackhandler, errorhandler, httpMethod, asynch])`

See also

`cfajaxproxy`, `ColdFusion.navigate`, "Using the ColdFusion.Ajax.submitForm function" on page 630 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`formId`	The `ID` or `name` attribute of the form.
`URL`	The URL to which to submit the form.
`callbackhandler`	The JavaScript function to handle a normal response. The function must take a single argument, that contains the response body. This method is used only if the form submission is asynchronous.
`errorhandler`	The JavaScript function to handle an HTTP error response. The function must take two arguments: the HTTP status code, and the error message. This method is used only if the form submission is asynchronous.
`httpMethod`	The HTTP method to use for the submission, must be one of the following: • `GET` • `POST` (the default)
`asynch`	A Boolean value specifying whether to submit the form asynchronously. The default value is `true`.

Returns

If the `asynch` argument is `false`, returns the response body. Otherwise, the function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you must use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function.

Note: *This function does not submit the contents of file fields.*

Example

See "Using the ColdFusion.Ajax.submitForm function" on page 630 in the *ColdFusion Developer's Guide*.

ColdFusion.getElementValue

Description

Gets the value of an attribute of a bindable ColdFusion control.

Function syntax

`ColdFusion.getElementValue(elementId [, formId, attributeName])`

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`elementId`	The `ID` or `name` attribute of the control.
`formId`	The `ID` attribute of the form that contains the control. Omit this attribute if the element ID is unique on the page. If you omit this attribute and the element ID is not unique, the function uses the first element on the page with the specified ID.
`attributeName`	The control attribute to get; by default, the `value` attribute, or, for `cfselect`, the value of the selected element in the control. For `cfgrid` controls, you must use this attribute and specify the name of the column whose value you are getting; the function returns the entry in the currently selected row. For `cftree` controls, you must use this attribute and specify `PATH` or `NODE`. The function returns the item path or node value of the currently selected tree item.

Returns

The value of the specified attribute.

Usage

You can bind to, and get the attribute values of, the following HTML-format controls:

- `cfgrid`
- `cfinput` controls with checkbox, datefield, file, hidden, radio, or text types
- `cfselect`
- `cftextarea`
- `cftree`

ColdFusion.Grid.getGridObject

Description

Gets the underlying Ext (Ext JS JavaScript library) object for the specified HTML format grid.

Function syntax

`ColdFusion.Grid.getGridObject(name)`

See also

`cfgrid`, `ColdFusion.Grid.refresh`, `ColdFusion.Grid.sort`, Ext JS - JavaScript Library Documentation, "Using HTML format grids" on page 631 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The value of the `name` attribute of the `cfgrid` tag for which you want the object.

Returns

If the grid is editable, an object of type `Ext.grid.EditableGrid`; otherwise, an object of type `Ext.grid.Grid`.

Usage

Use this function to get the Ext toolkit (`Ext.grid`) object that underlies the ColdFusion HTML format `cfgrid` control. You can then use the raw object to modify the displayed grid. For documentation on the objects and how to manage them, see the Ext documentation.

ColdFusion.Grid.refresh

Description

Manually refreshes a displayed grid.

Function syntax

`ColdFusion.Grid.refresh(name [, preservePage])`

See also

`cfgrid`, `ColdFusion.Grid.getGridObject`, `ColdFusion.Grid.sort`, "Using HTML format grids" on page 631 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The value of the `name` attribute of the `cfgrid` tag to refresh.
`preservePage`	A Boolean value specifying whether to redisplay the current page of data (`true`), or display the first page of data (`false`, the default). This attribute applies only if the grid data requires multiple grid pages to display.

Returns

This function does not return a value.

Usage

This function is useful to refresh a grid when an event occurs that changes the underlying data but does not normally trigger a grid update.

Example

The following code snippet comes from an example that lets users delete rows from a grid. When the user selects a grid row and clicks the delete button, the AJAX proxy calls a `mycfc.deleteRow` function to delete the row from the database. When the function returns successfully, the proxy calls `ColdFusion.Grid.refresh` to update the grid and remove the row.

```
<cfajaxproxy bind="cfc:mycfc.deleteRow({deletebutton@click},{mygrid.id@none}"
    onSuccess="ColdFusion.Grid.refresh('mygrid', true)">
```

ColdFusion.Grid.sort

Description

Sorts the specified HTML format grid.

Function syntax

`ColdFusion.Grid.sort(name [, columnName, direction])`

See also

`cfgrid`, `ColdFusion.Grid.getGridObject`, `ColdFusion.Grid.refresh`, "Using HTML format grids" on page 631 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The value of the `name` attribute of the `cfgrid` tag to sort.
`columnName`	The name of the column that determines the sort order.
`direction`	The sort direction. Must be one for these values: • `ASC` (default) • `DESC`

Returns

This function does not return a value.

Usage

This function sorts the data displayed by the grid by using a case-insensitive sort for string data, or a numeric sort for numeric data. It uses the specified column contents to determine the displayed grid order. When a grid has a remote data source, the bound CFC function that provides the data gets the column name and sort direction in the `cfgridsortcolumn` and `cfgridsortdirection` bind attributes. The CFC function must use these values and perform the sort appropriately.

ColdFusion.Layout.collapseArea

Description

Collapses an area of a border layout.

Function syntax

`ColdFusion.Layout.collapseArea(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.expandArea`, `ColdFusion.Layout.getTabLayout`, `ColdFusion.Layout.showArea`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`layout`	The `name` attribute of the border layout that contains the area to collapse.
`layoutArea`	The position in the layout of the area to collapse. Must be one of the following: `bottom`, `left`, `right`, or `top`.

Returns

This function does not return a value.

Usage

This function has no effect if the area is already collapsed.

Example

The following code snippet collapses the left area of the layout border layout when the user clicks the button.

```
<cfinput name="collapse2" width="100" value="Collapse Area 2" type="button"
        onClick="ColdFusion.Layout.collapseArea('thelayout', 'left');">
```

ColdFusion.Layout.createTab

Description

Creates a new tab and layout area in a ColdFusion tabbed layout.

Function syntax

`ColdFusion.Layout.createTab(layout, layoutArea, Title, URL [, configObject])`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.disableTab`, `ColdFusion.Layout.enableTab`, `ColdFusion.Layout.hideTab`, `ColdFusion.Layout.selectTab`, `ColdFusion.Layout.showTab`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`layout`	The `name` attribute of the tabbed layout in which to add the tab
`layoutArea`	The name to assign to the layout area that is created for the new tab. Must be unique on the page.
`title`	The text to display on the tab. You can use HTML mark-up to control the title appearance.
`URL`	The URL from which to get the layout area contents. This attribute can use URL parameters to pass data to the page. ColdFusion uses standard page path resolution rules to locate the page.
`configObject`	An object containing window configuration parameters. For details, see "Usage".

Returns

This function does not return a value.

Usage

This function dynamically creates tabs in a tabbed layout; it is equivalent to putting a `cflayoutarea` tag inside a `cflayout` tag with a `type` attribute of `tag`. The *configuration* parameter defines tab characteristics; it can have any or all of the following entries:

Entry	Default	Description
align	The `cflayout` tag `align` attribute value	Specifies how to align child controls within the layout area. The following values are valid: • `center` • `justify` • `left` • `right`
callbackhandler		A function that will be called when the layout tab body has loaded. This function must not take any arguments.
closable	false	A Boolean value specifying whether the user can close the window. If `true`, the tab has an X close icon.
disabled	false	A Boolean value specifying whether the tab is disabled, that is, whether user can select the tab to display its contents. Disabled tabs are greyed out. Ignored if there is a `true selected` entry.

Entry	Default	Description
errorhandler		A function that will be called if an error occurs in loading the tab body. This function must take two arguments: • The HTTP status code, or -1 if the error is not a HTTP error • An error message
inithide	false	A Boolean value specifying whether the tab is initially hidden. To show an initially hidden tab, use the `ColdFusion.Layout.showTab` function.
overflow	auto	Specifies how to display child content whose size would cause the tab layout area to overflow the window boundaries. The following values are valid: • `auto`: Show scroll bars when necessary. • `hidden`: Do not allow access to overflowing content. • `scroll`: Always show horizontal and vertical scroll bars, even if they are not needed. • `visible`: Content can display outside the bounds of the layout area. **Note:** In Internet Explorer, layout areas with the visible setting expand to fit the size of the contents, rather than having the contents extend beyond the layout area.
selected	false	A Boolean value specifying whether this tab is initially selected so that its contents appears in the layout.
style		A CSS style specification that controls the appearance of the layout area.

Example

The following example creates a tabbed layout with one tab. When you click the button it creates a second tab that is immediately visible and selected.

The main page looks as follows:

```
<html>
<head>
</head>
<body>
<cfform name="layouts">
    <cfinput type="button" name="CreateTab"
        onClick="ColdFusion.Layout.createTab('tabLayout','tab2',
            'Tab 2','_tabUrl.cfm',{inithide:false,selected:true})"
        value="Create Tab">
</cfform>

<cflayout type="tab" name="tabLayout">
    <cflayoutarea name="tab1" title="Tab 1" align="left">
        Default Tab
    </cflayoutarea>
</cflayout>
</body>
</html>
```

The _tabURL.cfm page looks as follows:

```
<h3>Tab 2</h3>
This is a simple tab
```

ColdFusion.Layout.disableTab

Description

Disables the specified tab so it cannot be selected.

Function syntax

`ColdFusion.Layout.disableTab(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.createTab`, `ColdFusion.Layout.enableTab`, `ColdFusion.Layout.hideTab`, `ColdFusion.Layout.selectTab`, `ColdFusion.Layout.showTab`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
layout	The `name` attribute of the tabbed layout that contains the area to disable.
layoutArea	The `name` attribute of the tab layout area to disable.

Returns

This function does not return a value.

Usage

This function has no effect on the currently selected tab. A disabled tab is greyed.

Example

The following example lets you enable and disable a tab by clicking a link.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>

<body>
<!--- The tabheight attribute sets the height of all tab content areas. --->
<cflayout type="tab" name="mainTab" tabheight="300px" style="width:400px">
    <cflayoutarea title="First Tab" name="tab1">
        <h2>The First Tab</h2>
        Here are the contents of the first tab.
    </cflayoutarea>

    <cflayoutarea title="Second Tab" name="tab2">
        <h2>The Second Tab</h2>
        This is the content of the second tab.
    </cflayoutarea>
</cflayout>

<p>
Use these links to test disabling/enabling via JavaScript.
Note that you cannot disable the currently selected tab.<br />
<a href="" onClick="ColdFusion.Layout.enableTab('mainTab','tab1');
    return false;">Click here to enable tab 1.</a><br />
<a href="" onClick="ColdFusion.Layout.disableTab('mainTab','tab1');
```

```
    return false;">Click here to disable tab 1.</a><br />
</p>
</body>
</html>
```

ColdFusion.Layout.enableTab

Description

Enables the specified tab so it can be selected.

Function syntax

`ColdFusion.Layout.enableTab(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.createTab`, `ColdFusion.Layout.disableTab`, `ColdFusion.Layout.hideTab`, `ColdFusion.Layout.selectTab`, `ColdFusion.Layout.showTab`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`layout`	The `name` attribute of the tabbed layout that contains the area to enable.
`layoutArea`	The `name` attribute of the tab layout area to enable.

Returns

This function does not return a value.

Example

See `ColdFusion.Layout.disableTab`

ColdFusion.Layout.expandArea

Description

Expands an area of a border layout.

Function syntax

`ColdFusion.Layout.expandArea(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.collapseArea`, `ColdFusion.Layout.getTabLayout`, `ColdFusion.Layout.showArea`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
layout	The `name` attribute of the border layout that contains the area to expand.
layoutArea	The position in the layout of the area to expand. Must be one of the following: `bottom`, `left`, `right`, or `top`.

Returns

This function does not return a value.

Usage

This function has no effect if the area is already expanded.

Example

The following code snippet expands the left area of the layout border layout when the user clicks the button.

```
<cfinput name="expand2" width="100" value="Expand Area 2" type="button"
        onClick="ColdFusion.Layout.expandArea('thelayout', 'left');">
```

ColdFusion.Layout.getBorderLayout

Description

Gets the underlying Ext (Ext JS JavaScript library) object for the specified bordered layout.

Function syntax

`ColdFusion.Layout.getBorderLayout(`*`name`*`)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.getTabLayout`, Ext JS - JavaScript Library Documentation, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
name	The value of the `name` attribute of the border type `cflayout` tag for which you want the object.

Returns

An object of type `Ext.BorderLayout`.

Usage

Use this function to get the Ext toolkit (`Ext.BorderLayout`) object that underlies the ColdFusion HTML format `cflayout` control. You can then use the raw object to modify the displayed layout. For documentation on the objects and how to manage them, see the Ext documentation.

ColdFusion.Layout.getTabLayout

Description

Gets the underlying Ext (Ext JS JavaScript library) object for the specified tabbed layout.

Function syntax

`ColdFusion.Layout.getTabLayout(`*`name`*`)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.getBorderLayout`, Ext JS - JavaScript Library Documentation, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The value of the `name` attribute of the border type `cflayout` tag for which you want the object.

Returns

An object of type `Ext.BorderLayout`.

Usage

Use this function to get the Ext toolkit (`Ext.BorderLayout`) object that underlies the ColdFusion HTML format `cflayout` control. You can then use the raw object to modify the displayed layout. For documentation on the objects and how to manage them, see the Ext documentation.

ColdFusion.Layout.hideArea

Description

Hides an area of a border layout.

Function syntax

`ColdFusion.Layout.hideArea(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.collapseArea`, `ColdFusion.Layout.expandArea`, `ColdFusion.Layout.showArea`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`layout`	The `name` attribute of the border layout that contains the area to hide.
`layoutArea`	The position in the layout of the area to hide. Must be one of the following: `bottom`, `left`, `right`, or `top`.

Returns

This function does not return a value.

Usage

This function has no effect if the area is already hidden.

Example

The following code snippet hides the left area of the layout border layout when the user clicks the button.

```
<cfinput name="hide2" width="100" value="Hide Area 2" type="button"
        onClick="ColdFusion.Layout.hideArea('thelayout', 'left');">
```

ColdFusion.Layout.hideTab

Description

Hides the specified tab and its layout area.

Function syntax

`ColdFusion.Layout.hideTab(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.createTab`, `ColdFusion.Layout.disableTab`, `ColdFusion.Layout.enableTab`, `ColdFusion.Layout.selectTab`, `ColdFusion.Layout.showTab`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`layout`	The `name` attribute of the tabbed layout that contains the area to hide.
`layoutArea`	The `name` attribute of the tab layout area to hide.

Returns

This function does not return a value.

Example

The following example creates a layout with two tabs. Click the buttons to show and hide the second tab.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>

<body>
<cflayout type="tab" name="tabLayout" tabheight="300px"
    style="width:400px">
    <cflayoutarea title="First Tab" name="tab1">
        <h2>The First Tab</h2>
        Here are the contents of the first tab.
    </cflayoutarea>

    <cflayoutarea title="Second Tab" name="tab2">
        <h2>The Second Tab</h2>
        This is the content of the second tab.
    </cflayoutarea>
</cflayout>
<br />

<cfform name="layouts">
    <cfinput type="button" name="ShowTab" value="Show Tab"
        onClick="ColdFusion.Layout.showTab('tabLayout','tab2')">
    <cfinput type="button" name="ShowTab" value="Hide Tab"
        onClick="ColdFusion.Layout.hideTab('tabLayout','tab2')">
</cfform>
</body>
</html>
```

ColdFusion.Layout.selectTab

Description

Selects the specified tab and displays its layout area.

Function syntax

`ColdFusion.Layout.selectTab(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.createTab`, `ColdFusion.Layout.disableTab`, `ColdFusion.Layout.enableTab`, `ColdFusion.Layout.hideTab`, `ColdFusion.Layout.showTab`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`layout`	The `name` attribute of the tabbed layout that contains the area to select.
`layoutArea`	The `name` attribute of the tab layout area to select.

Returns

This function does not return a value.

Usage

This function has no effect on a disabled tab.

Example

The following code lets you select each of the two tabs in a layout.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>

<body>
<cflayout type="tab" name="mainTab" tabheight="300px" style="width:400px">
    <cflayoutarea title="First Tab" name="tab1">
        <h2>The First Tab</h2>
        Here are the contents of the first tab.
    </cflayoutarea>

    <cflayoutarea title="Second Tab" name="tab2">
        <h2>The Second Tab</h2>
        This is the content of the second tab.
    </cflayoutarea>
</cflayout>

<p>
Use these links to test selecting tabs via JavaScript:<br />
<a href="" onClick="ColdFusion.Layout.selectTab('mainTab','tab1');
    return false;">Click here to select tab 1.</a><br />
<a href="" onClick="ColdFusion.Layout.selectTab('mainTab','tab2');
    return false;">Click here to select tab 2.</a><br />
</p>
```

```
</body>
</html>
```

ColdFusion.Layout.showArea

Description

Shows an area of a border layout that was hidden by using the `cflayoutarea` tag `inithide` attribute or the `ColdFusion.Layout.hideArea()` JavaScript function.

Function syntax

`ColdFusion.Layout.showArea(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.collapseArea`, `ColdFusion.Layout.expandArea`, `ColdFusion.Layout.getTabLayout`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`layout`	The `name` attribute of the border layout that contains the area to show.
`layoutArea`	The position in the layout of the area to show. Must be one of the following: `bottom`, `left`, `right`, or `top`.

Returns

This function does not return a value.

Usage

This function does not show an area that a user closed by clicking the `x` icon on the title bar. Other areas move if needed to accommodate the area.

This function has no effect if the area is already visible.

Example

The following code snippet shows the left area of the layout border layout when the user clicks the button.

```
<cfinput name="show2" width="100" value="Show Area 2" type="button"
        onClick="ColdFusion.Layout.showArea('thelayout', 'left');">
```

ColdFusion.Layout.showTab

Description

Shows a tab that was hidden by using the `inithide` attribute of the `cflayoutarea` tag or the `hideTab()` JavaScript function.

Function syntax

`ColdFusion.Layout.showTab(layout, layoutArea)`

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.createTab`, `ColdFusion.Layout.disableTab`, `ColdFusion.Layout.enableTab`, `ColdFusion.Layout.hideTab`, `ColdFusion.Layout.selectTab`, "Using layouts" on page 617 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`layout`	The `name` attribute of the tabbed layout that contains the tab to show.
`layoutArea`	The `name` attribute of the tab layout area whose tab you want to show.

Returns

This function does not return a value.

Usage

This function shows only the tab of a layout area; it does not show the display area. To show the display area of a hidden tab, call this function, followed by `ColdFusion.Layout.selectTab`.

This function does not show a tab that a user closed by clicking the `x` icon on the tab.

Example

See `ColdFusion.Layout.hideTab`.

ColdFusion.Log.debug

Description

Displays a debug-level message in a log window.

Function syntax

`ColdFusion.Log.debug(message [, category])`

See also

`ColdFusion.Log.dump`, `ColdFusion.Log.error`, `ColdFusion.Log.info`, "Logging information" on page 672 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`message`	The text message to display in the log window. The log message can include HTML markup and JavaScript variables.
`category`	A category identifier that you can use in the logging window to filter the output. You can specify any arbitrary category in this function. The default value is `global`.

Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you must use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function.

The log window appears if you specify `a` URL parameter of the format `cfdebug` or `cfdebug="true"` in your page request and you select the Enable AJAX Debug Log Window option on the ColdFusion Administrator Debugging & Logging > Debug Output Settings page.

Example

```
ColdFusion.Log.debug("<b>Debug argument:</b><br>" + arg.A, "Pod A");
```

ColdFusion.Log.dump

Description

Displays a debug-level message in the log window that shows a `cfdump`-like representation of a complex JavaScript object. The log window does not have a separate dump level.

Function syntax

`ColdFusion.Log.dump(object [, category])`

See also

`ColdFusion.Log.debug`, `ColdFusion.Log.error`, `ColdFusion.Log.info`, "Logging information" on page 672 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`object`	The variable whose contents you want to display. You cannot specify additional contents, such as a text message, when you dump a complex object. To provide additional information, also use the `ColdFusion.Log.debug` function.
`category`	A category identifier that you can use in the logging window to filter the output. You can specify any arbitrary category in this function. The default value is `global`.

Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you must use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function.

The log window appears if you specify a URL parameter of the format `cfdebug` or `cfdebug="true"` in your page request and you select the Enable AJAX Debug Log Window option on the ColdFusion Administrator Debugging & Logging > Debug Output Settings page.

Example

```
ColdFusion.Log.dump(objArg, "Pod A");
```

ColdFusion.Log.error

Description

Displays an error-level message in a log window.

Function syntax

`ColdFusion.Log.error(message [, category])`

See also

`ColdFusion.Log.debug`, `ColdFusion.Log.dump`, `ColdFusion.Log.info`, "Logging information" on page 672 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`message`	The text message to display in the log window. The log message can include HTML markup and JavaScript variables.
`category`	A category identifier that you can use in the logging window to filter the output. You can specify any arbitrary category in this function. The default value is `global`.

Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you must use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function.

The log window appears if you specify `a` URL parameter of the format `cfdebug` or `cfdebug="true"` in your page request and you select the Enable AJAX Debug Log Window option on the ColdFusion Administrator Debugging & Logging > Debug Output Settings page.

Example

```
ColdFusion.Log.error("<b>Invalid value:</b><br>" + arg.A, "Pod A");
```

ColdFusion.Log.info

Description

Displays an information-level message in a log window.

Function syntax

`ColdFusion.Log.info(message [, category])`

See also

`ColdFusion.Log.debug`, `ColdFusion.Log.dump`, `ColdFusion.Log.error`, "Logging information" on page 672 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`message`	The text message to display in the log window. The log message can include HTML markup and JavaScript variables.
`category`	A category identifier that you can use in the logging window to filter the output. You can specify any arbitrary category in this function. The default value is `global`.

Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you must use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function.

The log window appears if you specify `a` URL parameter of the format `cfdebug` or `cfdebug="true"` in your page request and you select the Enable AJAX Debug Log Window option on the ColdFusion Administrator Debugging & Logging > Debug Output Settings page.

Example

```
ColdFusion.Log.info("<b>arg.A is:</b><br>" + arg.A, "Window Z");
```

ColdFusion.navigate

Description

Displays the output of a link target in an AJAX `cfdiv`, `cflayoutarea`, `cfpod`, or `cfwindow` container. When the browser follows a link that is populated by this function, the link does not replace the current page. Instead, it populates the control specified by the `container` attribute.

Function syntax

`ColdFusion.navigate(URL [, container, callbackhandler, errorhandler, httpMethod, formId])`

See also

`AjaxLink`, `cfajaximport`, `ColdFusion.Ajax.submitForm`, "Controlling container contents" on page 623 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`URL`	The URL of the link.
`container`	The `name` attribute value of the control in which to display the link output. The control must be a container control such as `cfdiv`, `cflayoutarea`, `cfpod`, or `cfwindow`. If you omit this argument, the link is treated as a normal URL and the entire page is refreshed.
`callbackhandler`	The name of a JavaScript function to call after the target has been displayed.
`errorhandler`	The name of a JavaScript function to call if an error occurs when this function executes. The function can take two parameters: an HTTP error code, and an error message.
`formId`	The `ID` or `name` attribute of a form to submit to the URL.
`httpMethod`	The HTTP method to use when navigating to the URL: • `GET` (the default) • `POST`

Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you must use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function.

The `callbackhandler` parameter can be useful for changing the display after the contents has been displayed. For example, before you make the `ColdFusion.navigate` call you might make a pod's title bar italic to indicate loading; you could then use the `callbackhandler` function to switch it back to normal or make it bold once navigate completes. Similarly, you could use a `callbackhandler` to update the page number in a book reader.

The `FormID` attribute lets you specify a form to submit to the specified URL. You can use the `ColdFusion.Navigate` function with this attribute to submit form data asynchronously from outside the form, for example, when the user clicks a menu item, and to direct the returned results to a specific container control.

Example

When the user clicks the link in window 1, the `ColdFusion.navigate` function replaces the text in window 2 with the contents of windowsrc.cfm, and then calls the myCallback callback handler, which changes the innerHTML of the callback div region.

The main application page looks as follows:

```
<html>
<head>
<!--- The Callback handler puts text in the window.cfm callback div. --->
<script language="javascript">
    var mycallBack = function(){
        document.getElementById("callback").innerHTML = "<br><br><b>This is printed by the
callback handler.</b>";
    }

<!--- The error handler pops an alert with the error code and message. --->
    var myerrorhandler = function(errorCode,errorMessage){
        alert("[In Error Handler]" + "\n\n" + "Error Code: " + errorCode + "\n\n" + "Error
Message: " + errorMessage);
    }
</script>
</head>

<body>
<cfwindow name="w1" title="CF Window 1" initShow=true
    x=10 y=10 width="200">
    This is a cfwindow control.<br><br>
    <a href="javascript:ColdFusion.navigate('windowsource.cfm','w2',
        mycallBack,myerrorhandler);">Click</a> to navigate Window 2</a>
</cfwindow>

<cfwindow name="w2" title="CF Window 2" initShow=true
    x=250 y=10 width="200">
    This is a second cfwindow control.
</cfwindow>
</body>
</html>
```

The windowsource.cfm page looks as follows:

```
This is markup from "windowsource.cfm"
<!-- The callback handler puts its output in the following div block. -->
<div id="callback"></div>
```

ColdFusion.setGlobalErrorHandler

Description

Specifies a function that gets called, in place of the ColdFusion AJAX default error handler, if an error occurs when using a ColdFusion AJAX feature.

Function syntax

`ColdFusion.setGlobalErrorHandler(`*`functionName`*`)`

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`functionName`	The name of the JavaScript function to execute when there is an error in ColdFusion AJAX code, such as a binding error. This function must take a single argument, the error message string.

Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you must use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function.

The global error handler displays information about errors that occur in ColdFusion AJAX features. The default global error handler displays an alert with the error message. You can use this function to create a custom global error handler, for example, to display a custom error window with additional information about your application.

ColdFusion.Tree.getTreeObject

Description

Gets the underlying object for the specified HTML format tree.

Function syntax

`ColdFusion.Tree.getTreeObject(name)`

See also

`cftree`, `cfajaximport`, `ColdFusion.Tree.refresh`, "Using HTML format trees" on page 636 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The value of the `name` attribute of the `cftree` tag for which you want the object.

Returns

An object of type `YAHOO.widget.TreeView`.

Usage

Use this function to get the Yahoo User Interface Library `YAHOO.widget.TreeView` object that underlies the HTML format `cftree` control. You can then use the raw object to modify the displayed tree. For documentation on the objects and how to manage them, see the Yahoo toolkit documentation.

ColdFusion.Tree.refresh

Description

Refreshes an HTML format tree and updates it with the latest values of all items.

Function syntax

`ColdFusion.Tree.refresh(name)`

See also

`cftree`, `cfajaximport`, `ColdFusion.Tree.getTreeObject`, "Using HTML format trees" on page 636 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The value of the `name` attribute of the `cftree` tag for which you want the object.

Returns

An object of type YAHOO.widget.TreeView.

Usage

Use this function to manually update the tree. If you populate the tree by using a bind expression, the `refresh` call causes the bind expression to be reevaluated and repopulates the tree root nodes. You should use this function any time you must get the latest data from the server independent of an event that triggers the `cftree` bind expression, for example, to you might use this function to periodically refresh a file/folder tree to represent the current status of the server.

ColdFusion.Window.create

Description

Creates a ColdFusion pop-up window. This function is equivalent to the `cfwindow` tag.

Function syntax

`ColdFusion.Window.create(name, title, URL [, configuration])`

See also

`cfwindow`, `ColdFusion.Window.getWindowObject`, `ColdFusion.Window.hide`, `ColdFusion.Window.onHide`, `ColdFusion.Window.onShow`, `ColdFusion.Window.show`, `ColdFusion.Tree.getTreeObject`, "Using pop-up windows" on page 620 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The name of the window. This attribute is required to interact with the window, including to dynamically show or hide it. If a window with the specified name already exists, the function will show that window, and will ignore the remaining parameters; otherwise, the name must be unique on the page.
`title`	The text to display on the window title bar. You can use HTML mark-up to control the title appearance.
`URL`	The URL from which to get the window body contents. This attribute can use URL parameters to pass data to the page. ColdFusion uses standard page path resolution rules to locate the page. **Note**: If the page specified in this attribute contains tags that use ColdFusion AJAX features, such as the `cfform`, `cfgrid`, and `cfpod` tags, you must identify the tags in a `cfajaximport` tag on the page that includes this function. For more information, see `cfajaximport`.
`configuration`	An object containing window configuration parameters. For details, see "Usage".

Returns

This function does not return a value.

Usage

This function is equivalent to the `cfwindow` tag.

If you do not also use a `cfwindow` tag on a page that calls this function, you must specify a `cfajaximport` tag on the page and specify `cfwindow` in the `tags` attribute. Doing so ensures that the page includes the necessary JavaScript to create the window. For example, use the following line if you do not have to import the JavaScript for any other ColdFusion AJAX features.:

```
<cfajaximport tags="cfwindow">
```

The *configuration* parameter defines the window characteristics; it can have any or all of the following entries:

Entry	Default	Description
callbackhandler		A function that is called when the window body loads. This function must not take any arguments.
center	false	A Boolean value that specifies whether to center the window over the browser window. • If `true`, ColdFusion ignores the `x` and `y` attribute values. • If `false`, and you do not specify `x` and `y` attributes, ColdFusion centers the window.

Entry	Default	Description
closable	true	A Boolean value that specifies whether the user can close the window. If `true`, the window has an X close icon.
draggable	true	A Boolean value that specifies whether the user can drag the window. To drag the window, click the mouse on the title bar and hold the button down while dragging. If the window does not have a title, users cannot drag it.
errorhandler		A function that is called if an error occurs in loading the window body. This function must take two arguments: • The HTTP status code, or -1 if the error is not a HTTP error • An error message
height	300	Height of the window in pixels. If you specify a value greater than the available space, the window occupies the available space and the resize handles do not appear.
initshow	false	A Boolean value that specifies whether to display the window when the containing page first displays. If this value is `false`, use the `ColdFusion.Window.show` JavaScript function to display the window.
minheight	0	The minimum height, in pixels, to which users can resize the window.
minwidth	0	The minimum width, in pixels, to which users can resize the window.
modal	false	A Boolean value that specifies whether the window is modal, that is, whether the user can interact with the main window while this window is displaying. If `true`, the user *cannot* interact with the main window.
resizable	true	A Boolean value that specifies whether the user can resize the window.
width	500	Width of the window in pixels. If you specify a value greater than the available space, the window occupies the available space and the resize handles do not appear.
x		The X (horizontal) coordinate of the upper-left corner of the window, relative to the browser window. ColdFusion ignores this attribute if the `center` attribute value is `true`, and if you do not set the `y` attribute value.
y		The Y (vertical) coordinate of the upper-left corner of the window, relative to the browser window. ColdFusion ignores this attribute if the `center` attribute value is `true`, and if you do not set the `x` attribute value.

Note: *Entry names in the configuration object must be all-lowercase.*

Example

The following minimal CFML application creates a window and gets the window contents from the hello1.cfm file.

```
<cfajaximport tags="cfwindow">

<cfform name="test">
    <cfinput type="button" name="x" value="Create Window"
        onClick="ColdFusion.Window.create('Window1', 'This is a CF window',
        'http://localhost:8500/My_stuff/AjaxUI/Book/hello1.cfm',
        {x:100,y:100,height:300,width:400,modal:false,closable:false,
        draggable:true,resizable:true,center:true,initshow:true,
        minheight:200,minwidth:200 })">
</cfform>
```

The hello1.cfm file can be as simple as the following line:

```
Hello from hello1.cfm
```

ColdFusion.Window.getWindowObject

Description

Gets the underlying object for the specified window.

Function syntax

`ColdFusion.Window.getWindowObject(name)`

See also

`cfwindow`, `ColdFusion.Window.create`, `ColdFusion.Window.hide`, `ColdFusion.Window.onHide`, `ColdFusion.Window.onShow`, `ColdFusion.Window.show`, "Using pop-up windows" on page 620 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The value of the `name` attribute of the `cfwindow` tag for which you want the object.

Returns

An object of type `Ext.BasicDialog`.

Usage

Use this function to get the Ext JavaScript Library `Ext.BasicDialog` object that underlies the HTML format `cfwindow` control. You can then use the raw object to modify the displayed window. For documentation on the objects and how to manage them, see the Ext JavaScript library documentation.

ColdFusion.Window.hide

Description

Hides a window that is currently displayed.

Function syntax

`ColdFusion.window.hide(name)`

See also

`cfwindow`, `ColdFusion.Window.create`, `ColdFusion.Window.getWindowObject`, `ColdFusion.Window.onHide`, `ColdFusion.Window.onShow`, `ColdFusion.Window.show`, "Using pop-up windows" on page 620 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`name`	The `name attribute` of the window to hide.

Returns

This function does not return a value.

Usage

This tag has no effect if the window is already hidden.

Example

The following code lets you show and hide a window by clicking buttons:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>

<cfwindow name="testWindow" initshow=true title="test window" closable=true>
    Window contents
</cfwindow>

<cfform>
    <cfinput name="hidebutton" type="button" value="Hide Window"
        onclick="javascript:ColdFusion.Window.hide('testWindow');"/>
    <cfinput name="showbutton" type="button" value="Show Window"
        onclick="javascript:ColdFusion.Window.show('testWindow');"/>
</cfform>
</body>
</html>
```

ColdFusion.Window.onHide

Description

Specifies a function to run each time a specific window hides.

Function syntax

`ColdFusion.Window.onHide(windowName, handler)`

See also

`cfwindow`, `ColdFusion.Window.create`, `ColdFusion.Window.getWindowObject`, `ColdFusion.Window.hide`, `ColdFusion.Window.onShow`, `ColdFusion.Window.show`, "Using pop-up windows" on page 620 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`windowName`	The name of the window. The handler function runs whenever this window hides.
`handler`	The JavaScript function to run when the window hides.

Returns

This function does not return a value.

Usage

The function specified in the *`handler`* parameter can optionally take one parameter, which contains the window name.

Example

The following example uses the `ColdFusion.Window.onHide` function to display an alert with information about the window when you click a button that hides the window:

```
<head>
    <script language="javascript">
        function onhide(name) {
            alert("window hidden = " + name);
        }

        function test() {
            ColdFusion.Window.onHide("testWindow", onhide);
            ColdFusion.Window.hide("testWindow");
        }
    </script>
</head>
<body>

<cfwindow name="testWindow" initshow=true title="test window"
closable=true>
    Window contents
</cfwindow>

<cfform>
    <cfinput name="button" value="Hide Window" onclick="javascript:test()" type="button"/>
```

```
</cfform>
</body>
</html>
```

ColdFusion.Window.onShow

Description

Specifies a function to run each time a specific window shows, including when you create a window and specify an `initShow` attribute or configuration entry value of `true`.

Function syntax

`ColdFusion.Window.onShow(windowName, handler)`

See also

`cfwindow`, `ColdFusion.Window.create`, `ColdFusion.Window.getWindowObject`, `ColdFusion.Window.hide`, `ColdFusion.Window.onHide`, `ColdFusion.Window.show`, "Using pop-up windows" on page 620 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
`windowName`	The name of the window. The handler function runs whenever this window shows.
`handler`	The JavaScript function to run when the window shows.

Returns

This function does not return a value.

Usage

The function specified in the *`handler`* parameter can optionally take one parameter, which contains the window name.

One use for this function is to fetch window data only when the window shows. You could use a `cfajaxproxy` tag to create a JavaScript proxy for a CFC function that provides the data, and then a `ColdFusion.Window.onShow` function to specify a function that calls the proxy function and updates the window contents with the new data.

Example

The following example uses the ColdFusion.Window.onShow function to display an alert with information about the window when you click a button that shows the window:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <script language="javascript">
        function onshow(name) {
            alert("window shown = " + name);
        }

        function test() {
            ColdFusion.Window.onShow("testWindow", onshow);
            ColdFusion.Window.show("testWindow");
        }
    </script>
</head>
<body>

<cfwindow name="testWindow" initshow=false title="test window"
```

```
    closable=true>
    Window contents
</cfwindow>

<cfform>
    <cfinput name="button" value="show Window" onclick="javascript:test()" type="button"/>
</cfform>
</body>
</html>
```

ColdFusion.Window.show

Description

Shows a window that is currently hidden.

Function syntax

`ColdFusion.Window.show(name)`

See also

`cfwindow`, `ColdFusion.Window.create`, `ColdFusion.Window.getWindowObject`, `ColdFusion.Window.hide`, `ColdFusion.Window.onHide`, `ColdFusion.Window.onShow`, "Using pop-up windows" on page 620 in the *ColdFusion Developer's Guide*

History

ColdFusion 8: Added this function

Parameters

Parameter	Description
name	The `name attribute` of the window to show.

Returns

This function does not return a value.

Usage

This function shows a window that you created with an `initShow` attribute or parameter value of `false`, or that you hid by calling the `ColdFusion.Window.hide` function. It does not show a window that a user closed by clicking the `X` icon on the title bar.

This tag has no effect if the window is already hidden.

Example

See the example at `ColdFusion.Window.hide`.

Chapter 6: ColdFusion Flash Form Style Reference

You can specify styles in ColdFusion forms tags when you display the form or form element in Flash format.

Contents

Note: *The column labeled* ***Inh*** *indicates whether a style is inherited by child controls, such as the form controls in a vbox.*

Styles valid for all controls

The following styles are valid for all ColdFusion Flash format form tags except for `cfformitem` tags with the following `type` attributes, which do not take `style` attributes:

- html
- space

These styles do not cause errors when used in all other tags. However, many styles do not have any effect when used in some tags.

Style	Inh	Description
backgroundAlpha	N	Alpha (transparency) level of the SWF file or image defined by `backgroundImage`. Valid values range from 0 (transparent) to 100 (opaque). The default value is 100.
backgroundColor	Y	Format: color; background color of the control. Has no effect if specified in a `cfform` control tag, which uses the `background-color` style to control the color. Also ignored by `cfinput` tags of `type` button, img, submit, radio, and checkbox, because they are completely filled with the button face or other graphics.
backgroundDisabledColor	Y	Format: color; background color of components when disabled. The default value is ##EFEEEF (light gray).
backgroundSize	N	Scales the image specified by `backgroundImage` to different percentage sizes. By default, the value is auto, which maintains the original size of the image. A value of 100% stretches the image to fit the entire screen. You must include the percent sign with the value.
barColor	Y	Format: color; color of the outer bar.
borderCapColor	Y	Format: color; outside left and outside right color for skins.
borderColor	Y	Format: color; black section of a three-dimensional border or the color section of a two-dimensional border.
borderSides	N	Bounding box sides. Only used when `borderStyle="solid"`. Space-delimited string containing the sides of the border to show. Order is not important. The default value is "left top right bottom".
borderStyle	Y	Bounding box style. The possible values are: • inset (default) • none • outset • solid
borderThickness	N	Bounding box thickness. Only used when borderStyle="solid". The default value is 1.
color	Y	Format: color; text color of a component's label.
cornerRadius	N	Radius of component corners. The default value is 0.
disabledColor	Y	Format: color; color of the component if it is disabled.
dropShadow	N	Format: Boolean; controls the visibility of the component's drop shadow. The default value is false. This style must be used with borderStyle="solid". For drop shadows to appear on containers, set backgroundColor or backgroundImage. Otherwise, since the default background of a container is transparent, the shadow appears behind the container.
errorColor	Y	Format: color; color of the error text.
fillColors	N	Format: color; colors used to tint the background of the control. Pass the same color for both values for "flat" looking control. The default value is ##E6EEEE,##FFFFFF.

Style	Inh	Description
fontFamily	Y	Comma-separated list of fonts to use, in descending order of desirability. You can use any font family name. If you specify a generic font name, it is converted to an appropriate device font. Flash can only use fonts that are installed on the client system.
fontSize	Y	Format: length; size of the text.
fontStyle	Y	Determines whether the text is italic. Recognized values are normal and italic. The default value is normal.
fontWeight	Y	Determines whether the text is bold. Recognized values are normal and bold. The default value is normal.
highlightColor	Y	Format: color; color of the control when it is in focus.
horizontalGap	N	Format: length; number of pixels between children in the horizontal direction.
leading	N	Additional vertical space between lines of text. The default value is no leading.
marginLeft	N	Format: length; number of pixels between the container's left border and its content area.
marginRight	N	Format: length; number of pixels between the container's right border and its content area.
scrollTrackColor	Y	Format: color; scroll track for a scroll bar. The default value is ##EFEEEF (light gray).
selectedFillColors	N	Format: colors; two colors used to tint the background of the control when in its selected state. Pass the same color for both values for "flat" looking control. The default value is undefined, which means the colors will be derived from themeColor.
textAlign	Y	Aligns text in a container. Recognized values are left, right, and center. The default value is right.
textDecoration	N	Determines whether the text is underlined or not. Recognized values are none and underline. The default value is none.
textIndent	Y	Format: length; offset of first line of text from the left side of the container. The default value is 0.
themeColor	Y	Format: color; background color of a component. The possible values are: • haloGreen • haloBlue • haloOrange • haloSilver
verticalGap	N	Format: length; number of pixels between children in the vertical direction.

Styles for cfform

The following styles apply to the `cfform` tag:

Style	Inh	Description
background-color		Format: color; background color of the form.
indicatorGap	Y	Format: length; number of pixels between the label and child components. The default value is 14.
labelWidth	Y	Format: length; width of the form labels. The default value is the length of the longest label in the form.
marginBottom	N	Format: length; number of pixels between the container's bottom border and its content area. The default value is 16.
marginTop	N	Format: length; number of pixels between the container's top border and its content area. The default value is 16.
verticalGap	N	Format: length; number of pixels between children in the vertical direction. The default value is 8.

Styles for cfformgroup with horizontal or vertical type atributes

The following styles apply to the `cfformgroup` tag with `type` attributes `horizontal` or `vertical`:

Style	Inh	Description
horizontalAlign	N	Horizontal alignment of children. Possible values are left, center, and right. The default value is left.
horizontalGap	N	Format: length; number of pixels between children in the horizontal direction. The default value is 6.
indicatorGap	Y	Format: length; number of pixels between the label and child components. The default value is 14.
labelWidth	Y	Format: length; width of the form labels. The default value is the length of the longest label in the form.
marginBottom	N	Format: length; number of pixels between the container`s bottom border and its content area. The default value is 0.
marginTop	N	Format: length; number of pixels between the container`s top border and its content area. The default value is 0.
verticalGap	N	Format: length; number of pixels between children in the vertical direction. The default value is 6.

Styles for box-style cfformgroup elements

The following styles apply to the `cfformgroup` tag with the following `type` attributes. Some types have additional attributes, which are listed in the following sections.

- hbox
- vbox
- hdividedbox
- vdividedbox
- panel
- tile
- page

Style	Inh	Description
horizontalAlign	N	Horizontal alignment of children in the container. The default value is left. Possible values are left, center, and right.
horizontalGap	N	Format: length; number of pixels between children in the horizontal direction. The default value is 8 (6 for a tile container).
marginBottom	N	Format: length; number of pixels between the container's bottom border and its content area. The default value is 0.
marginTop	N	Format: length; number of pixels between the container's top border and its content area. The default value is 0.
verticalAlign	N	Vertical alignment of children in the container. The default value is top. Possible values are top, middle, and bottom.
verticalGap	N	Format: length; number of pixels between children in the vertical direction. The default value is 8 (6 for a tile container).

Styles specific to cfformgroup with hdividedbox or vdividedbox type attributes

The following additional styles apply to the `cfformgroup` tag with `type="hdividedbox"`, or `type="vdividedbox"`:

Style	Inh	Description
dividerAffordance	N	Format: length; width (hdividedbox) or height (vdividedbox) in pixels of the area of the divider that the user can select with the mouse pointer. The default value is 6.
dividerColor	Y	Format: color; color of the dividers in their up state. The default value is ##AAAAAA.
dividerThickness	N	Format: length; thickness in pixels of the dividers. The default value is 4.

Styles specific to cfformgroup with panel type attribute

The following additional styles apply to the `cfformgroup` tag with `type="panel"`:

Style	Inh	Description
cornerRadius	N	Format: length; radius of corners of the window frame. The default value is 8.
dropShadow	N	Boolean value specifying whether the panel has a drop shadow. The default value is true.
footerColors	Y	Format: color; comma-delimited list of two colors used to draw the footer (ControlBar) background. The first color is the top color. The second color is the bottom color. The default value is ##F4F5F7, ##E1E5EB.
headerColors	Y	Format: color; comma-delimited list of two colors used to draw the header. The first color is the top color. The second color is the bottom color. The default value is ##E1E5EB, ##F4F5F7.
headerHeight	N	Format: length; height of the header. The default value is 28.
panelBorderStyle	N	Border style for the bottom two corners of the container. The top two corners are always round. Possible values are default, which configures the container to have square corners, and roundCorners, which defines rounded corners. To configure the top corners to be square, set cornerRadius to 0. The default value is default.
shadowDirection	N	Direction of drop shadow. Possible values are "left", "center", and "right". The default value is "center".
shadowDistance	N	Distance of drop shadow. Negative values move shadow above the panel. The default value is 2.

Styles for cfformgroup with accordion type attribute

The following styles apply to the `cfformgroup` tag with `type="accordion"`:

Style	Inh	Description
headerHeight	N	Format: length; height of the accordion container buttons, in pixels. The default value is 22.
marginBottom	N	Format: length; number of pixels between the container`s bottom border and its content area. The default value is -1.
marginTop	N	Format: length; number of pixels between the container`s top border and its content area. The default value is -1.
openDuration	N	Format: time; duration, in milliseconds, of the transition from one child panel to another. The default value is 250.
verticalGap	N	Format: length; number of pixels between children in the vertical direction. The default value is -1.

Styles for cfformgroup with tabnavigator type attribute

The following styles apply to the `cfformgroup` tag with the `type="tabnavigator"`:

Style	Inh	Description
horizontalAlign	N	Horizontal alignment of children. The default value is left. Possible values are left, center, and right. Because the preferred width of each tab in the tab navigator container is the size of the label text, you must use the `tabWidth` style to increase the width of the tab to a size larger than its preferred width to see different alignments.
horizontalGap	N	Format: length; number of pixels between children in the horizontal direction. The default value is 6.
tabHeight	N	Format: length; default tab height, in pixels. The default value is 22.
tabWidth	N	Format: length; width of the tabs, in pixels. If undefined, the default tab widths are automatically calculated from the label text. If the width of the container is smaller than the width of the label text, the labels are truncated. If a tab label is truncated, Flash displays a tooltip with the full label text when a user moves the mouse pointer over the tab. If you specify an explicit tab width, labels do not automatically shrink to fit if they do not fit in the available space.

Styles for cfformitem with hrule or vrule type attributes

The following styles apply to the `formitem` tag with `type="hrule"` or `type="vrule"`:

Style	Inh	Description
color	Y	Format: color; color of the line. according to the following rules: • If `strokeWidth` is 1, the color of the entire line. • If `strokeWidth` is 2 (default), the color of the top line. • If `strokeWidth` is greater than 2, the color of the top and left edges of the rectangle. The default value is ##C4CCCC.
shadowColor	Y	Format: color; shadow color of the line, as follows:. • If `strokeWidth` is 1, does nothing. • If `strokeWidth` is 2 (default), the color of the bottom line. • If `strokeWidth` is greater than 2, the color of the bottom and right edges of the rectangle. The default value is ##D4D0C8.
strokeWidth	Y	Thickness of the rule in pixels, as follows: • If `strokeWidth` is 1, the rule is a 1-pixel-wide line. • If `strokeWidth` is 2 (default), the rule is two adjacent 1-pixel-wide horizontal lines. • If `strokeWidth` is greater than 2, the rule is a hollow rectangle with 1-pixel-wide edges. The default value is 2.

Styles for cfinput with radio, checkbox, button, image, or submit type attributes

The following styles apply `cfinput` tags with the following `type` attribute values:

- button
- checkbox
- image
- radio
- submit

In some cases, a style applies only to the subset of these input types, as specified in the description.

Style	Inh	Description
borderThickness	N	Thickness of border "ring". A value of 0 means no border. Any value greater than 2 creates a glowing "ring" around the button. The default value is 3.
cornerRadius	N	Radius of corners. The default value is 5.
horizontalGap	N	Gap between the label and the image in an `img` input when labelPlacement = "left" or "right". The default value is 2.
repeatDelay	N	Format: time; number of milliseconds to wait after the first buttonDown event before repeating buttonDown events at the repeatInterval. The default value is 500.
repeatInterval	N	Format: time; number of milliseconds between buttonDown events if you press and hold a button. The default value is 35.
symbolBackgroundColor	Y	Format: color; background color of check boxes and radio buttons. The default value is ##FFFFFF (white).
symbolBackgroundDisabledColor	Y	Format: color; background color of check boxes and radio buttons when disabled. The default value is ##EFEEEF (light gray).
symbolBackgroundPressedColor	Y	Format: color; background color of check boxes and radio buttons when pressed. The default value is ##FFFFFF (white).
symbolColor	Y	Format: color; the check mark of a check box or the dot of a radio button. The default value is ##000000 (black).
symbolDisabledColor	Y	Format: color; check mark or radio button dot color if the control is disabled. The default value is ##848384 (dark gray).
texRollOverColor	Y	Format: color; text color of the label as you move the mouse pointer over the control. The default value is ##2B333C.
textSelectColor	Y	Format: color; text color of the label as you select the control. The default value is ##000000.
verticalGap	N	Gap between the label and the image in an `img` input when labelPlacement = "top" or "bottom". The default value is 2.

Styles for cftextarea tag and cfinput with text, password, or hidden type attributes

The following style applies to the following tags and tag-attribute combinations:

- `textarea`
- `cfinput type="hidden"`
- `cfinput type="password"`
- `cfinput type="text"`

Style	Inh	Description
disabledColor	Y	Format: color; disabled color of the Text Area.

Styles for cfselect with size attribute value of 1

The following styles apply to the `cfselect` tag when the `size` attribute is 1; that is, if the control displays one option at a time, with a drop-down list (also known as a combobox):

Style	Inh	Description
alternatingRowColors	Y	Format: comma delimited list of colors for rows in an alternating pattern. Value can be a list of two or more colors. Use only if you do not specify a `backgroundColor` style.
closeDuration	N	Time to close the drop-down list, in milliseconds. The default value is 250.
openDuration	N	Time to close the drop-down list, in milliseconds. The default value is 250.
rollOverColor	Y	Format: color; color of the background when the user rolls over an item. The default value is ##0EFFD6.
selectionColor	Y	Format: color; color of the background when the user selects an item. The default value is ##0DFFC1.

Styles for cfselect with size attribute value greater than 1

The following styles apply to the `cfselect` tag when the `size` attribute is greater than 1; that is, if the control is a list box that displays two or more options at a time:

Style	Inh	Description
alternatingRowColors	Y	Type: comma-delimited list of colors for rows in an alternating pattern. Value can be a list of two or more colors.
marginBottom	N	Format: length; number of pixels between the bottom of the row and the bottom of the text in the row. The default value is 0.
marginTop	N	Format: length; number of pixels between the top of the row and the top of the text in the row. The default value is 0.
rollOverColor	Y	Format: color; color of the background when the user moves the mouse pointer over the link. The default value is ##0EFFD6.
selectionColor	Y	Format: color; color of the background when the user selects the link. The default value is ##0DFFC1.
selectionDuration	N	The duration of the selection animation, in milliseconds. The default value is 250. Set to 0 to disable animation.
textRollOverColor	Y	Format: color; text color when the user moves the mouse pointer over the selection. The default value is ##02B33C.
textSelectedColor	Y	Format: color; text color when selected. The default value is ##005F33.

Styles for cfcalendar tag and cfinput with dateField type attribute

The following styles apply to the `cfcalendar` tag and `dateField` type of the `cfinput` tag:

Style	Inh	Description
headerColors	Y	Format: color; colors of the band at the top of the DateChooser control. Specify two values, separated by a comma. For a solid band, use the same color for both values. The default value is ##E6EEEE,##FFFFFF.
rollOverColor	Y	Format: color; color of the background when the user moves the mouse pointer over the DateField. The default value is ##E3FFD6.
selectionColor	Y	Format: color; color of the background when the user selects the DateField. The default value is ##CDFFC1.
todayColor	Y	Format: color; color of today's date. The default value is ##2B333C.

Styles for the cfgrid tag

The following styles apply to the `cfgrid` tag:

Style	Inh	Description
horizontalAlign	N	Horizontal alignment of children in the container. The default value is left. Possible values are left, center, and right.
horizontalGap	N	Number of pixels between children in the horizontal direction. The default value is 8.
marginBottom	N	Number of pixels between the container's bottom border and its content area. The default value is 0.
marginTop	N	Number of pixels between the container's top border and its content area. The default value is 0.
verticalAlign	N	Vertical alignment of children in the container. The default value is top. Possible values are top, middle, and bottom.
verticalGap	N	Number of pixels between children in the vertical direction. The default value is 8.

Styles for the cftree tag

The following styles apply to the `cftree` tag:

Style	Inh	Description
alternatingRowColors	Y	Type: Array; colors for rows in an alternating pattern. Value can be an Array of two or more colors.
depthColors	Y	Type: Array; array of colors used in the Tree control, in descending order.
indentation	N	Indentation for each tree level, in pixels. The default value is 8.
openDuration	N	Format: time; length of an open or close transition, in milliseconds. The default value is 250.
rollOverColor	Y	Format: color; color of the background when the user moves the mouse pointer over the link. The default value is ##E3FFD6.
selectionColor	Y	Format: color; color of the background when the user selects the link. The default value is ##CDFFC1.
selectionDuration	N	The duration of the selection animation, in milliseconds. The default value is 250. Set to 0 to disable animation.
textRollOverColor	Y	Format: color; color of the text when the user moves the mouse pointer over the entry. The default value is ##02B33C.
textSelectedColor	Y	Format: color; color of the text when the user selects the entry. The default value is ##005F33.

Chapter 7: Application.CFC Reference

You implement methods in Application.cfc to handle ColdFusion application events and set variables in the CFC to configure application characteristics.

Contents

Note: *Although Windows is case-insensitive, you should always start the Application.cfc filename with an uppercase A. Both application.cfc and Application.cfc are reserved words.*

Note: *If your application has an Application.cfc, and an Application.cfm or onRequestend.cfm page, ColdFusion ignores the CFM pages*

Application variables

The This scope for the Application.cfc contains several built-in variables which correspond to the attributes that you set in the `cfapplication` tag. You set the values of these variables in the CFC initialization code, before you define the CFC methods. You can access the variables in any method.

The following table briefly describes the variables that you can set to control the application behavior. For more details, see the `cfapplication` tag.

Variable	Default	Description
name	no name	The application name. If you do not set this variable, or set it to the empty string, your CFC applies to the unnamed application scope, which is the ColdFusion J2EE servlet context. For more information on unnamed scopes see "Sharing data between ColdFusion pages and JSP pages or servlets" on page 934 in the *ColdFusion Developer's Guide*.
applicationTimeout	Administrator value	Life span, as a real number of days, of the application, including all Application scope variables. Use the CFML `CreateTimeSpan` function to generate this variable's value.
clientManagement	Administrator value	Whether the application supports Client scope variables.
clientStorage	Administrator value	Where Client variables are stored; can be cookie, registry, or the name of a data source.
loginStorage	cookie	Whether to store login information in the Cookie scope or the Session scope.
sessionManagement	no	Whether the application supports Session scope variables.
sessionTimeout	Administrator value	Life span, as a real number of days, of the user session, including all Session variables. Use the CFML `CreateTimeSpan` function to generate this variable's value.
setClientCookies	True	Whether to send CFID and CFTOKEN cookies to the client browser.
setDomainCookies	False	Whether to set CFID and CFTOKEN cookies for a domain (not just a host).
scriptProtect	Administrator value	Whether to protect variables from cross-site scripting attacks.

Variable	Default	Description
secureJSON	Administrator value	A Boolean value that specifies whether to add a security prefix in front of the value that a ColdFusion function returns in JSON-format in response to a remote call. The default value is the value of the Prefix serialized JSON setting in the Administrator Server Settings > Settings page (which defaults to `false)`. You can override this value in the `cffunction` tag. For more information see "Improving security" on page 674 in the *ColdFusion Developer's Guide*.
secureJSONPrefix	Administrator value	The security prefix to put in front of the value that a ColdFusion function returns in JSON-format in response to a remote call if the `secureJSON` setting is `true`. The default value is the value of the Prefix serialized JSON setting in the Administrator Server Settings > Settings page (which defaults to `//`, the JavaScript comment character`)`. For more information see "Improving security" on page 674 in the *ColdFusion Developer's Guide*.
welcomeFileList		A comma-delimited list of names of files. Tells ColdFusion not to call the `onMissingTemplate` method if the files are not found. Use this variable to prevent ColdFusion from invoking the `onMissingTemplate` handler if all of the following items are true: • Your web server (e.g., web.xml file) has a welcome file list with CFML pages such as index.cfm that it tries to run if a URL specifies a path ending in a directory. • The web server sends a request for CFML pages the welcome list to ColdFusion without first determining if the page exists. • You want to support directory browsing in directories that do not have any of the files on the welcome file list. You specify this variable only if the Application.cfc file also specifies an `onMissingTemplate` handler. It should contain the same list of files as your web.xml welcome file list. **Note:** You do not need to use the welcomeFileList variable with most "pure" web servers, such as Apache. You do need to use the welcomeFileList variable with most integrated web and application servers, such as the integrated ColdFusion/JRun web server.

Method summary

The following table briefly describes the application event methods that you can implement in Application.CFC:

Method name	Method runs when
onApplicationEnd	The application ends: the application times out, or the server is stopped
onApplicationStart	The application first starts: the first request for a page is processed or the first CFC method is invoked by an event gateway instance, or a web services or Flash Remoting CFC.
onError	An exception occurs that is not caught by a try/catch block.
onMissingTemplate	ColdFusion received a request for a non-existent page.
onRequest	The onRequestStart method finishes. (This method can filter request contents.)
onRequestEnd	All pages in the request have been processed:
onRequestStart	A request starts
onSessionEnd	A session ends
onSessionStart	A session starts

All parameters to these methods are positional. You can use any names for these parameters.

When a request executes, ColdFusion runs the CFC methods in the following order:

1. onApplicationStart (if not run before for this application)
2. onSessionStart (if not run before for this session)
3. onRequestStart
4. onRequest
5. onRequestEnd

The onApplicationEnd, onSessionEnd, and onError CFCs are triggered by specific events.

onApplicationEnd

Description

Runs when an application times out or the server is shutting down.

Syntax

```
<cffunction name="onApplicationEnd" returnType="void">
    <cfargument name="ApplicationScope" required=true/>
    ...
</cffunction>
```

See also

onApplicationStart, Method summary, "Managing the application with Application.cfc" on page 229 in the *ColdFusion Developer's Guide*

Parameters

ColdFusion passes the following parameters to the method:

Parameters	Description
ApplicationScope	The application scope.

Returns

This method does not return a value; do not use the cfreturn tag.

Usage

Use this method for any clean-up activities that your application requires when it shuts down, such as saving data in memory to a database, or to log the application end to a file. You cannot use this method to display data on a user page, because it is not associated with a request. The application ends, even if this method throws an exception.

If you call this method explicitly, ColdFusion does not end the application; it does execute the method code, but does not lock the Application scope while the method executes.

You must use the *ApplicationScope* parameter to access the application scope; you cannot reference the scope directly; for example, use Arguments.ApplicationScope.myVariable, not Application.myVariable. This method can access the Server scope directly, but it does not have access to Session or Request scopes.

Note: *The application times out only if it is inactive for the time-out period. Sessions do not end, and the onSessionEnd method is not called when an application ends. For more information, see onSessionEnd.*

Example

```
<cffunction name="onApplicationEnd">
    <cfargument name="ApplicationScope" required=true/>
    <cflog file="#This.Name#" type="Information"
        text="Application #Arguments.ApplicationScope.applicationname# Ended" >
</cffunction>
```

onApplicationStart

Description

Runs when ColdFusion receives the first request for a page in the application.

Syntax

```
<cffunction name="onApplicationStart" returnType="boolean">
    ...
    <cfreturn Boolean>
</cffunction>
```

See also

onApplicationEnd, Method summary, “Managing the application with Application.cfc” on page 229 in he *ColdFusion Developer’s Guide*

Returns

A Boolean value: True if the application startup code ran successfully; False, otherwise. You do not need to explicitly return a True value if you omit the `cffunction` tag `returntype` attribute.

Usage

Use this method for application initialization code; for example, use it to set Application scope variables, to determine whether a required data source or other resource is available, or to log the application start. You do not have to lock the Application scope if you set Application variables in this method, and you can reference Application scope variables as you normally do; for example, as Application.myVariable.

This method can access the requested page’s Variables scope only if the Application.cfc file includes an `onRequest` method that calls the page.

If you call this method explicitly, ColdFusion does not start the application; it does execute the method code, but does not lock the Application scope while the method executes.

If this method throws an uncaught exception or returns False, the application does not start and ColdFusion does not process any pages in the application. In this case, ColdFusion will run the `onApplicationStart` method the next time a user requests a page in the application.

Example

The following example tests for the availability of a database. If the database is not available it reports and logs the error, and does not start the application; if it is available, the method initializes two Application scope variables.

```
<cffunction name="onApplicationStart">
    <cftry>
        <!--- Test whether the DB is accessible by selecting some data. --->
        <cfquery name="testDB" dataSource="cfdocexamples" maxrows="2">
            SELECT Emp_ID FROM employee
        </cfquery>
        <!--- If we get a database error, report an error to the user, log the
                error information, and do not start the application. --->
        <cfcatch type="database">
            <cfoutput>
                This application encountered an error<br>
                Please contact support.
            </cfoutput>
            <cflog file="#This.Name#" type="error"
                text="cfdocexamples DB not available. message: #cfcatch.message#
    Detail: #cfcatch.detail# Native Error: #cfcatch.NativeErrorCode#" >
```

```
            <cfreturn False>
        </cfcatch>
    </cftry>
    <cflog file="#This.Name#" type="Information" text="Application Started">
    <!--- You do not have to lock code in the onApplicationStart method that sets
            Application scope variables. --->
    <cfscript>
        Application.availableResources=0;
        Application.counter1=1;
    </cfscript>
    <cfreturn True>
</cffunction>
```

onError

Description

Runs when an uncaught exception occurs in the application.

Syntax

```
<cffunction name="onError" returnType="void">
    <cfargument name="Exception" required=true/>
    <cfargument name="EventName" type="String" required=true/>
    ...
</cffunction>
```

See also

Method summary, "Handling errors in Application.cfc" on page 232 in the *ColdFusion Developer's Guide*

Parameters

ColdFusion passes the following parameters to the method:

Parameter	Description
Exception	The ColdFusion Exception object. For information on the structure of this object, see the description of the `cfcatch` variable in the `cfcatch` description.
EventName	The name of the event handler that generated the exception. If the error occurs during request processing and you do not implement an `onRequest` method, this is the empty string.

Returns

This method does not return a value; do not use the `cfreturn` tag.

Usage

Use this method to handle errors in an application-specific manner. This method overrides any error handlers that you set in the ColdFusion Administrator or in `cferror` tags. It does not override try/catch blocks.

Whether the `onError` method can display output depends on where the error takes place, as follows:

- The `onError` method can display a message to the user if an error occurs during an `onApplicationStart`, `onSessionStart`, `onRequestStart`, `onRequest`, or `onRequestEnd` event method, or while processing a request.
- The `onError` method cannot display output to the user if the error occurs during an `onApplicationEnd` or `onSessionEnd` event method, because there is no available page context; however, it can log an error message.

If the `onError` event handler is triggered by a scope-specific event method, such as `onSessionStart`, the error prevents further processing at the level of that scope and any lower scopes. An `onError` event triggered by an `onSessionStart` method, for example, prevents further processing in the session, but not in the application.

If an exception occurs while processing the `onError` method, or if the `onError` method uses a `cfthrow` tag, the ColdFusion standard error handling mechanisms handle the exception. These mechanisms include: any error handlers specified by `cferror` tags in the Application.cfc initialization code, the site-wide error handler specified in the ColdFusion Administrator, and ColdFusion default error page. Therefore, you can use the `onError` method as a filter to handle selected errors, and use other ColdFusion error-handling techniques for the remaining errors.

Example

```
<cffunction name="onError">
    <cfargument name="Exception" required=true/>
    <cfargument type="String" name="EventName" required=true/>
    <!--- Log all errors. --->
```

```
    <cflog file="#This.Name#" type="error"
            text="Event Name: #Arguments.Eventname#" >
    <cflog file="#This.Name#" type="error"
            text="Message: #Arguments.Exception.message#">
    <cflog file="#This.Name#" type="error"
        text="Root Cause Message: #Arguments.Exception.rootcause.message#">
    <!--- Display an error message if there is a page context. --->
    <cfif NOT (Arguments.EventName IS "onSessionEnd") OR
            (Arguments.EventName IS "onApplicationEnd")>
        <cfoutput>
            <h2>An unexpected error occurred.</h2>
            <p>Please provide the following information to technical support:</p>
            <p>Error Event: #Arguments.EventName#</p>
            <p>Error details:<br>
            <cfdump var=#Arguments.Exception#></p>
        </cfoutput>
    </cfif>
</cffunction>
```

onMissingTemplate

Description

Runs when a request specifies a non-existent CFML page.

Syntax

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

See also

Method summary, "Handling errors in Application.cfc" on page 232 in the *ColdFusion Developer's Guide*

Parameters

ColdFusion passes the following parameters to the method:

Parameter	Description
targetPage	The path from the web root to the requested CFML page.

Returns

A Boolean value. `True` or no return value specifies that the event has been processed. `False` specifies that the event was not processed.

Usage

ColdFusion invokes this method when it encounters a file not found condition, that is, when a URL specifies a CFML page that does not exist.

The onMissingTemplate function should return `true` to indicate that the event has been processed, or return `false` to indicate that the event has not been processed. If the function does not return a value, it is assumed to be `true`. If the function returns `false`, ColdFusion invokes the standard error handler. If an error occurs within the `onMissingTemplate` function, the error handler is not invoked. Therefore, you should use try/catch blocks in your missing template handler and, if the catch block cannot handle the error, it should set the function return value to `false` so the standard error handler can report the error.

If the `onMissingTemplate` function is invoked, the `onApplicationStart` and `onSessionStart` event handlers are first invoked, if appropriate, but the `onRequestStart`, `onRequest` and `onRequestEnd` handlers are not invoked, and processing of the request terminates when the `onMissingTemplate` handler returns.

All standard scopes, including the Application, Session, and Client scopes, are available in the `onMissingTemplate` function, if they are enabled.

To include the contents of a page in the `onMissingTemplate` function, use the `cfinclude` tag. Do not any other method to include or redirect other page content, including tags and functions such as `cflocation`, `GetPageContext().forward()`, and `GetPageContext().include()`.

Use the `This.welcomeFileList` variable to keep this function from executing if all of the following are true:

- Your web server uses a welcome file list with one or more CFML files (such as index.cfm), that it tries to access when a user enters a URL that ends with a directory name

• The web server sends a request for a CFML page on the welcome list to ColdFusion without first determining if the page exists.

• You want to allow users to browse web directories that do not have any files on the list.

For more information, see `welcomeFileList` in Application variables.

Example

```
<!--- The web.xml welcome-file-list includes index.cfm.
    To allow web browsing, specify index.cfm in This.welcomFileList. --->
<cfset This.welcomeFileList="index.cfm">

<cffunction name="onMissingTemplate">
    <cfargument name="targetPage" type="string" required=true/>
    <!--- Use a try block to catch errors. --->
    <cftry>
        <!--- Log all errors. --->
        <cflog type="error" text="Missing template: #Arguments.targetPage#">
        <!--- Display an error message. --->
        <cfoutput>
            <h3>#Arguments.targetPage# could not be found.</h2>
            <p>You requested a non-existent ColdFusion page.<br />
                Please check the URL.</p>
        </cfoutput>
        <cfreturn true />
        <!--- If an error occurs, return false and the default error
            handler will run. --->
        <cfcatch>
            <cfreturn false />
        </cfcatch>
    </cftry>
</cffunction>
```

onRequest

Description

Runs when a request starts, after the `onRequestStart` event handler. If you implement this method, it **must** explicitly call the requested page to process it.

Syntax

```
<cffunction name="onRequest" returnType="void">
    <cfargument name="targetPage" type="String" required=true/>
    ...
    <cfinclude template="#Arguments.targetPage#">
    ...
</cffunction>
```

See also

`onRequestStart`, `onRequestEnd`, Method summary, "Managing requests in Application.cfc" on page 230 in the *ColdFusion Developer's Guide*

Parameters

ColdFusion passes the following parameters to the method:

Parameter	Description
`targetPage`	Path from the web root to the requested page.

Returns

This method does not return a value; do not use the `cfreturn` tag.

Usage

This event handler provides an optional request filter mechanism for CFML page requests (that is, .cfm pages requested using a browser). Use it to intercept requests to target pages and override the default behavior of running the requested pages. The following rules specify where and how you use the `onRequest` method.

- Implement this method only if the following are true:
    - The directory, and any subdirectories affected by this Application.cfc contain CFM files and do not contain any CFC files that are intended to be accessed as web services, using Flash Remoting, or using an event gateway.
    - You want to intercept the request and process it in a special way.
- If you do not implement this method, ColdFusion automatically calls the target page (or the CFC for a web service, Flash Remoting, or event gateway event).
- If you implement this method, it **must** explicitly call the target page, normally by using a `cfinclude` tag.
- Do **not** implement the onRequest method in any Application.cfc file that affects .cfc files that implement web services, process Flash Remoting or event gateway requests; ColdFusion will not execute the requests if you implement this method.
- Code in this method that precedes the call to the target page can perform the same functions as the `onRequestStart` method, and shares the Variables scope with the target page.
- Code in this method that follows the call to the target page can perform the same functions as the `onRequestEnd` method, and shares the Variables scope with the target page.
- If you implement this method, you can also implement the `onRequestStart` and `onRequestEnd` methods.

You can use this method to do preprocessing that is required for all requests. Typical uses include filtering and modifying request page contents (such as removing extraneous white space), or creating a switching mechanism that determines the exact page to display based on available parameters.

Example

```
<cffunction name="onRequest">
    <cfargument name="targetPage" type="String" required=true/>
    <cfset var content="">
    <cfsavecontent variable="content">
        <cfinclude template="#Arguments.targetPage#">
    </cfsavecontent>
    <cfoutput>
        #replace(content, "report", "MyCompany Quarterly Report", "all")#
    </cfoutput>
</cffunction>
```

onRequestEnd

Description

Runs at the end of a request, after all other CFML code.

Syntax

```
<cffunction name="onRequestEnd" returnType="void">
    <cfargument type="String" name="targetPage" required=true/>
    ...
</cffunction>
```

See also

onRequestStart, onRequest, Method summary, "Managing requests in Application.cfc" on page 230 in the *ColdFusion Developer's Guide*

Parameters

ColdFusion passes the following parameters to the method:

Parameter	Description
targetPage	Path from the web root to the requested page.

Returns

This method does not return a value; do not use the cfreturn tag.

Usage

This method has the same purpose as the onRequestEnd.cfm page. (You cannot use an onRequestEnd.cfm page if you have an Application.cfc file for your application.) This method runs before the request terminates; therefore, it can access the page context, and can generate output.

This method can be useful for gathering performance metrics, or for displaying dynamic footer information.

This method can access the requested page's Variables scope only if the Application.cfc file includes an onRequest method that calls the page. You can use Request scope variables to share data with the requested page, even if the Application.cfc file does not have an onRequest method.

If you call this method explicitly, ColdFusion does not end the request, but does execute the method code.

Example

The following example displays one of two footer pages depending on whether the user has logged in:

The onRequestEnd method in Application.cfc contains the following code:

```
<cffunction name="onRequestEnd">
    <cfargument type="String" name="targetPage" required=true/>
    <cfset theAuthuser=getauthuser()>
    <cfif theAuthUser NEQ "">
        <cfinclude template="authuserfooter.cfm">
    <cfelse>
        <cfinclude template="noauthuserfooter.cfm">
    </cfif>
</cffunction>
```

A very simple authuserfooter.cfm page consists of the following code:

```
<cfoutput>
```

```
    <h3>Thank you for shopping at our store, #theAuthUser#!</h3>
</cfoutput>
```

A very simple noauthuserfooter.cfm page consists of the following code:

```
<cfoutput>
    <h3>Remember, only registered users get all our benefits!</h3>
</cfoutput>
```

To test this example, implement code for logging in a user, or try the example with and without the following line in the `onRequestStart` Application.cfc method:

```
<cfloginuser name="Robert Smith" password="secret" roles="customer">
```

onRequestStart

Description

Runs when a request starts.

Syntax

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

See also

onRequest, onRequestEnd, Method summary, "Managing requests in Application.cfc" on page 230 in the *ColdFusion Developer's Guide*

Parameters

ColdFusion passes the following parameters to the method:

Parameters	Description
targetPage	Path from the web root to the requested page.

Returns

A Boolean value. Return False to prevent ColdFusion from processing the request. You do not need to explicitly return a True value if you omit the cffunction tag returntype attribute.

Usage

This method runs at the beginning of the request. It is useful for user authorization (login handling), and for request-specific variable initialization, such as gathering performance statistics.

If this method throws an exception (for example, if it uses the cfthrow tag), ColdFusion handles the error and does not process the request further.

If you call this method explicitly, ColdFusion does not start a request, but does execute the method code.

This method can access the requested page's Variables scope only if the Application.cfc file includes an onRequest method that calls the page. You can use Request scope variables to share data with the requested page even if Application.cfc does not have an onRequest method.

Example

This example uses the authentication code generated by the ColdFusion Dreamweaver Login wizard to ensure that the user is logged in. For Beta 2, the wizard generates code that is appropriate for Application.cfm only. To use this code with the Application.CFC, delete the generated Application.CFM

```
<cffunction name="onRequestStart">
    <cfargument name="requestname" required=true/>
    <!--- Authentication code, generated by the Dreamweaver Login wizard.
    <cfinclude template="mm_wizard_application_include.cfm">
    <!--- Regular maintenance is done late at night. During those hours, tell
             people to come back later, and do not process the request further. --->
    <cfscript>
        if ((Hour(now()) gt 1) and (Hour(now()) lt 3)) {
            WriteOutput("The system is undergoing periodic maintenance.
                Please return after 3:00 AM Eastern time.");
```

```
            return false;
        } else {
            this.start=now();
            return true;
        }
    </cfscript>
</cffunction>
```

onSessionEnd

Description

Runs when a session ends.

Syntax

```
<cffunction name="onSessionEnd" returnType="void">
    <cfargument name="SessionScope" required=True/>
    <cfargument name="ApplicationScope" required=False/>
    ...
</cffunction>
```

See also

onSessionStart, Method summary, “Managing sessions in Application.cfc” on page 230 in the *ColdFusion Developer’s Guide*

Parameters

ColdFusion passes the following parameters to the method:

Parameter	Description
SessionScope	The Session scope
ApplicationScope	The Application scope

Returns

This method does not return a value; do not use the cfreturn tag.

Usage

Use this method for any clean-up activities when the session ends. A session ends when the session is inactive for the session time-out period. You can, for example, save session-related data, such as shopping cart contents or whether the user has not completed an order, in a database, or do any other required processing based on the user’s status. You might also want to log the end of the session, or other session related information, to a file for diagnostic use.

If you call this method explicitly, ColdFusion does not end the session; it does execute the method code, but does not lock the Session.

You cannot use this method to display data on a user page, because it is not associated with a request.

You can access shared scope variables as follows:

- You must use the *SessionScope* parameter to access the Session scope. You cannot reference the Session scope directly; for example, use Arguments.SessionScope.myVariable, not Session.myVariable.
- You must use the *ApplicationScope* parameter to access the Application scope. You cannot reference the Application scope directly; for example, use Arguments.ApplicationScope.myVariable, not Application.myVariable. Use a named lock when you reference variables in the Application scope, as shown in the example.
- You can access the Server scope directly; for example, Server.myVariable.
- You cannot access the Request scope.

Sessions do not end, and the onSessionEnd method is not called when an application ends. The onSessionEnd does not execute if there is no active application, however.

Example

The following method decrements an Application scope session count variable and logs the session length.

```
<cffunction name="onSessionEnd">
    <cfargument name = "SessionScope" required=true/>
    <cfargument name = "AppScope" required=true/>
    <cfset var sessionLength = TimeFormat(Now() - SessionScope.started,
        "H:mm:ss")>
    <cflock name="AppLock" timeout="5" type="Exclusive">
        <cfset Arguments.AppScope.sessions = Arguments.AppScope.sessions - 1>
    </cflock>
    <cflog file="#This.Name#" type="Information"
        text="Session #Arguments.SessionScope.sessionid# ended. Length: #sessionLength#
Active sessions: #Arguments.AppScope.sessions#>
</cffunction>
```

onSessionStart

Description

Runs when a session starts.

Syntax

```
<cffunction name="onSessionStart" returnType="void">
    ...
</cffunction>
```

See also

onSessionEnd, Method summary, "Managing sessions in Application.cfc" on page 230 in the *ColdFusion Developer's Guide*

Returns

This method does not return a value; do not use the cfreturn tag.

Usage

This method is useful for initializing Session scope data, such as a shopping cart, or setting session-specific Application scope variables, such as for tracking the number of active sessions. You never need to lock the Session scope to set its variables using this method.

If you call this method explicitly, ColdFusion does not start a session; it does execute the method code, but does not lock the Session scope.

This method can access the requested page's Variables scope only if the Application.cfc file includes an onRequest method that calls the page.

Example

The following onSessionStart example initializes some Session scope variables and increments an Application scope counter of active sessions.

```
<cffunction name="onSessionStart">
    <cfscript>
        Session.started = now();
        Session.shoppingCart = StructNew();
        Session.shoppingCart.items =0;
    </cfscript>
        <cflock scope="Application" timeout="5" type="Exclusive">
            <cfset Application.sessions = Application.sessions + 1>
    </cflock>
</cffunction>
```

Chapter 8: ColdFusion Event Gateway Reference

Java interfaces are available for building ColdFusion custom CFXs in Java.

Contents

Note: *The following CFML functions also apply to gateway application development:* `GetGatewayHelper`, `SendGatewayMessage`.

Gateway development interfaces and classes

The ColdFusion event gateway system is defined in the coldfusion.eventgateway package. Gateway developers implement two interfaces and use several classes, as follows:

- Gateway interface
- GatewayHelper interface
- GatewayServices class
- CFEvent class
- Logger class

Gateway interface

`coldfusion.eventgateway.Gateway`

Interface for implementing ColdFusion event gateways.

A class that implements this interface defines a ColdFusion event gateway type that you can use in ColdFusion applications. The class must implement the following methods:

Signature	Description
GatewayName([String id[, StringconfigFile]])	The gateway constructor.
String getGatewayID()	Returns the gateway ID.
GatewayHelper getHelper()	Returns an instance of the GatewayHelper class for this gateway type. instance, or null if the gateway does not have a GatewayHelper class.
int getStatus()	Gets the event gateway status.
String outgoingMessage(coldfusion.eventgateway.CFEvent *cfmessage*)	Handles a message sent by ColdFusion and processes it to send to a message receiver.
void restart()	Restarts a running event gateway.
void setCFCListeners(String[] *listeners*)	Identifies the CFCs that listen for incoming messages from the event gateway.
void setGatewayID(*String id*)	Sets the gateway ID that uniquely identifies the Gateway instance.
void start()	Starts the event gateway.
void stop()	Stops the event gateway.

Constructor

Description

Instantiates a gateway.

Category

Event Gateway Development

Syntax

```
public void gatewayName()
public void gatewayName(String id)
public void gatewayName(String id, String configFile)
```

See also

`setGatewayID`, "Class constructor" on page 1135 in the *ColdFusion Developer's Guide*.

Parameters

Parameter	Description
`id`	The identifier for the gateway instance
`configFile`	The absolute path to the gateway configuration file.

Usage

If your gateway requires a configuration file, use the constructor with two parameters. Otherwise, you can use either the default constructor or the single parameter version; ColdFusion always uses the `setGatewayID` method to set the ID.

Example

The following example shows the two argument constructor implemented in the ColdFusion SocketGateway class:

```
public SocketGateway(String id, String configpath) {
        propsFilePath=configpath;
        try {
            FileInputStream propsFile = new FileInputStream(propsFilePath);
            properties.load(propsFile);
            propsFile.close();
            this.loadProperties();
        }
        catch (FileNotFoundException f) {
            // do nothing. use default value for port.
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        gatewayID = id;
        gatewayService = GatewayServices.getGatewayServices();
    }
```

getGatewayID

Description

Returns the gateway ID that identifies the Gateway instance.

Category

Event Gateway Development

Syntax

```
public String getGatewayID()
```

See also

`setGatewayID`, "Providing Gateway class service and information routines" on page 1137 in the *ColdFusion Developer's Guide*.

Usage

This method returns a string value that is set by the `setGatewayID` method.

Example

The following example is the ColdFusion SocketGateway class `getGatewayID` method:

```
public String getGatewayID()
    {
        return gatewayID;
    }
```

getHelper

Description

Returns an instance of the gatewayHelper class, if any for the gateway type.

Category

Event Gateway Development

Syntax

```
public GatewayHelper getHelper()
```

See also

`GatewayHelper interface`; “Providing Gateway class service and information routines” on page 1137 in the *ColdFusion Developer’s Guide*.

Returns

A coldfusion.eventgateway.GatewayHelper class instance, or null if the gateway does not have a GatewayHelper class.

Usage

ColdFusion calls this method when a ColdFusion application calls the CFML `GetGatewayHelper` function. The application then uses the gatewayHelper object methods to call gateway-specific utility methods, such as instant message buddy management methods.

Example

The following example is the ColdFusion SocketGateway class `getHelper` method:

```
public GatewayHelper getHelper()
    {
        // SocketHelper class implements the GatewayHelper interface
        return new SocketHelper();
    }
```

getStatus

Description
Returns the gateway status.

Category
Event Gateway Development

Syntax
```
public int getStatus()
```

See also
"Providing Gateway class service and information routines" on page 1137 in the *ColdFusion Developer's Guide*

Returns
An integer status value. The Gateway interface defines the following status constants:

- STARTING
- RUNNING
- STOPPING
- STOPPED
- FAILED

Example
The following example is the ColdFusion SocketGateway class `getStatus` method:

```
public int getStatus()
    {
        return status;
    }
```

outgoingMessage

Description
Sends a message from ColdFusion to a message receiver.

Category
Event Gateway Development

Syntax
```
public String outgoingMessage(coldfusion.eventgateway.CFEvent message)
```

See also
"Responding to a ColdFusion function or listener CFC" on page 1141 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`message`	A coldfusion.eventgateway.CFEvent instance containing the message to be sent.

Returns
A gateway-specific string, such as a message ID or a status indicator.

Usage
This method handles a message sent by ColdFusion and processes it as needed by the gateway type to send a message to the (usually external) message receiver. ColdFusion calls this method when the listener method of a listener CFC returns a message or when a ColdFusion application calls the `SendGatewayMessage` function. ColdFusion passes the String returned by this method back as the return value of a CFML `SendGatewayMessage` function.

Example
The following example is the ColdFusion SocketGateway class `outgoingMessage` method:

```
public String outgoingMessage(coldfusion.eventgateway.CFEvent cfmsg)
{
    String retcode="ok";
    // Get the table of data returned from the event handler
    Map data = cfmsg.getData();
    String message = (String) data.get("MESSAGE");
    // find the right socket to write to from the socketRegistry hashtable
    if (cfmsg.getOriginatorID() != null && message != null)
    {
        SocketServerThread st =
            ((SocketServerThread)socketRegistry.get(cfmsg.getOriginatorID()));
        if(st != null)
            st.writeOutput(message);
        else
        {
            log.error("Cannot send outgoing message. OriginatorID '" +
                cfmsg.getOriginatorID() + "' is not a valid socket id.");
            retcode="failed";
        }
    }
    else if (data.get("OriginatorID") != null && message != null)
    {
        SocketServerThread st =
            ((SocketServerThread)socketRegistry.get(data.get("OriginatorID")));
```

```
        if(st != null)
            st.writeOutput(message);
        else
        {
            log.error("Cannot send outgoing message. OriginatorID '" +
                data.get("OriginatorID") + "' is not a valid socket id.");
            retcode="failed";
        }
    }
    else
    {
        log.error("Cannot send outgoing message. OriginatorID/MESSAGE is not
            available.");
        retcode="failed";
    }
    return retcode;
}
```

restart

Description

Stops a gateway if it is running and starts it up.

Category

Event Gateway Development

Syntax

```
public void restart()
```

See also

start, stop

Usage

In most cases, you implement this method as a call to the stop method followed by a start method, but you may be able to optimize the restart method based on the type of gateway.

Example

The following example is the ColdFusion SocketGateway class `restart` method:

```
public void restart()
{
    stop();
    start();
}
```

setCFCListeners

Description

Sets the array of listener CFCs that the gateway sends messages to.

Category

Event Gateway Development

Syntax

```
public void setCFCListeners(String[] listeners)
```

See also

`Constructor`, `getGatewayID`, `setCFCPath`, "Providing Gateway class service and information routines" on page 1137 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`listeners`	Array of absolute file paths to CFCs to which the gateway forwards messages when it gets events.

Usage

When ColdFusion starts a gateway instance, it calls this method with the names in the instance's listener list in the ColdFusion Administrator. ColdFusion can also call this method if the ColdFusion Administrator listener list changes while the gateway is running.

Example

The following example is the ColdFusion SocketGateway class `setCFCListeners` method:

```
public void setCFCListeners(String[] listeners)
{
    ArrayList aListeners = new ArrayList();
    for(int i = 0; i<listeners.length; i++)
    {
        aListeners.add(listeners[i]);
    }
        // Try not to pull the rug out from underneath a running message
    synchronized (cfcListeners)
    {
        cfcListeners = aListeners;
    }
}
```

setGatewayID

Description

Sets the gateway ID that uniquely identifies the Gateway instance.

Category

Event Gateway Development

Syntax

```
public void setGatewayID(String id)
```

See also

`Constructor`, `getGatewayID`, "Providing Gateway class service and information routines" on page 1137 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`id`	The identifier for this gateway instance.

Usage

This method sets a string value that is returned by the `getGatewayID` method. ColdFusion calls this method to set the gateway ID with the value specified in the gateway instance configuration in the ColdFusion Administrator before it starts the event gateway, even if the Gateway constructor also sets the ID.

Example

The following example is the ColdFusion SocketGateway class `setGatewayID` method:

```
public void setGatewayID(String id)
    {
        gatewayID = id;
    }
```

start

Description
starts a gateway running.

Category
Event Gateway Development

Syntax
```
public void start()
```

See also
`restart`, `stop`, "The start method" on page 1137 in the *ColdFusion Developer's Guide*

Usage
Start a gateway by performing any required initialization. This method starts any listener thread or threads that monitor the gateway's event source. The ColdFusion Administrator calls this function when it starts a gateway instance.

This method should update the status information that is returned by the `getStatus` method to indicate when the gateway is starting and when the gateway is running.

The ColdFusion Administrator Gateway Types page lets you specify a time-out for the gateway startup, and whether to kill the gateway on startup time-out. If you enable the kill option and the `start` method does not return in the time-out period, ColdFusion kills the thread that called this function.

Example
The following example is the ColdFusion SocketGateway class `restart` method:

```
public void start()
    {
        status = STARTING;
        listening=true;
        // Start up event generator thread
        Runnable r = new Runnable()
        {
            public void run()
            {
                socketServer();
            }
        };
        Thread t = new Thread(r);
        t.start();
        status = RUNNING;
    }
```

stop

Description
Stops a gateway if it is running.

Category
Event Gateway Development

Syntax
```
public void stop()
```

See also
`restart`, `start`, "The stop method" on page 1138 in the *ColdFusion Developer's Guide*

Usage
Stops a gateway by performing any required clean-up operations. This method stops any listener thread or threads that monitor the gateway's event source and releases any other resources. The ColdFusion Administrator calls this function when it stops a gateway instance.

This method should update the status information that is returned by the `getStatus` method to indicate when the gateway is stopping and when the gateway is stopped.

Example
The following example is the ColdFusion SocketGateway class `stop` method:

```
public void stop()
    {
        status = STOPPING;
        listening=false;
        Enumeration e = socketRegistry.elements();
        while (e.hasMoreElements()) {
            try
            {
                ((SocketServerThread)e.nextElement()).socket.close();
            }
            catch (IOException e1) {
                e1.printStackTrace();
            }
        }
        if (serverSocket != null) {
            try
            {
                serverSocket.close();
            }
            catch (IOException e1) {
            }
            serverSocket = null;
        }
        status = STOPPED;
    }
```

GatewayHelper interface

coldfusion.eventgateway.GatewayHelper

ColdFusion includes a coldfusion.eventgateway.GatewayHelper Java marker interface, with no methods. Implement this interface to define a class that provides gateway-specific utility methods to the ColdFusion application or listener CFC. For example, an instant messaging event gateway might use a helper class to provide buddy list management methods to the application. The Gateway class must implement a `getHelper` method that returns the helper class, or null if you do not implement the interface.

For information on GatewayHelper classes, see "GatewayHelper class" on page 1133.

GatewayServices class

`coldfusion.eventgateway.GatewayServices`

The Gateway class uses the coldfusion.eventgateway.GatewayServices class to interact with the ColdFusion event gateway services. This class has the following methods:

Signature	Description
`GatewayServices getGatewayServices()`	Static method that returns the GatewayServices object.
`boolean addEvent(CFEvent msg)`	Sends a `CFEvent` instance to ColdFusion for dispatching to a listener CFC.
`coldfusion.eventgateway.LoggergetLogger([String logfile])`	Returns a ColdFusion logger object that the event gateway can use to log information in a file.
`int getMaxQueueSize()`	Returns the maximum size of the ColdFusion event queue, as set in the ColdFusion Administrator.
`int getMaxQueueSize()`	Returns the current size of the ColdFusion event queue that handles all messages for all gateways.

getGatewayServices

Description

Static method that returns the GatewayServices object. Gateway code can call this method at any time, if required.

Category

Event Gateway Development

Syntax

```
GatewayServices getGatewayServices()
```

See also

"GatewayServices class" on page 1132 in the *ColdFusion Developer's Guide*

Returns

The GatewayServices object.

Usage

Gateway constructors can call this method to get a convenient reference to the GatewayServices class and its methods.

Example

The following Socket gateway constructor code sets the GatewayServices variable:

```
public SocketGateway(String id)
    {
        gatewayID = id;
        gatewayService = GatewayServices.getGatewayServices();
    }
```

Calls to GatewayServices methods, such as the following, use the returned value.

```
boolean sent = gatewayService.addEvent(event);
```

addEvent

Description
Sends a CFEvent instance to ColdFusion for dispatching to a listener CFC.

Category
Event Gateway Development

Syntax
```
boolean addEvent(CFEvent msg)
```

See also
getMaxQueueSize, getMaxQueueSize, "Responding to incoming messages" on page 1139 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
msg	The CFEvent object containing the message to be queued for delivery to the listener CFC.

Returns
True if the event was added to the gateway services queue for delivery, false, otherwise. Therefore, a true response does not indicate that the message was delivered.

Usage
The event gateway must use this method to send incoming messages to the application for processing.

Example
The following example from the ColdFusion SocketGateway code sends an event to all listener CFCs:

```
for (int i = 0; i < listeners.length; i++) {
    String path = listeners[i];
    CFEvent event = new CFEvent(gatewayID);
    Hashtable mydata = new Hashtable();
    mydata.put("MESSAGE", theInput);
    event.setData(mydata);
    event.setGatewayType("SocketGateway");
    event.setOriginatorID(theKey);
    event.setCfcMethod(cfcEntryPoint);
    event.setCfcTimeOut(10);
    if (path != null)
        event.setCfcPath(path);
    boolean sent = gatewayService.addEvent(event);
    if (!sent)
        log.error("SocketGateway(" + gatewayID + ") Unable to put message on
            event queue. Message not sent from " + gatewayID + ", thread " +
            theKey + ".Message was " + theInput);
}
```

getLogger

Description

Returns a ColdFusion Logger object that the event gateway can use to log information in a file.

Category

Event Gateway Development

Syntax

```
coldfusion.eventgateway.Logger getLogger([String logfile])
```

See also

`Logger class`, "Logging events and using log files" on page 1142 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`logfile`	The name, without an extension, of a log file in the ColdFusion logs directory. ColdFusion automatically appends a .log extension to the name. If the file does not exist, ColdFusion creates it when it logs the first message. By default, ColdFusion logs to the eventgateway.log file.

Returns

A ColdFusion logger object

Usage

The Logger class has five methods: `debug`, `info`, `warn`, `error`, and `fatal`, that correspond to the severity level that is set in the log message. Each method takes a message string, a Throwable class object, or both.

If you pass a Throwable object to these methods, ColdFusion writes the exception information in the exceptions.log file.

Example

The ColdFusion example DirectoryWatcherGateway includes the following line in the constructor to get a logger object:

```
// We create our own log file, which will be named "watcher.log"
logger = gatewayService.getLogger("watcher");
```

The following code, from the start of the routine that loads information from the configuration file, uses this object to log the initialization.

```
// Load the properties file to get our settings
protected void loadconfig() throws ServiceRuntimeException
{
    // load config
    logger.info("DirectoryWatcher (" + gatewayID + ") Initializing
        DirectoryWatcher gateway with configuration file " + config);
.
.
.
```

getMaxQueueSize

Description

Returns the maximum size of the ColdFusion event queue, as set in the ColdFusion Administrator.

Category

Event Gateway Development

Syntax

```
int getMaxQueueSize()
```

See also

addEvent, getQueueSize

Returns

The integer maximum number of messages that the gateway services queue can hold.

Usage

If the queue length reaches this value, the `addEvent` method will not add its message to the processing queue. You can use this method and the `getQueueSize` method to control the rate of event queuing and to help diagnose any throughput problems in your gateways.

Example

The following example logs the queue size, maximum queue size, and other information if a `gateway-Service.addEvent` method fails to queue a message for delivery to a listener CFC. (It uses an internal method to construct the error message string.)

```
boolean sent = gatewayService.addEvent(cfmsg);
if (!sent)
{
    logger.error(RB.getString(this, "IMGateway.cantAddToQueue",
        gatewayType, gatewayID, ((path != null) ? path : "default"),
        Integer.ToString(gatewayService.getQueueSize()),
        Integer.ToString(gatewayService.getMaxQueueSize())));
}
```

getQueueSize

Description

Returns the current size of the ColdFusion event queue that handles all messages for all gateways.

Category

Event Gateway Development

Syntax

```
int getQueueSize()
```

See also

addEvent, getMaxQueueSize

Returns

The integer number of messages in the gateway message queue that are waiting to be delivered to CFCs.

Usage

You can use this method and the `getMaxQueueSize` method to control the rate of event queuing and to help diagnose any throughput problems in your gateways.

Example

The following example logs the queue size, maximum queue size, and other information if a `gateway-Service.addEvent` method fails to queue a message for delivery to a listener CFC. (It uses an internal method to construct the error message string.)

```
boolean sent = gatewayService.addEvent(cfmsg);
if (!sent)
{
    logger.error(RB.getString(this, "IMGateway.cantAddToQueue",
        gatewayType, gatewayID, ((path != null) ? path : "default"),
        Integer.ToString(gatewayService.getQueueSize()),
        Integer.ToString(gatewayService.getMaxQueueSize())));
}
```

CFEvent class

`coldfusion.gateway.CFEvent`

The Gateway class sends and receives `CFEvent` instances to communicate with the ColdFusion listener CFC or application. The `CFEvent` instances correspond to CFML CFEvent structures that ColdFusion application listener CFC methods receive and contain the message structures that ColdFusion application code sends to the gateway.

- The Gateway notifies ColdFusion of a message by sending a CFEvent instance in `GatewayServices.addEvent` method.
- The Gateway receives a CFEvent instance when ColdFusion calls the gateway's `outgoingMessage` method.

The CFEvent Class extends the java.util.Hashtable class and has the following methods:

Methods	Description
`CFEvent(String gatewayID)`	`CFEvent` constructor.
`String getGatewayID()`	Returns the gateway ID (set in the CFEvent constructor).
`void setCFCMethod(String method)` `String getCFCMethod()`	Sets or gets the name of the CFC method that receives an incoming message.
`void setCFCPath(String path)` `String getCFCPath()`	Sets or gets the path to the application listener CFC that processes the event.
`void setCFCTimeout(String seconds)` `String getCFCTimeout()`	Sets or gets the time-out, in seconds, for the listener CFC to process the event request.
`void setData(Map data)` `Map getData()`	Sets or gets the event data structure, which contains the message contents and any other gateway-specific information.
`void setGatewayType(String type)` `String getGatewayType()`	Sets or gets the event gateway type identifier, such as SMS.
`void setOriginatorID(String id)` `String getOriginatorID()`	Sets or gets the gateway- or protocol-specific Identity of the originator of a message.

CFEvent

Description

CFEvent constructor.

Category

Event Gateway Development

Syntax

```
CFEvent(String gatewayID)
```

See also

`getGatewayID`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`gatewayID`	The ID of the gateway. This parameter indicates the source of the message and must be the value that is passed in the Gateway constructor or set using the Gateway `setGatewayID` method. The SMS gateway ID must be 21 characters or fewer.

Usage

This method creates a container for an event gateway message that you send to ColdFusion gateway services in a `gatewayServices.addEvent` method for delivery to a CFC listener method.

Example

The following example, based on code for the ColdFusion asynchronous CFML gateway, sends a message to that the gateway has received to a CFC:

```
public String outgoingMessage(coldfusion.eventgateway.CFEvent cfmsg)
{
    // Get the data
    Map data = cfmsg.getData();
    boolean status = true;
    if (data != null)
    {
        // create an event
        CFEvent event = new coldfusion.eventgateway.CFEvent(gatewayID);
        //set the event field values
        event.setGatewayType("CFMLGateway");
        event.setOriginatorID("CFMLGateway");
        event.setData(data);
        // send it to the event service
        status = gatewayService.addEvent(event);
    }
        return new Boolean(status).ToString();
}
```

getCFCMethod

Description

Gets the name of the CFC method that processes the message.

Category

Event Gateway Development

Syntax

```
String getCFCMethod()
```

See also

`getCFCPath`, `getCFCTimeout`, `setCFCMethod`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Returns

For incoming messages, the name of the method that gateway services will call in the listener CFC, as set by the `setCFCMethod` method. If `setCFCMethod` has not been called, returns null, and not `onIncomingMessge`, which ColdFusion gateway services uses by default. Outgoing messages that are returned by a CFC in response to an incoming message also have the CFC method name in this field if the gateway set the field on the incoming message.

Usage

Most event gateways do not need to use this method. This method could be useful if a gateway sends messages to multiple CFC Methods and must determine which method is responding.

getCFCPath

Description
Gets the path to the listener CFC that processes this message.

Category
Event Gateway Development

Syntax
```
String getCFCPath()
```

See also
`getCFCMethod`, `getCFCTimeout`, `setCFCPath`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Returns
An absolute path to the application listener CFC that will process the event, as set by the `setCFCPath` method. If the `setCFCPath` method has not been called, returns null, not the path specified in the ColdFusion Administrator and used by default by gateway services. Outgoing messages that are returned by a CFC in response to an incoming message also have the CFC method name in this field if the gateway set the field on the incoming message.

Usage
Most event gateways do not need to use this method. This method could be useful if a gateway sends messages to multiple CFCs and must determine which CFC is responding.

getCFCTimeout

Description

Gets the time-out, in seconds, for the listener CFC to process the event request.

Category

Event Gateway Development

Syntax

```
String getCFCTimeout()
```

See also

`getCFCMethod`, `getCFCPath`, `setCFCTimeout`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Returns

The listener CFC time-out, in seconds, as set by the `setCFCTimeout` method, or null.

Usage

Most gateways do not need to use this function.

When ColdFusion calls a listener CFC method to process the event, and the CFC does not process the event in the specified time-out period, ColdFusion terminates the request and logs an error in application.log file. By default ColdFusion uses the Timeout Request value set on the Server Settings page in the ColdFusion Administrator.

getData

Description
Returns the data Map that contains the message contents and other gateway-specific information.

Category
Event Gateway Development

Syntax
```
Map getData()
```

See also
setData, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Returns
The event data structure, or null. This structure includes the message contents being passed by the gateway and any other gateway-specific information.

Usage
The contents of the data Map depends on the event gateway type. Typical fields include the message contents, originator ID, destination ID, and if a gateway (such as the ColdFusion SMS gateway) supports multiple commands, the command.

Note: *The returned Map object has case-insensitive keys.*

Example
The following outgoingMessage method from the SocketGateway example gateway gets the message contents from the CFEvent data field of an outgoing message. If the CFEvent object does not include an OriginatorID field, it also tries to get the originator ID from the data field.

```
public String outgoingMessage(coldfusion.eventgateway.CFEvent cfmsg)
{
    String retcode="ok";
    // Get the table of data returned from the event handler
    Map data = cfmsg.getData();
    String message = (String) data.get("MESSAGE");
    // find the right socket to write to from the socketRegistry hashtable
    if (cfmsg.getOriginatorID() != null)
        ((SocketServerThread)socketRegistry.get(cfmsg.getOriginatorID())).
            writeOutput(message);
    else if (data.get("OriginatorID") != null)
        ((SocketServerThread)socketRegistry.get(data.get("OriginatorID"))).
            writeOutput(message);
    else {
        System.out.println("cannot send outgoing message. OriginatorID is not
            available.");
            retcode="failed";
    }
    return retcode;
}
```

getGatewayID

Description
Returns the gateway ID field of the CFEvent object.

Category
Event Gateway Development

Syntax
```
String getGatewayID(CFEvent event)
```

See also
`CFEvent`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Returns
The gateway ID of the CFEvent object, or null.

Usage
Most gateways do not need to use this method. The gateway ID is set in the `CFEvent` constructor and normally corresponds to the gateway that is handling the event.

getGatewayType

Description
Returns the gateway type field of the CFEvent object.

Category
Event Gateway Development

Syntax
```
String getGatewayType()
```

See also
`setGatewayType`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Returns
The gateway type of the CFEvent object, or null.

Usage
Most gateways do not need to use this method.

getOriginatorID

Description

Identifies the originator of an incoming message. Some gateway types also use this field for the destination of an outgoing message.

Category

Event Gateway Development

Syntax

```
String getOriginatorID()
```

See also

`setOriginatorID`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Returns

The protocol-specific identifier of the message originator, or null.

Example

The `outgoingMessage` method of the SocketGateway example gateway uses the `getOriginatorID` method to determine the destination of an outgoing message. This way, a listener CFC that sends a response back to the originator does not have to explicitly set a destination in the return variable. If the field is empty, (as it is in messages sent by the CFML `SendGatewayMessage` function) the gateway tries to get the destination from the `CFevent` data field.

```
public String outgoingMessage(coldfusion.eventgateway.CFEvent cfmsg)
{
    String retcode="ok";
    // Get the table of data returned from the event handler
    Map data = cfmsg.getData();
    String message = (String) data.get("MESSAGE");
    // find the right socket to write to from the socketRegistry hashtable
    if (cfmsg.getOriginatorID() != null)
        ((SocketServerThread)socketRegistry.get(cfmsg.getOriginatorID())).
            writeOutput(message);
    else if (data.get("OriginatorID") != null)
        ((SocketServerThread)socketRegistry.get(data.get("OriginatorID"))).
            writeOutput(message);
    else
    {
        System.out.println("cannot send outgoing message. OriginatorID is not
                available.");
        retcode="failed";
    }
    return retcode;
}
```

setCFCMethod

Description

Sets the name of the CFC method that will processes an incoming message.

Category

Event Gateway Development

Syntax

```
void setCFCMethod(String method)
```

See also

`getCFCMethod`, `setCFCPath`, `setCFCTimeout`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`method`	The method in the listener CFC that ColdFusion will call to process this event. If you do not use this method in your gateway, ColdFusion invokes the `onIncomingMessage` method,

Usage

Gateways that use a single CFC listener method do not need to use this method if the listener CFC method is named `onIncomingMessage`. For the sake of consistency, Adobe recommends that any event gateway with a single listener not override this default.

A gateway, such as the ColdFusion XMPP gateway, that uses different listener methods for different message types uses this method to identify the destination method.

Example

The following example code comes from the ColdFusion XMPP gateway incoming message handler. It creates a CFEvent object and sets the method that will handle tests based on the message type.

```
CFEvent cfmsg = new CFEvent(gatewayID);
cfmsg.setOriginatorID(sender);
cfmsg.setGatewayType(gatewayType);
if(messageType == IMessage.IM)
    {
    // default for normal messages
    cfmsg.setCfcMethod(onIncomingMessageFunction);
    }
    //if the message is an authorization request
else if(messageType == IMessage.AUTH_REQUEST)
    {
    cfmsg.setCfcMethod(onAddBuddyRequestFunction);
    message = "Requesting authorization to add '" + recipient + "' to '"
    + sender + "' buddy list and view '" + recipient + "' presence.";
    } // Code snipped here for brevity.
```

setCFCPath

Description
Specifies the listener CFC that will process this event.

Category
Event Gateway Development

Syntax
```
void setCFCPath(String path)
```

See also
getCFCPath, setCFCMethod, setCFCTimeout, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
path	An absolute path to the application listener CFC that will process the event. If you do not call this method in your gateway, ColdFusion uses the first path configured for the event gateway instance on the Event Gateways page in the ColdFusion Administrator.

Usage
By default, ColdFusion delivers messages to the CFC in the first path configured for the event gateway instance on the Event Gateways page in the ColdFusion Administrator.

If your application supports multiple listener CFCs, use this method to set each listener CFC and then call the `gatewayService.addEvent` method to send the event to the CFC.

Example
The following example code is based on the Socket gateway processInput method that takes input from the socket and sends it to the CFC listener methods. The listeners variable contains an array of listener CFCS and is set by the gateway's `setCFCListeners` method, which ColdFusion calls when it starts the gateway.

```
for (int i = 0; i < listeners.length; i++)
{
    String path = listeners[i];
    CFEvent event = new CFEvent(gatewayID);
    Hashtable mydata = new Hashtable();
    mydata.put("MESSAGE", theInput);
    event.setData(mydata);
    event.setGatewayType("SocketGateway");
    event.setOriginatorID(theKey);
    event.setCFCMethod(cfcEntryPoint);
    event.setCFCTimeout(10);
    if (path != null)
        event.setCFCPath(path);boolean sent = gatewayService.addEvent(event);
}
```

setCFCTimeout

Description

Sets the time-out, in seconds, during which the listener CFC must process the event request and return before ColdFusion gateway services terminates the request.

Category

Event Gateway Development

Syntax

```
void setCFTimeout(String timeout)
```

See also

getCFCTimeout, setCFCMethod, setCFCPath, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
timeout	The CFC time-out period, in seconds.

Usage

When ColdFusion calls a listener CFC method to process the event, and the CFC does not return in the specified time-out period, ColdFusion terminates the request and logs an error in the application.log file.

If you do not use this method, ColdFusion uses the Timeout Request value set on the Server Settings page in the ColdFusion Administrator.

Use this method if your messages require a longer or shorter time-out period than standard ColdFusion HTML requests.

Example

The following example code is based on the Socket gateway processInput method that takes input from the socket and sends it to the CFC listener methods. It sets the CFC time-out to 10 seconds.

```
for (int i = 0; i < listeners.length; i++)
{
    String path = listeners[i];
    CFEvent event = new CFEvent(gatewayID);
    Hashtable mydata = new Hashtable();
    mydata.put("MESSAGE", theInput);
    event.setData(mydata);
    event.setGatewayType("SocketGateway");
    event.setOriginatorID(theKey);
    event.setCfcMethod(cfcEntryPoint);
    event.setCfcTimeOut(10);
    if (path != null)
        event.setCfcPath(path);
    boolean sent = gatewayService.addEvent(event);
}
```

setData

Description
Adds the gateway-specific data, including any message contents, as a Java Map to the CFEvent object

Category
Event Gateway Development

Syntax
```
void setData(Map data)
```

See also
getData, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
data	The incoming message and any additional gateway-specific event data.

Usage
The number of fields and their contents depend on the event gateway type. The Map keys must be strings.

Because Coldfusion is not case sensitive, it converts the Map passed in the setData method to a case insensitive Map. As a result, do not create entries in the data with names that differ only in case.

Example
The following code shows the routine from the example JMS gateway that handles incoming messages. It puts the JMS message ID and contents in a data HashMap, and uses it in the setData method:

```
public void handleMessage(String msg, String topicName, String msgID) {
    coldfusion.eventgateway.Logger log = getGatewayServices().getLogger();
    Map data = new HashMap();
    CFEvent cfMsg = new CFEvent(getGatewayID());
    data.put("msg", msg);
    data.put("id", msgID);
    cfMsg.setData(data);
    cfMsg.setOriginatorID(topicName);
    cfMsg.setGatewayType("JMS");
    if (sendMessage(cfMsg)) {
            log.info("Added message '" + msgID + "' to queue.");
    } else {
            log.error("Failed to add message '" + msgID + "' to queue.");
    }
}
```

setGatewayType

Description

Identifies the type of event gateway.

Category

Event Gateway Development

Syntax

```
void setGatewayType(String gatewayType)
```

See also

`getGatewayType`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`gatewayType`	A gateway type identifier.

Usage

For the sake of consistency, use the same name in this method and in the Type Name field when you add the event gateway type in the ColdFusion Administrator. Gateway application CFCs that handle multiple gateway types, such as those in an instant messaging application that handles multiple instant messaging providers, could use this field to determine the protocol type and any gateway type-specific actions.

Example

The following code shows the routine from the example JMS gateway that handles incoming messages. It sets the gateway type to JMS:

```
public void handleMessage(String msg, String topicName, String msgID) {
    coldfusion.eventgateway.Logger log = getGatewayServices().getLogger();
    Map data = new HashMap();
    CFEvent cfMsg = new CFEvent(getGatewayID());
    data.put("msg", msg);
    data.put("id", msgID);
    cfMsg.setData(data);
    cfMsg.setOriginatorID(topicName);
    cfMsg.setGatewayType("JMS");
    if (sendMessage(cfMsg)) {
            log.info("Added message '" + msgID + "' to queue.");
    } else {
            log.error("Failed to add message '" + msgID + "' to queue.");
    }
}
```

setOriginatorID

Description

Identifies the originator of an incoming message.

Category

Event Gateway Development

Syntax

```
void setOriginatorID(String originatorID)
```

See also

`getOriginatorID`, "CFML CFEvent structure" on page 1366, "CFEvent class" on page 1132 in *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`originatorID`	The gateway or protocol-specific ID of the message originator.

Example

The following code shows the routine from the example JMS gateway that handles incoming messages. It sets the originator ID to the name of the JMS topic that the gateway handles:

```
public void handleMessage(String msg, String topicName, String msgID) {
    coldfusion.eventgateway.Logger log = getGatewayServices().getLogger();
    Map data = new HashMap();
    CFEvent cfMsg = new CFEvent(getGatewayID());
    data.put("msg", msg);
    data.put("id", msgID);
    cfMsg.setData(data);
    cfMsg.setOriginatorID(topicName);
    cfMsg.setGatewayType("JMS");
    if (sendMessage(cfMsg)) {
            log.info("Added message '" + msgID + "' to queue.");
    } else {
            log.error("Failed to add message '" + msgID + "' to queue.");
    }
}
```

Logger class

`coldfusion.eventgateway.Logger`

Note: *This class is in the coldfusion.log package, not the coldfusion.eventgateway package, which contains all other event gateway-related interfaces and classes.*

The Logger class logs messages to a file in the ColdFusion logs directory. (You set this directory on the ColdFusion Administrator Logging Settings page.) The `coldfusion.eventgateway.GatewayServices.getLogger()` method returns an instance of the Logger class. The Logger class has the following methods:

Signature	Description
`debug`	Writes a debugging message to the log file.
`error`	Writes an error message to the log file.
`fatal`	Writes a fatal error to the log file.
`info`	Writes an informational message to the log file.
`warn`	Writes a warning message to the log file.

debug

Description

Writes a log entry with a debugging severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

```
debug(String message)
debug(Throwable th)
debug(String message, Throwable th)
```

See also

`error`, `fatal`, `info`, `warn`, `getLogger`, "Logging events and using log files" on page 1142 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`message`	The message to include in the log entry.
`th`	A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory.

Usage

Use this method to send a debugging message to the ColdFusion logging subsystem.

By default, ColdFusion does **not** write debugging messages to the log file. To have debug messages appear in the log file, change the priority entry in *cf_root*\lib\neo-logging.xml (in the server configuration) or *cf_root*/WEB-INF\cfusion\lib\neo-logging.xml (in the J2EE configuration. Change the following entry:

```
<var name='priority'>
    <string>information</string>
</var>
```

to the following:

```
<var name='priority'>
    <string>debug</string>
</var>
```

With debug priority, ColdFusion writes messages with a severity of "debug" to the log file specified in the `getLogger` method that returned the Logger instance (or the default log file).

Example

The ColdFusion instant messaging gateways use the following line to log information about incoming administrative messages or errors only when debugging priority is on.

```
// code to process incoming administrative messages or errors
logger.debug(gatewayType + "Gateway (" + gatewayID + ") admin message: " +
            msg.getMessage());
```

error

Description

Writes a log entry with an error severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

```
error(String message)
error(Throwable th)
error(String message, Throwable th)
```

See also

debug, fatal, info, warn, getLogger, "Logging events and using log files" on page 1142 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
message	The message to include in the log entry.
th	A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory.

Usage

Use this method to send an error message to the ColdFusion logging subsystem. ColdFusion writes messages with a severity of "error" to the log file specified in the getLogger method that returned the Logger instance (or the default log file).

Example

The ColdFusion example SocketGateway class includes the following code in the outgoingMessage method. It writes an error message if the message's originator ID does not correspond to an open socket.

```
SocketServerThread st =
        ((SocketServerThread)socketRegistry.get(cfmsg.getOriginatorID()));
    if(st != null)
        st.writeOutput(message);
    else {
        log.error("Cannot send outgoing message. OriginatorID '" +
            cfmsg.getOriginatorID() + "' is not a valid socket id.");
        retcode="failed";
    }
```

fatal

Description

Writes a log entry with a fatal severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

```
fatal(String message)
fatal(Throwable th)
fatal(String message, Throwable th)
```

See also

debug, error, info, warn, getLogger, "Logging events and using log files" on page 1142 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
message	The message to include in the log entry.
th	A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory.

Usage

Use this method to send a fatal error message to the ColdFusion logging subsystem. ColdFusion will write a messages with a severity of "fatal" to the log file specified in the getLogger method that returned the Logger instance (or the default log file).

info

Description

Writes a log entry with an information severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

```
info(String message)
info(Throwable th)
info(String message, Throwable th)
```

See also

debug, error, fatal, warn, getLogger, "Logging events and using log files" on page 1142 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
message	The message to include in the log entry.
th	A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory. Not normally used with this method.

Usage

Use this method to send an informational message to the ColdFusion logging subsystem. ColdFusion writes messages with a severity of "information" to the log file specified in the getLogger method that returned the Logger instance (or the default log file).

ColdFusion normally logs all information severity messages, so you should not use this severity for debugging messages or for events that happen frequently.

Example

The ColdFusion example DirectoryWatcherGateway class includes the following line at the top of its loadconfig method that loads the gateway's configuration file. It writes a message including the gateway ID and configuration file.

```
logger.info("DirectoryWatcher (" + gatewayID + ") Initializing
        DirectoryWatcher gateway with configuration file " + config);
```

warn

Description

Writes a log entry with a warning severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

```
warn(String message)
warn(Throwable th)
warn(String message, Throwable th)
```

See also

debug, error, fatal, info, getLogger, "Logging events and using log files" on page 1142 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
message	The message to include in the log entry.
th	A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory.

Usage

Use this method to send a warning message to the ColdFusion logging subsystem. ColdFusion writes messages with a severity of "warning" to the log file specified in the getLogger method that returned the Logger instance (or the default log file).

Example

The ColdFusion example SocketWatcherGateway class includes the following code in its constructor to load a configuration file. If it cannot load the file, it converts the exception information to a string and logs a warning that includes the gateway ID, and the exception information. It also passes the exception to the warn method

```
propsFilePath=configpath;
try {
    FileInputStream propsFile = new FileInputStream(propsFilePath);
    properties.load(propsFile);
    propsFile.close();
    this.loadProperties();
    }
catch (IOException e) {
    // do nothing. use default value for port.
    log.warn("SocketGateway(" + gatewayID + ") Unable to read configuration file
        " + propsFilePath + ": " + e.ToString() + ".Using default port.", e);
    }
```

CFML CFEvent structure

The CFML listener CFC methods receive messages in the form of a CFEvent structure that corresponds to the CFEvent class that gateway developers use. This structure has the following fields. Some of the fields might not be used by all gateways. All fields contain text or numeric values except the Data field, which contains a structure.

Field	Description
GatewayID	The event gateway that sent the event or will handle the outgoing message. The value is the ID of an event gateway instance configured on the ColdFusion Administrator Gateways page. If the application calls the `SendGatewayMessage` function to respond to the event gateway, it uses this ID as the function's first parameter.
Data	A structure containing the event data, including the message. The `Data` structure contents depend on the event gateway type. This field corresponds to the `SendGatewayMessage` function's second parameter.
OriginatorID	The originator of the message. The value depends on the protocol or event gateway type. Some event gateways might require this value in response messages to identify the destination of the response. Identifies the sender of the message.
GatewayType	The type of event gateway, such as SMS. An application that can process messages from multiple event gateway types can use this field. This value is the gateway type name that is specified by the event Gateway class. It is not necessarily the same as the gateway type name in the ColdFusion Administrator.
CFCPath	The location of the listener CFC. The listener CFC does not need to use this field.
CFCMethod	The listener method that ColdFusion invokes to process the event. The listener CFC does not need to use this field.
CFCTimeout	The time-out, in seconds, for the listener CFC to process the event request. The listener CFC does not need to use this field.

IM gateway methods and commands

The XMPP and IBM Sametime gateways implement CFC methods to receive messages, use the gatewayHelper object methods to manage the gateway, and use outgoing message commands to send messages. The following sections describe these methods and commands:

- IM Gateway CFC incoming message methods
- IM gateway message sending commands
- IM Gateway GatewayHelper class methods

IM Gateway CFC incoming message methods

You write the following CFC methods to handle incoming messages from an XMPP or Lotus Sametime instant messaging gateway.

Note: *The method names assume a default gateway configuration. ColdFusion lets you change the method names and disable event types in the gateway configuration file.*

Method	Message type
`onAddBuddyRequest`	Requests from other IM users to add the gateway ID as their buddy
`onAddBuddyResponse`	Responses from others to requests from your gateway to add them to your buddy lists. Also used by buddies to ask to be removed from your list.
`onBuddyStatus`	Online status information messages
`onIMServerMessage`	Error and administrative messages from the IM server
`onIncomingMessage`	Instant messages

onAddBuddyRequest

Description

Handles incoming requests for users to add the gateway user name as one of their buddies.

Syntax

```
onAddBuddyRequest(CFEvent)
```

See also

onIncomingMessage, onAddBuddyResponse, onBuddyStatus, onIMServerMessage

Parameters

The method must take one parameter, a CFEvent structure with the following fields:

Field	Description
gatewayType	Gateway type, either XMPP or SAMETIME
gatewayID	The ID of the gateway instance, as configured in ColdFusion Administrator
originatorID	The IM ID of the message originator
cfcMethod	This CFC method; by default, onAddBuddyRequest.
data.MESSAGE	The message that was sent with the request
data.SENDER	The sender's ID; identical to the originatorID field value
data.RECIPIENT	The recipient's ID, as specified in the gateway's configuration file
data.TIMESTAMP	The date and time when the message was sent

Returns

The function can optionally return a value to send a response message. The return structure must contain the following fields:

Field	Description
command	One of the following: • accept Accept the request to add you as a buddy. ColdFusion adds the user to the permit list of users that can get status information. • decline Deny request to add you as a buddy. ColdFusion adds the user to the deny list of users that can get status information. • noact Take no action. ColdFusion does not respond to the requestor.
buddyID	ID to which to send the message. Normally, the value of the CFEvent.data.SENDER field. Not used with the noact command.
reason	A text message describing the reason for the action. Not used with the noact command.

Example

The following example searches for the requested buddy's name in a data source and, if it finds a unique entry, adds the buddy and updates the buddy's status information in an Application scope buddyStatus structure. If it doesn't find the name, it declines the buddy. If there are multiple entries for the buddy name in the database, it tells the gateway not to respond. It logs all actions.

```
<cffunction name="onAddBuddyRequest">
    <cfargument name="CFEvent" type="struct" required="YES">
    <cfquery name="buddysearch" datasource="cfdocexamples">
```

```
        SELECT IM_ID
        FROM Employees
        WHERE IM_ID = '#CFEvent.Data.SENDER#'
    </cfquery>
    <cflock scope="APPLICATION" timeout="10" type="EXCLUSIVE">
        <cfscript>
            // If the name is in the DB once, accept; if it is missing, decline.
            // If it is in the DB multiple times, take no action.
            if (buddysearch.RecordCount IS 0) {
                action="decline";
                reason="Invalid ID";
            }
            else if (buddysearch.RecordCount IS 1) {
                action="accept";
                reason="Valid ID";
                //Add the buddy to the buddy status structure only if accepted.
                if (NOT StructKeyExists(Application,
                        "buddyStatus")) {
                    Application.buddyStatus=StructNew();
                }
                if (NOT StructKeyExists(Application.buddyStatus,
                        CFEvent.Data.SENDER)) {
                    Application.buddyStatus[#CFEvent.Data.SENDER#]=StructNew();
                }
                Application.buddyStatus[#CFEvent.Data.SENDER#].status=
                    "Accepted Buddy Request";
                Application.buddyStatus[#CFEvent.Data.SENDER#].timeStamp=
                    CFEvent.Data.TIMESTAMP;
                Application.buddyStatus[#CFEvent.Data.SENDER#].message=
                    CFEvent.Data.MESSAGE;
            }
            else {
                action="noact";
            }
        </cfscript>
    </cflock>
    <!--- Log the request and decision information. --->
    <cflog file="#CFEvent.GatewayID#Status"
            text="onAddBuddyRequest; SENDER: #CFEvent.Data.SENDER# MESSAGE:
#CFEvent.Data.MESSAGE# TIMESTAMP: #CFEvent.Data.TIMESTAMP# ACTION: #action#">
    <!--- Return the action decision. --->
    <cfset retValue = structNew()>
    <cfset retValue.command = action>
    <cfset retValue.BuddyID = CFEvent.DATA.SENDER>
    <cfset retValue.Reason = reason>
    <cfreturn retValue>
</cffunction>
```

onAddBuddyResponse

Description

Handles incoming responses from other users to requests from the gateway to be added to their buddy lists. Also receives requests from buddies to have you remove them from your buddy list.

Syntax

```
onAddBuddyResponse(CFEvent)
```

See also

onIncomingMessage, onAddBuddyRequest, onBuddyStatus, onIMServerMessage

Parameters

The method must take one parameter, a CFEvent structure with the following fields:

Field	Description
gatewayType	Gateway type, either XMPP or SAMETIME.
gatewayID	The ID of the gateway instance, as configured in ColdFusion Administrator.
originatorID	The IM ID of the message originator.
cfcMethod	This CFC method; by default, onAddBuddyResponse.
data.MESSAGE	One of the following: • accept The request was accepted. • decline The request was declined, or the buddy is asking you to remove them from your list.
data.SENDER	The sender's ID; identical to the originatorID.
data.RECIPIENT	The recipient's ID, as specified in the gateway's configuration file.
data.TIMESTAMP	The date and time when the message was sent.

Returns

The function does not return a value.

Example

The following example adds the buddy's status to the Application scope buddyStatus structure if the message sender accepted an add buddy request. It logs all responses.

```
<cffunction name="onAddBuddyResponse">
    <cfargument name="CFEvent" type="struct" required="YES">
    <cflock scope="APPLICATION" timeout="10" type="EXCLUSIVE">
        <cfscript>
            //Do the following only if the buddy accepted the request.
            if (NOT StructKeyExists(Application, "buddyStatus")) {
                Application.buddyStatus=StructNew();
                }
            if (#CFEVENT.Data.MESSAGE# IS "accept") {
                //Create a new entry in the buddyStatus record for the buddy.
                if (NOT StructKeyExists(Application.buddyStatus,
                        CFEvent.Data.SENDER)) {
                    Application.buddyStatus[#CFEvent.Data.SENDER#]=StructNew();
                }
                //Set the buddy status information to indicate buddy was added.
                Application.buddyStatus[#CFEvent.Data.SENDER#].status=
                    "Buddy accepted us";
```

```
                Application.buddyStatus[#CFEvent.Data.SENDER#].timeStamp=
                    CFEvent.Data.TIMESTAMP;
                Application.buddyStatus[#CFEvent.Data.SENDER#].message=
                    CFEvent.Data.MESSAGE;
            }
        </cfscript>
    </cflock>
    <!--- Log the information for all responses. --->
    <cflog file="#CFEvent.GatewayID#Status"
        text="onAddBuddyResponse; BUDDY: #CFEvent.Data.SENDER# RESPONSE:
#CFEvent.Data.MESSAGE# TIMESTAMP: #CFEvent.Data.TIMESTAMP#">
</cffunction>
```

onBuddyStatus

Description

Handles incoming messages indicating online status (presence) changes of users on the gateway's buddy list.

Syntax

```
onBuddyStatus(CFEvent)
```

See also

onIncomingMessage, onAddBuddyRequest, onAddBuddyResponse, onIMServerMessage

Parameters

The method must take one parameter, a CFEvent structure with the following fields:

Field	Description
gatewayType	Gateway type, either XMPP or SAMETIME.
gatewayID	The ID of the Gateway instance, as configured in ColdFusion Administrator.
originatorID	The IM ID (buddy name) of the message originator.
cfcMethod	This CFC method; by default, onIMServerMessage.
data.BUDDYNAME	The sender's buddy name, or ID; identical to the originatorID.
data.BUDDYNICKNAME	The buddy's display name or nickname.
data.BUDDYSTATUS	The buddy's status; one of the following: • ONLINE • OFFLINE • AWAY • DO NOT DISTURB **XMPP only** • NOT AVAILABLE • FREE TO CHAT **Sametime only** • IDLE Use the IMGatewayHelper getCustomAwayMessage method to get any custom message that the buddy sent when changing status.
data.BUDDYGROUP	The group that the buddy belongs to.
data.RECIPIENT	The recipient's ID, as specified in the gateway's configuration file.
data.TIMESTAMP	The date and time when the message was sent.

Note: *You configure the buddy's nickname and group when you use the gatewayHelper object* `addBuddy` *method to add a buddy.*

Returns

The function does not return a value.

Example

The following example keeps an Application scope structure up-to-date with a buddy's status. It also uses the gatewayhelper object `getBuddyStatus` method to get the buddy's custom away message, if any.

```
<cffunction name="onBuddyStatus">
    <cfargument name="CFEvent" type="struct" required="YES">
    <!--- Get the gatewayhelper object and to get the info for this buddy. --->
    <!--- This is used to get the buddy's custom away message. --->
    <cfset helper = getGatewayHelper("MYIM")>
    <cfset mybuddyinfo=helper.getBuddyInfo(CFEvent.Data.BUDDYNAME)>

    <cflog file="#CFEvent.GatewayID#Status" type="Information"
        text="in OnbuddyStatus, sender is #CFEvent.OriginatorID#">
    <cflock scope="APPLICATION" timeout="10" type="EXCLUSIVE">
        <cfscript>
        // Create the status structures if they don't exist.
            if (NOT StructKeyExists(Application, "buddyStatus")) {
                Application.buddyStatus=StructNew();
            }
            if (NOT StructKeyExists(Application.buddyStatus,
                    CFEvent.Data.BUDDYNAME)) {
                Application.buddyStatus[#CFEvent.Data.BUDDYNAME#]=StructNew();
            }
            // Save the buddy status, timestamp, and custom away message
            Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].status=
                CFEvent.Data.BUDDYSTATUS;
            Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].timeStamp=
                CFEvent.Data.TIMESTAMP;
            // The following assumes that the buddy is in only one group.
            Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].customAway=
                mybuddyinfo[1].BUDDYCUSTOMAWAYMESSAGE;
        </cfscript>
    </cflock>
    <!--- log the info, for debugging purposes only --->
    <cfset temp=Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].status>
    <cflog file="#CFEvent.GatewayID#Status" type="Information" text=
        "Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].status is #temp#">
    <cfset temp=Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].timeStamp>
        <cflog file="#CFEvent.GatewayID#Status" type="Information" text=
          "Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].timestamp is #temp#">
        <cflog file="#CFEvent.GatewayID#Status" type="Information" text=
            "Buddy Custom Away Message is mybuddyinfo[1].BUDDYCUSTOMAWAYMESSAGE#">
</cffunchtion>
```

onIMServerMessage

Description

Handles incoming error and status messages from the IM server.

Syntax

```
onIMServerMessage(CFEvent)
```

See also

onIncomingMessage, onAddBuddyRequest, onAddBuddyResponse, onBuddyStatus

Parameters

This method must take one parameter, a CFEvent structure with the following fields:

Field	Description
gatewayType	Gateway type, either XMPP or SAMETIME
gatewayID	The ID of the gateway instance, as configured in ColdFusion Administrator
originatorID	The IM ID (buddy name) of the message originator
cfcMethod	This CFC method; by default, onIMServerMessage
data.MESSAGE	The message sent by the server
data.SENDER	The sender's ID; identical to the originatorID
data.RECIPIENT	The recipient's ID, as specified in the gateway's configuration file
data.TIMESTAMP	The date and time when the message was sent

Example

The following example logs the sender, message, and a timestamp when an IM server sends an error or status message:

```
<cffunction name="onIMServerMessage">
    <!--- This function just logs the message. --->
    <cfargument name="CFEvent" type="struct" required="YES">
    <cflog file="#CFEvent.GatewayID#Status"
        text="onIMServerMEssage; SENDER: #CFEvent.OriginatorID#MESSAGE:
#CFEvent.Data.MESSAGE# TIMESTAMP: #CFEvent.Data.TIMESTAMP#">
</cffunction>
```

onIncomingMessage

Description

Handles incoming instant messages from other users. Optionally returns a response to the message sender.

Syntax

```
onIncomingMessage(CFEvent)
```

See also

onAddBuddyRequest, onAddBuddyResponse, onBuddyStatus, onIMServerMessage, "Handling incoming messages" on page 1089 in the *ColdFusion Developer's Guide*

Parameters

The method must take one parameter, a CFEvent structure with the following fields:

Field	Description
gatewayType	Gateway type, either XMPP or SAMETIME.
gatewayID	The ID of the Gateway instance as configured in ColdFusion Administrator.
originatorID	The IM ID of the message originator.
cfcMethod	This CFC method; by default, onIncomingMessage.
data.MESSAGE	The message that was received.
data.SENDER	The sender's ID; identical to the originatorID
data.RECIPIENT	The recipient's ID, as specified in the gateway's configuration file
data.TIMESTAMP	The date and time when the message was sent

Returns

The function can optionally return a value to send a response message. The return structure must contain the following fields:

Field	Description
command	Normally omitted. You can also specify submit.
buddyID	ID to which to send the message. Normally, the value of the input parameter's Data.SENDER field.
message	The message contents.

Example

The following example shows a simple onIncomingMessage method that echoes a message back to the sender.

```
<cffunction name="onIncomingMessage">
    <cfargument name="CFEvent" type="struct" required="YES">
    <cfset input_mesg = CFEvent.data.MESSAGE>
    <cfset retValue = structNew()>
    <cfset retValue.command = "submit">
    <cfset retValue.buddyID = CFEvent.originatorID>
    <cfset retValue.message = "Message Received:" & input_mesg>
    <cfreturn retValue>
</cffunction>
```

IM gateway message sending commands

You use the `SendGatewayMessage` CFML function or the return value of a CFC listener method to send outgoing messages. The ColdFusion MX 7 IM gateway accepts the following outgoing message commands:

Command	Description
`submit`	(Default) Sends a normal message to another IM user.
`accept`	Accepts an add buddy request. Adds the buddy to the list of IDs that get your presence information and sends an acceptance message to the buddy ID.
`decline`	Declines an add buddy request and sends a rejection message to the buddy ID.
`noact`	Tells the gateway to take no action. The gateway logs a message that indicates that it took no action, and contains the gateway type, gateway ID, and buddy ID.

The message structure that you return in the gateway listener CFC function or use as the second parameter in the CFML `SendGatewayMessage` function can have the following fields. The table lists the fields and the commands in which they are used, and describes the field's use.

Field	Commands	Description
buddyID	All	The destination user ID
command	All	The command; defaults to submit if omitted
message	submit	A text message to send to the destination user
reason	accept, decline	A text description of the reason for the action or other message to send to the add buddy requestor

In typical use, a ColdFusion application uses the accept, decline, and noact commands in the return value of the `onAddBuddyRequest` method, and uses the submit command (or no command, because submit is the default command) in `SendGatewayMessage` CFML functions and the return value of the `onIncomingMessage` CFC method.

IM Gateway GatewayHelper class methods

The GatewayHelper class returned by the CFML `GetGatewayHelper` function includes the following methods:

addBuddy	getDenyList	getStatusAsString	removeDeny
addDeny	getName	getStatusTimeStamp	removePermit
addPermit	getNickName	isOnline	setNickName
getBuddyInfo	getPermitList	numberOfMessagesReceived	setPermitMode
getBuddyList	getPermitMode	numberOfMessagesSent	setStatus
getCustomAwayMessage	getProtocolName	removeBuddy	

addBuddy

Description

Adds a buddy to the buddy list for the gateway user ID and asks to have the IM server send messages with the buddy's online presence state to the gateway.

Syntax

```
Boolean = addBuddy(name, nickname, group)
```

See also

`getBuddyInfo`, `getBuddyList`, `removeBuddy`, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`name`	The unique instant messaging user name for the person about whom you want to receive periodic status messages.
`nickname`	The nickname that the application can use to refer to the user.
`group`	The name of the group you wish to add the user to in your Buddy List. If the group specified does not exist, it will be created. If the group parameter is the empty string, the gateway uses the General group.

Returns

True if the ID was added to the gateway's buddy list; False, otherwise.

Usage

This method adds the buddy to the buddy list for the gateway's ID and sends a subscription request (to automatically get presence information about the buddy's online status) to the remote buddy. It does not wait for a response from the buddy, so it returns True (and the gateway adds the buddy to the list) even if the buddy denies the subscription request. Use the listener CFC `onAddBuddyResponse` method to monitor the buddy's response. If the CFEvent.data.MESSAGE field value is decline, the listener method can call the gatewayHelper object `removeBuddy` method to remove the buddy from the buddy list.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

addDeny

Description

Tells the IM server to add the specified user to the deny list for the gateway's user ID. If the gateway's permit mode value is DENY_SOME, the specified user cannot receive messages on the gateway's presence state.

Syntax

```
Boolean = addDeny(name, nickname, group)
```

See also

addPermit, getDenyList, getPermitList, getPermitMode, removeDeny, removePermit, setPermitMode, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
name	The unique instant messaging user name for the person about whom you want to deny access to status messages.
nickname	The nickname that the application can use to refer to the user. Can be the empty string.
group	The name of the group that you want to add the user to in your buddy list. If the group specified does not exist, it is created. If the group parameter is the empty string, the gateway uses the General group.

Returns

True if the ID was added to the deny list; False, otherwise.

Note: *XMPP permission management is included in the XMPP 1.0 draft specification, but several XMPP servers that were available at the time of the ColdFusion 8 release do not support permission management. If the server does not support permission management, this function always returns False*

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

addPermit

Description

Tells the IM server to add the specified user to the permit list for the gateway's user ID. If the gateway's permit mode is PERMIT_SOME, the specified user receive messages on the gateway's presence state.

Syntax

```
Boolean = addPermit(name, nickname, group)
```

See also

addDeny, getDenyList, getPermitList, getPermitMode, removeDeny, removePermit, setPermitMode, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
name	The unique instant messaging user name for the person about whom you want to deny access to status messages.
nickname	The nickname that the application can use to refer to the user. Can be the empty string.
group	The name of the group you want to add the user to in your Buddy List. If the group specified does not exist, it is created. If the group parameter is the empty string, the gateway uses the General group.

Returns

True if the ID was added to the permit list; false, otherwise.

Note: *XMPP permission management is included in the XMPP 1.0 draft specification, but several XMPP servers that were available at the time of the ColdFusion 8 release do not support permission management. If the server does not support permission management, this function always returns False.*

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getBuddyInfo

Description

Gets information about the specified user from the buddy list, deny list, and permit list.

Syntax

```
array = getBuddyInfo(name)
```

See also

addBuddy, getBuddyList, removeBuddy, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
name	The unique instant messaging user name for the person about whom you want to get information.

Returns

An array of structures, with one structure for each information record found. The method finds one record for each group that the user belongs to in each of the lists (buddy, permit, deny) that contains the specified name. Each structure has the following fields. Some fields might not be meaningful for some IM protocols. If there is no information for a field, it is blank.

Field	Description
BUDDYNAME	The user's unique ID.
BUDDYGROUP	The group to which the user belongs.
BUDDYNICKNAME	The nickname that you have assigned to the user.
BUDDYPROTOCOL	The instant messaging protocol. JABBER (for XMPP) or SAMETIME, or an empty string (if the server did not return a value).
BUDDYSTATUS	The user's presence state, can by any of the following: • ONLINE • OFFLINE • AWAY • DND (displays as DO NOT DISTURB) **XMPP only** • NA (displays as NOT AVAILABLE) • FREE_TO_CHAT (displays as FREE TO CHAT) **Sametime only** • IDLE
BUDDYSIGNONTIME	The date and time when the user signed onto the IM server. Empty if the user is not currently signed on. Always an empty string for XMPP and Sametime.
BUDDYSTATUSTIME	The date and time when the user's status most recently changed.
BUDDYCUSTOMAWAYMESSAGE	The custom away message that the user has set to explain the current status, if any.
BUDDYOWNER	A string representing the client and protocol associated with this ID, in the format *client@protocol*.

Field	Description
BUDDYLISTTIYPE	The type of list that this buddy record is in; one of the following: • BUDDY_LIST The list of users whose presence status information the gateway can receive. • DENY_LIST The list of users who cannot get presence information about the gateway ID. • PERMIT_LIST The list of users who can send presence information messages to the gateway ID. • REVERSE_LIST The list of users who do not allow messages to us.
BUDDYIDLETIME	If the buddy status is IDLE, how long the buddy has been idle. Always 0 for XMPP or SameTime.
BUDDYISMOBILE	True or False, indicating whether the user is on a mobile device. Always False for XMPP or SameTime.
BUDDYWARNINGPERCENT	The user's warning percentage value. Always 0 for XMPP or SameTime.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods. For an example of using this method to get the buddy custom away message, see `onBuddyStatus`.

getBuddyList

Description

Gets the buddy list for the gateway's user ID.

Syntax

```
array = getBuddyList()
```

See also

addBuddy, getBuddyInfo, removeBuddy, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

An array of IDs (buddy names) of the users on the gateway's buddy list, a list of instant messaging IDs that this gateway normally communicates with.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getCustomAwayMessage

Description

Returns the gateway's custom away message if it has been set by the gatewayHelper object `setStatus` method.

Syntax

```
string = getCustomAwayMessage()
```

See also

`getStatusAsString`, `getStatusTimeStamp`, `isOnline`, `setStatus`, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The gateway's custom away message if it has been set by the GatewayHelper object `setStatus` method.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getDenyList

Description

Returns the list of users that the IM server has been told not to send state information about the gateway, if the permit mode is set to DENY_SOME.

Syntax

```
array = getDenyList()
```

See also

addDeny, addPermit, getPermitList, getPermitMode, removeDeny, removePermit, setPermitMode, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

An array of IDs (buddy names) of the users on the gateway's deny list, the list of IDs to which the IM server does not send presence status information.

Note: *XMPP permission management is included in the XMPP 1.0 draft specification, but several XMPP servers that were available at the time of the ColdFusion 8 release do not support permission management. If the server does not support permission management, this function always returns False.*

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getName

Description

Returns the gateway's user name.

Syntax

```
string = getName()
```

See also

`getProtocolName`, `numberOfMessagesReceived`, `numberOfMessagesSent`, `setNickName`, "Using the Gateway-Helper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The gateway's user name, as specified in gateway configuration file.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getNickName

Description

Returns the gateway's nickname (display name), if it has been set using the gatewayHelper object `setNickName` method.

Syntax

```
string = getNickName()
```

See also

`getName`, `getProtocolName`, `numberOfMessagesReceived`, `numberOfMessagesSent`, `setNickName`, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The gateway's nickname, if any; empty string, otherwise.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getPermitList

Description

Returns the list of users that the IM server has been told to send state information about the gateway.

Syntax

```
array = getPermitList()
```

See also

addDeny, addPermit, getDenyList, getPermitMode, removeDeny, removePermit, setPermitMode, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

An array of IDs (buddy names) of the users on the gateway's permit list, the list of IDs to which the IM server sends presence status information if the permit mode is set to PERMIT_SOME.

Note: *XMPP permission management is included in the XMPP 1.0 draft specification, but several XMPP servers that were available at the time of the ColdFusion 8 release do not support permission management. If the server does not support permission management, this function always returns False.*

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getPermitMode

Description

Gets the gateway's permit mode from the IM server. The permit mode determines whether all users can get the gateway's online state information, or whether the server uses a permit list or a deny list to control which users get state information.

Syntax

```
string = getPermitMode()
```

See also

addDeny, addPermit, getDenyList, getPermitList, removeDeny, removePermit, setPermitMode, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The gateway's permit mode; one of the following values:

Mode	Description
PERMIT_ALL	(Default) Permits all users to be aware of the gateway's online presence and state.
PERMIT_SOME	Permits only users in the permit list to be aware of the gateway's online presence and state.
DENY_SOME	Prevents the users in the deny list from being aware of the gateway's online presence and state.

Note: *XMPP permission management is included in the XMPP 1.0 draft specification, but several XMPP servers that were available at the time of the ColdFusion 8 release do not support permission management. If the server does not support permission management, this function always returns PERMIT_ALL.*

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getProtocolName

Description

Gets the name of the gateway's instant messaging protocol.

Syntax

```
string = getProtocolName()
```

See also

getName, getNickName, numberOfMessagesReceived, numberOfMessagesSent, setNickName, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The gateway's protocol, as determined by the gateway type; one of the following values:

- JABBER (for XMPP)
- SAMETIME

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getStatusAsString

Description

Gets the online status of the gateway as a text string.

Syntax

```
string = getStatusAsString()
```

See also

`getCustomAwayMessage`, `getStatusTimeStamp`, `isOnline`, `setStatus`, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The gateway's online status; one of the following:

- ONLINE
- OFFLINE
- AWAY
- DO NOT DISTURB

XMPP only

- NOT AVAILABLE
- FREE TO CHAT

Sametime only

1 IDLE

Usage

The DO NOT DISTURB, NOT AVAILABLE, and FREE TO CHAT strings differ from the status values that you use in the `setStatus` method, which does not allow spaces in the status names.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

getStatusTimeStamp

Description

Gets the date and time that the gateway changed its online status.

Syntax

```
date-time object = getStatusTimeStamp()
```

See also

`getCustomAwayMessage`, `getStatusAsString`, `isOnline`, `setStatus`, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The date and time that the gateway changed its online status, normally by calling the `setStatus` gatewayHelper object method.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

isOnline

Description

Determines whether the gateway is connected to the instant messaging server.

Syntax

```
Boolean = isOnline()
```

See also

getCustomAwayMessage, getStatusAsString, getStatusTimeStamp, setStatus, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

True, if the gateway is connected to the IM server; False, otherwise.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

numberOfMessagesReceived

Description

Gets the number of messages received by the gateway since it was started.

Syntax

```
integer = numberOfMessagesReceived()
```

See also

getName, getNickName, getProtocolName, numberOfMessagesSent, setNickName, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The number of messages received by the gateway since it was started.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

numberOfMessagesSent

Description

Gets the number of messages sent by the gateway since it was started.

Syntax

```
integer = numberOfMessagesSent()
```

See also

`getName`, `getNickName`, `getProtocolName`, `numberOfMessagesReceived`, `setNickName`, "Using the Gateway-Helper object" on page 1095 in the *ColdFusion Developer's Guide*

Returns

The number of messages sent by the gateway since it was started.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

removeBuddy

Description

Removes an ID from a group in the buddy list for the gateway and tells the IM server not to send the gateway messages with the buddy's online presence state.

Syntax

```
Boolean = removeBuddy(name, group)
```

See also

`addBuddy`, `getBuddyInfo`, `getBuddyList`, `removeDeny`, `removePermit`, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`name`	The unique instant messaging user name for the person to remove from the buddy list.
`group`	The name of the group from which you want to remove the user. If the parameter is the empty string, the gateway uses the General group.

Returns

True if the ID was removed from the group; False, otherwise.

Usage

If the user is in multiple groups in your buddy list, you remove the buddy separately from each group. The IM server does not stop sending status updates until you remove the name from all groups.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

removeDeny

Description

Removes an ID from a group in the deny list for the gateway. If the gateway's permit mode is DENY_SOME, the specified user can receive messages on the gateway's presence state.

Syntax

```
Boolean = removeDeny(name, group)
```

See also

addDeny, addPermit, getDenyList, getPermitList, getPermitMode, removeBuddy, removePermit, setPermitMode, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
name	The unique instant messaging user name for the person to remove from the deny list.
group	The name of the group from which you want to remove the user. If the parameter is the empty string, the gateway uses the General group.

Returns

True if the ID was removed from the group; False, otherwise.

Note: *XMPP permission management is included in the XMPP 1.0 draft specification, but several XMPP servers that were available at the time of the ColdFusion 8 release do not support permission management. If the server does not support permission management, this function always returns False.*

Usage

If the user is in multiple groups in your deny list, you remove the user separately from each group. The IM server enables sending status updates if you remove the name any group.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

removePermit

Description

Removes an ID from a group in the permit list for the gateway. If the gateway's permit mode is PERMIT_SOME, the specified user cannot receive messages on the gateway's presence state.

Syntax

```
Boolean = removePermit(name, group)
```

See also

addDeny, addPermit, getDenyList, getPermitList, getPermitMode, removeBuddy, removeDeny, setPermitMode, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
name	The unique instant messaging user name for the person to remove from the permit list.
group	The name of the group from which you want to remove the user. If the parameter is the empty string, the gateway uses the General group.

Returns

True if the ID was removed from the group; False, otherwise.

Note: *XMPP permission management is included in the XMPP 1.0 draft specification, but several XMPP servers that were available at the time of the ColdFusion 8 release do not support permission management. If the server does not support permission management, this function always returns False.*

Usage

If the user is in multiple groups in your permit list, you remove the user separately from each group. However, the IM server stops sending status updates when you remove the user from the first group.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

setNickName

Description

Sets the gateway's nickname (display name).

Syntax

```
Boolean = setNickName(name)
```

See also

`getName`, `getNickName`, `getProtocolName`, `numberOfMessagesReceived`, `numberOfMessagesSent`, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`name`	The display name that you want to associate with this gateway. This name is not guaranteed to be unique for the protocol.

Returns

True if the nickname got set; false, otherwise.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

setPermitMode

Description

Sets the gateway's permit mode on the IM server. The permit mode determines whether all users can get the gateway's online state information, or whether the server uses a permit list or a deny list to control which users get state information.

Syntax

```
Boolean = setPermitMode(permitMode)
```

See also

addDeny, addPermit, getDenyList, getPermitList, getPermitMode, removeDeny, removePermit, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
permitMode	The permission mode, one of the following: • PERMIT_ALL Permits all users to be aware of the gateway's online presence and state. This is the default mode if you do not call this function. • PERMIT_SOME Permits only users in the permit list to be aware of the gateway's online presence and state. • DENY_SOME Prevents all users in the deny list from being aware of the gateway's online presence and state.

Returns

True if the permit mode was set; False otherwise.

Note: *XMPP permission management is included in the XMPP 1.0 draft specification, but several XMPP servers that were available at the time of the ColdFusion 8 release do not support permission management. If the server does not support permission management, this function returns False to all values except PERMIT_ALL.*

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

setStatus

Description

Sets the online presence status of the gateway, including any custom away message.

Syntax

```
Boolean = setStatus(status, customAwayMsg)
```

See also

`getCustomAwayMessage`, `getStatusAsString`, `getStatusTimeStamp`, `isOnline`, "Using the GatewayHelper object" on page 1095 in the *ColdFusion Developer's Guide*

Parameters

Parameter	Description
`status`	The gateway's online presence status; one of the following: • ONLINE • AWAY • DND (Do Not Disturb) **XMPP only** • NA (Not Available) • FREE_TO_CHAT **Sametime only:** • IDLE
`customAwayMsg`	A text string containing a custom message for the status. Can be the empty string if you do not need a custom away message.

Returns

True, if the operation was successful; False. otherwise. Passing an invalid status for the protocol causes this method to return False.

Usage

Do not use the `setStatus` method to go offline. Although the method accepts a parameter of OFFLINE, the gateway immediately resets itself to be online. To set the gateway offline, stop the gateway instance in the ColdFusion Administrator, or use the `stopGatewayInstance` method in the CFIDE.adminapi.eventgateway CFC.

Example

See "GatewayHelper example" on page 1096, in the *ColdFusion Developer's Guide*, which uses all GatewayHelper class methods.

SMS Gateway CFEvent structure and commands

This section describes the detailed contents of the following structures that you use in the SMS Gateway listener CFCs and CFML `SendGatewayMessage` functions:

- SMS Gateway incoming message CFEvent structure
- SMS gateway message sending commands

SMS Gateway incoming message CFEvent structure

The SMS gateway puts the following information in a CFEvent instance that it sends to the CFC listener method:

Field	Value
OriginatorID	Contents of the PDU `source_addr` field, the address of the device that sent the message.
CfcMethod	Listener CFC method name. Value of the configuration file `cfc-method` entry, or `onIncomingMessage` if the configuration file does not have this entry.
Data.MESSAGE	Contents of the `short_message` field of the PDU.
Data.sourceAddress	The address of the device that sent this message.
Data.destAddress	The address to which the message was sent; an address in the range specified by the gateway configuration file `address-range` setting.
Data.esmClass	Contents of the PDU esm_class field. Identifies the message type. A number in the range 0-255 representing a Byte value, where bits 2-5 (0-indexed) indicate the message type, and therefore the contents of the data.MESSAGE field, as follows. (Reserved values are omitted.) `xx0000xx` Normal message `xx0001xx` SMSC delivery receipt `xx0010xx` SME Delivery Acknowledgement `xx0100xx` SME Manual/User Acknowledgement `xx0110xx` Conversation abort (Korean CDMA only) `xx1000xx` Intermediate Delivery Notification For more information on this field, see the SMPP specification.
Data.protocol	Contents of the PDU protocol_id field. Meaningful for messages sent from GSM networks only. For more information, see the GSM 03.40 specification.
Data.priority	Contents of the PDU priority_flag field. A message priority level set by the originating SME, in the range 0-3; 0 is the lowest priority and 3 is the highest priority. The specific priority level meaning depends on the originating network. For more details, see the SMPP specification.
Data.registeredDelivery	Contents of the PDU registered_delivery field, indicating the type of delivery receipt or acknowledgement that the sender requested. A number in the range 0-32, representing the sum of the following values: 0 No SMS delivery receipt requested *or* 1 SMSC delivery receipt requested on delivery success or failure *or* 2 SMSC delivery receipt requested on delivery failure only Plus 0 No SME acknowledgement requested *or* 4 SME Delivery Acknowledgement requested *or* 8 SME Manual/User Acknowledgement requested *or* 12 Both Delivery and Manual/User Acknowledgements requested Plus 0 No Intermediate notification requested *or* 16 Intermediate notification requested

Field	Value
Data.DataCoding	Contents of the PDU data_coding field. Indicates the character set or the noncharacter data type of the message contents, as follows: `00000000` SMSC Default Alphabet `00000001` IA5 (CCITT T.50)/ASCII (ANSI X3.4) `00000010` Octet unspecified (8-bit binary) `00000011` Latin 1 (ISO-8859-1) `00000100` Octet unspecified (8-bit binary) `00000101` JIS (X 0208-1990) `00000110` Cyrillic (ISO-8859-5) `00000111` Latin/Hebrew (ISO-8859-8) `00001000` UCS2 (ISO/IEC-10646) `00001001` Pictogram Encoding `00001010` ISO-2022-JP (Music Codes) `00001101` Extended Kanji JIS(X 0212-1990) `00001110` KS C 5601 `11xxxxxx` GSM control use only; see the GSM 03.38 specification For more details, see the SMPP specification.
data.messageLength	The length of the short_message field.
GatewayType	Always SMS.

For more information on the meanings of some of these fields and how to handle incoming SMS messages an SMS gateway listener CFC method, see "Handling incoming messages" on page 1107 in the *ColdFusion Developer's Guide*.

SMS gateway message sending commands

ColdFusion applications that use gateways of the Short Message Service (SMS) type can send the following commands to the event gateway in an outgoing message:

- submit command
- submitMulti command
- data command

submit command

To send a message to a single destination address in an SMPP SUBMIT_SM PDU, the structure that you used in the *Data* parameter of a `SendGatewayMessage` function or the return variable of the CFC listener method has the following fields. For more information about these fields, see the documentation for the SUBMIT_MULTI PDU in the SMPP3.4 specification, which you can download from the SMS Forum at www.smsforum.net/.

Required fields

Field	Contents
command	If present, the value must be `submit`. If you omit this field, the event gateway sends a submit message.
shortMessage *or* messagePayload	The message contents. You must specify one of these fields, but not both. The SMPP specification imposes a maximum size of 254 bytes on the shortMessage field, and some carriers might limit its size further. The messagePayload field can contain up to 64K bytes; it must start with 0x0424, followed by two bytes specifying the payload length, followed by the message contents.
destAddress	Required. The address to which to send the message.
sourceAddress	The address of this application. You can omit this field; the configuration file specifies the application address.

Optional fields

You can set default values for the following optional fields in the SMS event gateway configuration file. For information on the default values, see "Configuring an SMS event gateway" on page 1105 in the *ColdFusion Developer's Guide*.

destAddress_npi	destAddress_ton	serviceType

The following optional fields do not have default values:

alertOnMsgDelivery	EsmClass	priorityFlag	smDefaultMsgId
callbackNum	ItsReplyType	PrivacyIndicator	SmsSignal
callbackNumAtag	ItsSessionInfo	protocolId	SourceAddrSubunit
callbackNumPresInd	LanguageIndicator	registeredDelivery	SourcePort
dataCoding	MoreMsgsToSend	replaceIfPresent	SourceSubaddress
DestAddrSubunit	MsMsgWaitFacilities	SarMsgRefNum	UserMessageReference
DestinationPort	MsValidity	SarSegmentSeqnum	UserResponseCode
DestSubaddress	NumberOfMessages	SarTotalSegments	UssdServiceOp
DisplayTime	PayloadType	scheduleDeliveryTime	validityPeriod

Example

The following example `onIncomingMessage` method of a listener CFC uses the submit command to echo incoming SMS messages to the message originator:

```
<cffunction name="onIncomingMessage" output="no">
    <cfargument name="CFEvent" type="struct" required="yes">
    <!--- Create a return structure that contains the message. --->
    <cfset retValue = structNew()>
    <cfset retValue.command = "submit">
    <cfset retValue.destAddress = arguments.CFEvent.originatorid>
    <cfset retValue.shortMessage = "Echo: " & CFEvent.Data.MESSAGE>
    <!--- Send the message back. --->
    <cfreturn retValue>
```

```
</cffunction>
```

submitMulti command

To send a single text message to multiple recipients using an SMPP SUBMIT_MULTI PDU, the *Data* parameter of a `SendGatewayMessage` function or the return variable of the CFC listener method usually has the following fields. For more information about these fields, see the documentation for the SUBMIT_MULTI PDU in the SMPP3.4 specification, which you can download from the SMS Forum at www.smsforum.net/.

Required fields

Field	Contents
command	Must be `submitMulti`.
shortMessage *or* messagePayload	The message contents. You must specify one of these fields, but not both. The SMPP specification imposes a maximum size of 254 bytes on the shortMessage field, and some carriers might limit its size further. The messagePayload field can contain up to 64K bytes; it must start with 0x0424, followed by two bytes specifying the payload length, followed by the message contents.
destAddress	A ColdFusion array of destination addresses (required). You cannot specify individual TON and NPI values for these addresses; all must conform to a single setting.
sourceAddress	The address of this application. You can omit this field; the configuration file specifies the application address.

Optional fields

The following optional fields can have default values set in the SMS event gateway configuration file. For information on the default values see “Configuring an SMS event gateway” on page 1105 in the *ColdFusion Developer’s Guide*.

destAddress_npi	destAddress_ton	serviceType

The following optional fields do not have default values:

alertOnMsgDelivery	DisplayTime	protocolId	SmsSignal
callbackNum	EsmClass	registeredDelivery	SourceAddrSubunit
callbackNumAtag	LanguageIndicator	replaceIfPresent	SourcePort
callbackNumPresInd	MsMsgWaitFacilities	SarMsgRefNum	SourceSubaddress
dataCoding	MsValidity	SarSegmentSeqnum	UserMessageReference
DestAddrSubunit	PayloadType	SarTotalSegments	validityPeriod
DestinationPort	priorityFlag	scheduleDeliveryTime	
DestSubaddress	PrivacyIndicator	smDefaultMsgId	

Example

The following example `onIncomingMessage` method sends a response that echoes an incoming message to the originator address, and sends a copy of the response to a second address:

```
<cffunction name="onIncomingMessage" output="no">
    <cfargument name="CFEvent" type="struct" required="yes">
    <!--- Get the message. --->
    <cfset data=cfevent.DATA>
    <cfset message="#data.message#">
    <!--- Create the return structure. --->
    <cfset retValue = structNew()>
    <cfset retValue.command = "submitmulti">
    <cfset retValue.destAddresses=arraynew(1)>
    <!--- One destination is incoming message originator;
```

```
                    get the address from CFEvent originator ID. --->
    <cfset retValue.destAddresses[1] = arguments.CFEvent.originatorid>
    <cfset retValue.destAddresses[2] = "12345">
    <cfset retValue.shortMessage = "echo: " & message>
<cfreturn retValue>
</cffunction>
```

data command

To send binary data to a single destination address in an SMPP DATA_SM PDU, the *Data* parameter of a `SendGatewayMessage` function or the return variable of the CFC listener method must have the following fields. For more information about these fields, see the documentation for the SUBMIT_MULTI PDU in the SMPP3.4 specification, which you can download from the SMS Forum at www.smsforum.net/.

Required fields

Field	Contents
command	Must be `data`.
messagePayload	The message data. To convert data to binary format, use the ColdFusion `ToBinary` function.
destAddress	The address to which to send the message.
sourceAddress	The address of this application. You can omit this field; the configuration file specifies the application address.

Optional fields

The following optional fields can have default values set in the SMS event gateway configuration file. For information on the default values see "Configuring an SMS event gateway" on page 1105 in the *ColdFusion Developer's Guide*.

destAddress_npi	destAddress_ton	serviceType

The following optional fields do not have default values:

alertOnMsgDelivery	DestTelematicsId	NetworkErrorCode	SetDpf
callbackNum	DisplayTime	NumberOfMessages	SmsSignal
callbackNumAtag	EsmClass	PayloadType	SourceAddrSubunit
callbackNumPresInd	ItsReplyType	PrivacyIndicator	SourceBearerType
dataCoding	ItsSessionInfo	QosTimeToLive	SourceNetworkType
DestAddrSubunit	LanguageIndicator	ReceiptedMessgeId	SourcePort
DestBearerType	MessageState	registeredDelivery	SourceSubaddress
DestNetworkType	MoreMsgsToSend	SarMsgRefNum	SourceTelematicsId
DestinationPort	MsMsgWaitFacilities	SarSegmentSeqnum	UserMessageReference
DestSubaddress	MsValidity	SarTotalSegments	UserResponseCode

Example

The following example `onIncomingMessage` method converts an incoming message to binary data, and sends the binary version of the message back to the originator address:

```
<cffunction name="onIncomingMessage" output="no">
    <cfargument name="CFEvent" type="struct" required="yes">
    <!--- Get the message --->
    <cfset data=CFEvent.DATA>
    <cfset message="#data.message#">
    <!--- Create the return structure --->
    <cfset retValue = structNew()>
    <cfset retValue.command = "data">
    <!--- Sending to incoming message originator; get value from CFEvent. --->
    <cfset retValue.destAddress = arguments.CFEvent.originatorid>
    <cfset retValue.messagePayload = tobinary(tobase64("echo: " & message))>
    <cfreturn retValue>
```

```
</cffunction>
```

CFML event gateway SendGatewayMessage data parameter

The ColdFusion CFML gateway type enables you to invoke CFC methods asynchronously. The structure that you use in the `SendGatewayMessage` function *data* parameter can include two types of fields:

- Any number of fields can contain arbitrary contents for use in by the CFC.
- Several optional fields can configure how the gateway delivers the information to the CFC.

The CFML gateway looks for the following optional fields, and, if they exist, uses them to determine how it delivers the message. Do not use these field names for data that you send to your CFC method.

Field	Use
cfcpath	Overrides the CFC path specified in the ColdFusion Administrator. This field lets you use a single gateway configuration in the ColdFusion Administrator multiple CFCs. This field sets the CFEvent object CFCPath variable.
method	Specifies the name of the method to invoke in the CFC. The default method is `onIncomingMessage`. This field lets you use a single gateway configuration in the ColdFusion Administrator for a CFC that has several methods. This field sets the CFEvent object CFCMethod variable.
originatorID	Sets the originatorID field of the CFEvent object that ColdFusion delivers to the CFC. The default value is CFMLGateway.
timeout	Sets the time-out, in seconds, during which the listener CFC must process the event request and return before ColdFusion gateway services terminates the request. The default value is the Timeout Request value set on the Server Settings page in the ColdFusion Administrator. Set this value if a request might validly take longer to process than the default time-out; for example, if the request involves a very long processing time. This field sets the CFEvent object CFCTimeout variable.

Example

The following example consists of a CFML page that sends a message to a logevent method in the file logger.CFC. The CFML page specifies the CFC and method to call, and sets the OriginatorID.

```
<h3>Sending an event using a generic CFML event gateway and specifying the CFC and
method.</h3>
<cfscript>
    status = False;
    props = structNew();
    props.cfcpath="C:\CFusionMX7\gateway\cfc\MyCFCs\logger.cfc";
    props.method="logEvent";
    props.OriginatorID=CGI.SCRIPT_NAME;
    props.Message="Replace me with a variable with data to log";
    props.file="GenericCFCtest";
    props.type="warning";
    status = SendGatewayMessage("DefaultCFC", props);
    if (status IS True)
        WriteOutput('Event Message "#props.Message#" has been sent.');
</cfscript>
```

The CFC method uses the OriginatorID and the message, file, and type fields of the CFEvent parameter's data field to specify the log file and message.

```
<cfcomponent>
    <cffunction name="logEvent" output="no">
        <cfargument name="CFEvent" type="struct" required="yes">
         <cfscript>
            if (NOT IsDefined("CFEvent.Data.file")) {
                CFEvent.Data.file="defaultEventLog"; }
            if (NOT IsDefined("CFEvent.Data.type")) {
```

```
                CFEvent.Data.type="information"; }
        </cfscript>
        <cflog text="Message from #CFEvent.originatorID#: #CFEvent.Data.message#"
            file="#CFEvent.data.file#" type="#CFEvent.Data.type#" >
    </cffunction>
</cfcomponent>
```

Chapter 9: ColdFusion C++ CFX Reference

ColdFusion includes CFXAPI classes and methods for building ColdFusion extensions.

Contents

C++ class overview

The following table lists the CFXAPI classes and methods:

Class	Methods
CCFXException class	CCFXException::GetError CCFXException::GetDiagnostics
CCFXQuery class	CCFXQuery::AddRow CCFXQuery::GetColumns CCFXQuery::GetData CCFXQuery::GetName CCFXQuery::GetRowCount CCFXQuery::SetData
CCFXRequest class	CCFXRequest::AddQuery CCFXRequest::AttributeExists CCFXRequest::CreateStringSet CCFXRequest::Debug CCFXRequest::GetAttribute CCFXRequest::GetAttributeList CCFXRequest::GetCustomData CCFXRequest::GetQuery CCFXRequest::ReThrowException CCFXRequest::SetCustomData CCFXRequest::SetVariable CCFXRequest::ThrowException CCFXRequest::Write CCFXRequest::WriteDebug
CCFXStringSet class	CCFXStringSet::AddString CCFXStringSet::GetCount CCFXStringSet::GetIndexForString CCFXStringSet::GetString

Deprecated class methods

The following CFXAPI classes and methods are deprecated. They do not work, and might cause an error, in later releases.

Class	Deprecated member	Deprecated as of this ColdFusion release
CCFXQuery Class	CCFXQuery::SetQueryString	ColdFusion MX
	CCFXQuery::SetTotalTime	ColdFusion MX
CCFXRequest Class	CCFXRequest::GetSetting	ColdFusion MX

CCFXException class

An abstract class that represents an exception thrown during processing of a ColdFusion Extension (CFX) procedure.

Exceptions of this type can be thrown by `CCFXRequest class`, `CCFXQuery class`, and `CCFXStringSet class`. Your ColdFusion Extension code must be written to handle exceptions of this type. For more information, see `CCFXRequest::ThrowException` and `CCFXRequest::ReThrowException`.

Class methods

`virtual LPCSTR GetError()`	The `CCFXException::GetError` function returns a general error message.
`virtual LPCSTR GetDiagnostics()`	The `CCFXException::GetDiagnostics` function returns detailed error information.

CCFXException::GetError

Description

Provides basic user output for exceptions that occur during processing.

CCFXException::GetDiagnostics

Description

Provides detailed user output for exception that occur during processing.

Example

This code block shows how GetError and GetDiagnostics work with ThrowException and ReThrowException.

```
// Write output back to the user here...
    pRequest->Write( "Hello from CFX_FOO2!" ) ;
    pRequest->ThrowException("User Error", "You goof'd...");

    // Output optional debug info
    if ( pRequest->Debug() )
    {
                        pRequest->WriteDebug( "Debug info..." ) ;
    }

    // Catch ColdFusion exceptions & re-raise them
    catch( CCFXException* e )
    {
                        // This is how you would pull the error information
                        LPCTSTR strError = e->GetError();
                        LPCTSTR strDiagnostic = e->GetDiagnostics();

                        pRequest->ReThrowException( e ) ;
    }

    // Catch ALL other exceptions and throw them as
    // ColdFusion exceptions (DO NOT REMOVE! --
    // this prevents the server from crashing in
    // case of an unexpected exception)
    catch( ... )
    {
```

```
                    pRequest->ThrowException(
                    "Error occurred in tag CFX_FOO2",
                    "Unexpected error occurred while processing tag." ) ;
}
```

CCFXQuery class

An abstract class that represents a query used or created by a ColdFusion Extension (CFX). Queries contain one or more columns of data that extend over a varying number of rows.

Class methods

`virtual int AddRow()`	`CCFXQuery::AddRow` adds a row to a query.
`virtual CCFXStringSet* GetColumns`	`CCFXQuery::GetColumns` retrieves a list of a query's column names.
`virtual LPCSTR GetData( int iRow, int iColumn )`	`CCFXQuery::GetData` retrieves a data element from a row and column of a query.
`virtual LPCSTR GetName()`	`CCFXQuery::GetName` retrieves the name of a query.
`virtual int GetRowCount()`	`CCFXQuery::GetRowCount` retrieves the number of rows in a query.
`virtual void SetData( int iRow, int iColumn, LPCSTR lpszData )`	`CCFXQuery::SetData` sets a data element within a row and column of a query.
`virtual void SetQueryString( LPCSTR lpszQuery )`	This function is deprecated. It might not work, and might cause an error, in later releases.
`virtual void SetTotalTime( DWORD dwMilliseconds )`	This function is deprecated. It might not work, and might cause an error, in later releases.

CCFXQuery::AddRow

Syntax

```
int CCFXQuery::AddRow(void)
```

Description

Add a row to the query. Call this function to append a row to a query.

Returns

Returns the index of the row that was appended to a query.

Example

The following example shows the addition of two rows to a three-column ('City', 'State', and 'Zip') query:

```
// First row

    int iRow ;
iRow = pQuery->AddRow() ;
    pQuery->SetData( iRow, iCity, "Minneapolis" ) ;
    pQuery->SetData( iRow, iState, "MN" ) ;
    pQuery->SetData( iRow, iZip, "55345" ) ;

    // Second row
    iRow = pQuery->AddRow() ;
    pQuery->SetData( iRow, iCity, "St. Paul" ) ;
    pQuery->SetData( iRow, iState, "MN" ) ;
    pQuery->SetData( iRow, iZip, "55105" ) ;
```

CCFXQuery::GetColumns

Syntax

```
CCFXStringSet* CCFXQuery::GetColumns(void)
```

Description

Retrieves a list of the column names contained in a query.

Returns

Returns an object of `CCFXStringSet class` that contains a list of the columns in the query. ColdFusion automatically frees the memory that is allocated for the returned string set, after the request is completed.

Example

The following example gets the list of columns, then iterates over the list, writing each column name back to the user:

```
// Get the list of columns from the query
    CCFXStringSet* pColumns = pQuery->GetColumns() ;
    int nNumColumns = pColumns->GetCount() ;

    // Print the list of columns to the user
    pRequest->Write( "Columns in query: " ) ;
    for( int i=1; i<=nNumColumns; i++ )
    {
        pRequest->Write( pColumns->GetString( i ) ) ;
        pRequest->Write( " " ) ;
    }
```

CCFXQuery::GetData

Syntax

```
LPCSTR CCFXQuery::GetData(int iRow, int iColumn)
```

Description

Gets a data element from a row and column of a query. Row and column indexes begin with 1. You can determine the number of rows in a query by calling `CCFXQuery::GetRowCount`. You can determine the number of columns in a query by retrieving the list of columns using `CCFXQuery::GetColumns`, and then calling `CCFXStringSet::GetCount` on the returned string set.

Returns

Returns the value of the requested data element.

Parameters

Parameter	Description
`iRow`	Row to retrieve data from (1-based)
`iColumn`	Column to retrieve data from (1-based)

Example

The following example iterates over the elements of a query and writes the data in the query back to the user in a simple, space-delimited format:

```
int iRow, iCol ;
    int nNumCols = pQuery->GetColumns()->GetCount() ;
```

```
    int nNumRows = pQuery->GetRowCount() ;
    for ( iRow=1; iRow<=nNumRows; iRow++ )
    {
        for ( iCol=1; iCol<=nNumCols; iCol++ )
        {
        pRequest->Write( pQuery->GetData( iRow, iCol ) ) ;
        pRequest->Write( " " ) ;
        }
        pRequest->Write( "<BR>" ) ;
    }
```

CCFXQuery::GetName

Syntax

```
LPCSTR CCFXQuery::GetName(void)
```

Description

Returns the name of a query.

Example

The following example retrieves the name of a query and writes it back to the user:

```
CCFXQuery* pQuery = pRequest->GetQuery() ;
    pRequest->Write( "The query name is: " ) ;
    pRequest->Write( pQuery->GetName() ) ;
```

CCFXQuery::GetRowCount

Syntax

```
int CCFXQuery::GetRowCount(void)
```

Description

Returns the number of rows contained in a query.

Example

The following example retrieves the number of rows in a query and writes it back to the user:

```
CCFXQuery* pQuery = pRequest->GetQuery() ;
    char buffOutput[256] ;
    wsprintf( buffOutput,
        "The number of rows in the query is %ld.",
        pQuery->GetRowCount() ) ;
    pRequest->Write( buffOutput ) ;
```

CCFXQuery::SetData

Syntax

```
void CCFXQuery::SetData(int iRow, int iColumn, LPCSTR lpszData)
```

Description

Sets a data element within a row and column of a query. Row and column indexes begin with 1. Before calling `SetData` for a given row, call `CCFXQuery::AddRow` and use the return value as the row index for your call to `SetData`.

Parameters

Parameter	Description
iRow	Row of data element to set (1-based)
iColumn	Column of data element to set (1-based)
lpszData	New value for data element

Example

The following example shows the addition of two rows to a three-column ('City', 'State', and 'Zip') query:

```
// First row
    int iRow ;
    iRow = pQuery->AddRow() ;
    pQuery->SetData( iCity, iRow, "Minneapolis" ) ;
    pQuery->SetData( iState, iRow, "MN" ) ;
    pQuery->SetData( iZip, iRow, "55345" ) ;

    // Second row
    iRow = pQuery->AddRow() ;
    pQuery->SetData( iCity, iRow, "St. Paul" ) ;
    pQuery->SetData( iState, iRow, "MN" ) ;
    pQuery->SetData( iZip, iRow, "55105" ) ;
```

CCFXRequest class

Abstract class that represents a request made to a ColdFusion Extension (CFX). An instance of this class is passed to the main function of your extension DLL. The class provides interfaces that can be used by the custom extension for the following actions:

- Reading and writing variables
- Returning output
- Creating and using queries
- Throwing exceptions

Class methods

`virtual BOOL AttributeExists( LPCSTR lpszName )`	`CCFXRequest::AttributeExists` checks whether the attribute was passed to the tag.
`virtual LPCSTR GetAttribute( LPCSTR lpszName )`	`CCFXRequest::GetAttribute` gets the value of the passed attribute.
`virtual CCFXStringSet* GetAttributeList()`	`CCFXRequest::GetAttributeList` gets an array of attribute names passed to the tag.
`virtual CCFXQuery* GetQuery()`	`CCFXRequest::GetQuery` gets the query that was passed to the tag.
`virtual LPCSTR GetSetting( LPCSTR lpszSettingName )`	`CCFXRequest::GetSetting` This method is deprecated. It might not work, and might cause an error, in later releases.
`virtual void Write( LPCSTR lpszOutput )`	`CCFXRequest::Write` writes text output back to the user.
`virtual void SetVariable( LPCSTR lpszName, LPCSTR lpszValue )`	`CCFXRequest::SetVariable` sets a variable in the template that contains this tag.
`virtual CCFXQuery* AddQuery( LPCSTR lpszName, CCFXStringSet* pColumns )`	`CCFXRequest::AddQuery` adds a query to the template that contains this tag.
`virtual BOOL Debug()`	`CCFXRequest::Debug` checks whether the tag contains the `Debug` attribute.
`virtual void WriteDebug( LPCSTR lpszOutput )`	`CCFXRequest::WriteDebug` writes text output into the debug stream.
`virtual CCFXStringSet* CreateStringSet()`	`CCFXRequest::CreateStringSet` allocates and returns a `CCFXStringSet` instance.
`virtual void ThrowException( LPCSTR lpszError, LPCSTR lpszDiagnostics )`	`CCFXRequest::ThrowException` throws an exception and ends processing of this request.
`virtual void ReThrowException( CCFXException* e )`	`CCFXRequest::ReThrowException` re-throws an exception that has been caught.
`virtual void SetCustomData( LPVOID lpvData )`	`CCFXRequest::SetCustomData` sets custom (tag specific) data to carry with a request.
`virtual LPVOID GetCustomData()`	`CCFXRequest::GetCustomData` gets custom (tag specific) data for a request.

CCFXRequest::AddQuery

Syntax

```
CCFXQuery* CCFXRequest::AddQuery(LPCSTR lpszName, CCFXStringSet* pColumns)
```

Description

Adds a query to the calling template. The query can be accessed by CFML tags (for example, `cfoutput` or `cftable`) within the template. After calling `AddQuery`, the query is empty (it has 0 rows). To populate the query with data, call the `CCFXQuery::AddRow` and `CCFXQuery::SetData` functions.

Returns

Returns a pointer to the query that was added to the template (an object of class `CCFXQuery`). The memory allocated for the returned query is freed automatically by ColdFusion after the request is completed.

Parameters

Parameter	Description
`lpszName`	Name of query to add to the template (must be unique)
`pColumns`	List of column names to be used in the query

Example

The following example adds a query named 'People' to the calling template. The query has two columns ('FirstName' and 'LastName') and two rows:

```
// Create a string set and add the column names to it
    CCFXStringSet* pColumns = pRequest->CreateStringSet() ;
    int iFirstName = pColumns->AddString( "FirstName" ) ;
    int iLastName = pColumns->AddString( "LastName" ) ;

    // Create a query that contains these columns
    CCFXQuery* pQuery = pRequest->AddQuery( "People", pColumns ) ;

    // Add data to the query
    int iRow ;
    iRow = pQuery->AddRow() ;
    pQuery->SetData( iRow, iFirstName, "John" ) ;
    pQuery->SetData( iRow, iLastName, "Smith" ) ;
    iRow = pQuery->AddRow() ;
    pQuery->SetData( iRow, iFirstName, "Jane" ) ;
    pQuery->SetData( iRow, iLastName, "Doe" ) ;
```

CCFXRequest::AttributeExists

Syntax

```
BOOL CCFXRequest::AttributeExists(LPCSTR lpszName)
```

Description

Checks whether the parameter was passed to the tag. Returns True if the parameter is available; False, otherwise.

Parameters

Parameter	Description
`lpszName`	Name of the parameter to check (case insensitive)

Example

The following example checks whether the user passed an attribute named DESTINATION to the tag, and throws an exception if the attribute was not passed:

```
if ( pRequest->AttributeExists("DESTINATION")==FALSE )
    {
        pRequest->ThrowException(
            "Missing DESTINATION parameter",
            "You must pass a DESTINATION parameter in "
            "order for this tag to work correctly." ) ;
    }
```

CCFXRequest::CreateStringSet

Syntax

```
CCFXStringSet* CCFXRequest::CreateStringSet(void)
```

Description

Allocates and returns an instance. Always use this function to create string sets, as opposed to directly using the new operator.

Returns

Returns an object of CCFXStringSet class. The memory allocated for the returned string set is freed automatically by ColdFusion after the request is completed

Example

The following example creates a string set and adds three strings to it:

```
CCFXStringSet* pColors = pRequest->CreateStringSet() ;
    pColors->AddString( "Red" ) ;
    pColors->AddString( "Green" ) ;
    pColors->AddString( "Blue" ) ;
```

CCFXRequest::Debug

Syntax

```
BOOL CCFXRequest::Debug(void)
```

Description

Checks whether the tag contains the Debug attribute. Use this function to determine whether to write debug information for a request. For more information, see CCFXRequest::WriteDebug.

Returns

Returns True if the tag contains the Debug attribute; False, otherwise.

Example

The following example checks whether the Debug attribute is present, and if it is, it writes a brief debug message:

```
if ( pRequest->Debug() )
    {
                        pRequest->WriteDebug( "Top secret debug info" ) ;
    }
```

CCFXRequest::GetAttribute

Syntax

```
LPCSTR CCFXRequest::GetAttribute(LPCSTR lpszName)
```

Description

Retrieves the value of the passed attribute. Returns an empty string if the attribute does not exist. (To test whether an attribute was passed to the tag, use `CCFXRequest::AttributeExists`.)

Returns

Returns the value of the attribute passed to the tag. If no attribute of that name was passed to the tag, an empty string is returned.

Parameters

Parameter	Description
`lpszName`	Name of the attribute to retrieve (case insensitive)

Example

The following example retrieves an attribute named `DESTINATION` and writes its value back to the user:

```
LPCSTR lpszDestination = pRequest->GetAttribute("DESTINATION") ;
    pRequest->Write( "The destination is: " ) ;
    pRequest->Write( lpszDestination ) ;
```

CCFXRequest::GetAttributeList

Syntax

```
CCFXStringSet* CCFXRequest::GetAttributeList(void)
```

Description

Gets an array of attribute names passed to the tag. To get the value of one attribute, use `CCFXRequest::GetAttribute`.

Returns

Returns an object of class `CCFXStringSet class` that contains a list of attributes passed to the tag. The memory allocated for the returned string set is freed automatically by ColdFusion after the request is completed.

Example

The following example gets the list of attributes and iterates over the list, writing each attribute and its value back to the user.

```
LPCSTR lpszName, lpszValue ;
    CCFXStringSet* pAttribs = pRequest->GetAttributeList() ;
    int nNumAttribs = pAttribs->GetCount() ;

    for( int i=1; i<=nNumAttribs; i++ )
    {
                        lpszName = pAttribs->GetString( i ) ;
                        lpszValue = pRequest->GetAttribute( lpszName ) ;
                        pRequest->Write( lpszName ) ;
                    pRequest->Write( " = " ) ;
                        pRequest->Write( lpszValue ) ;
                        pRequest->Write( "<BR>" ) ;
```

```
    }
```

CCFXRequest::GetCustomData

Syntax

```
LPVOID CCFXRequest::GetCustomData(void)
```

Description

Gets the custom (tag specific) data for the request. This method is typically used from within subroutines of a tag implementation to extract tag data from a request.

Returns

Returns a pointer to the custom data, or NULL if no custom data has been set during this request using CCFXRequest::SetCustomData.

Example

The following example retrieves a pointer to a request specific data structure of hypothetical type MYTAGDATA:

```
void DoSomeGruntWork( CCFXRequest* pRequest )
    {
            MYTAGDATA* pTagData =
                (MYTAGDATA*)pRequest->GetCustomData() ;

            ... remainder of procedure ...
    }
```

CCFXRequest::GetQuery

Syntax

```
CCFXQuery* CCFXRequest::GetQuery(void)
```

Description

Retrieves a query that was passed to a tag. To pass a query to a custom tag, you use the QUERY attribute. This attribute should be set to the name of a query (created using the cfquery tag or another custom tag). The QUERY attribute is optional and should be used only by tags that process an existing data set.

Returns

Returns an object of the CCFXQuery class that represents the query passed to the tag. If no query was passed to the tag, NULL is returned. The memory allocated for the returned query is freed automatically by ColdFusion after the request is completed.

Example

The following example retrieves the query that was passed to the tag. If no query was passed, an exception is thrown:

```
CCFXQuery* pQuery = pRequest->GetQuery() ;
    if ( pQuery == NULL )
    {
                        pRequest->ThrowException(
                            "Missing QUERY parameter",
                            "You must pass a QUERY parameter in "
                            "order for this tag to work correctly." ) ;
    }
```

CCFXRequest::ReThrowException

Syntax

```
void CCFXRequest::ReThrowException(CCFXException* e)
```

Description

Re-throws an exception that has been caught within an extension procedure. This function is used to avoid having C++ exceptions that are thrown by DLL extension code propagate back into ColdFusion. Catch ALL C++ exceptions that occur in extension code, and either re-throw them (if they are of the `CCFXException class`) or create and throw a new exception pointer using `CCFXRequest::ThrowException`.

Parameters

Parameter	Description
`e`	A CCFXException that has been caught

Example

The following code demonstrates how to handle exceptions in ColdFusion Extension DLL procedures:

```
try
    {
            ...Code that could throw an exception...
    }
    catch( CCFXException* e )
    {
            ...Do appropriate resource cleanup here...
            // Re-throw the exception
            pRequest->ReThrowException( e ) ;
    }
    catch( ... )
    {
            // Something nasty happened

            pRequest->ThrowException(
            "Unexpected error occurred in CFX tag", "" ) ;
    }
```

CCFXRequest::SetCustomData

Syntax

```
void CCFXRequest::SetCustomData(LPVOID lpvData)
```

Description

Sets custom (tag specific) data to carry with the request. Use this function to store request specific data to pass to procedures within your custom tag implementation.

Parameters

Parameter	Description
`lpvData`	Pointer to custom data

Example

The following example creates a request-specific data structure of hypothetical type MYTAGDATA and stores a pointer to the structure in the request for future use:

```
void ProcessTagRequest( CCFXRequest* pRequest )
            try
            {
                        MYTAGDATA tagData ;
                        pRequest->SetCustomData( (LPVOID)&tagData ) ;

                        ... remainder of procedure ...
    }
```

CCFXRequest::SetVariable

Syntax

`void CCFXRequest::SetVariable(LPCSTR lpszName, LPCSTR lpszValue)`

Description

Sets a variable in the calling template. If the variable name already exists in the template, its value is replaced. If it does not exist, a variable is created. The values of variables created using `SetVariable` can be accessed in the same manner as other template variables (for example, `#MessageSent#`).

Parameters

Parameter	Description
`lpszName`	Name of variable
`lpszValue`	Value of variable

Example

The following example sets the value of a variable named `'MessageSent'` based on the success of an operation performed by the custom tag:

```
BOOL bMessageSent;
    ...attempt to send the message...
    if ( bMessageSent == TRUE )
    {
                        pRequest->SetVariable( "MessageSent", "Yes" ) ;
    }
    else
    {
                        pRequest->SetVariable( "MessageSent", "No" ) ;
    }
```

CCFXRequest::ThrowException

Syntax

`void CCFXRequest::ThrowException(LPCSTR lpszError, LPCSTR lpszDiagnostics)`

Description

Throws an exception and ends processing of a request. Call this function when you encounter an error that does not allow you to continue processing the request. This function is almost always combined with the `CCFXRequest::ReThrowException` to protect against resource leaks in extension code.

Parameters

Parameter	Description
lpszError	Short identifier for error
lpszDiagnostics	Error diagnostic information

Example

The following example throws an exception indicating that an unexpected error occurred while processing a request:

```
char buffError[512] ;
    wsprintf( buffError,
            "Unexpected Windows NT error number %ld "
            "occurred while processing request.", GetLastError() ) ;

    pRequest->ThrowException( "Error occurred", buffError ) ;
```

CCFXRequest::Write

Syntax

```
void CCFXRequest::Write(LPCSTR lpszOutput)
```

Description

Writes text output back to the user.

Parameters

Parameter	Description
lpszOutput	Text to output

Example

The following example creates a buffer to hold an output string, fills the buffer with data, and writes the output back to the user:

```
CHAR buffOutput[1024] ;
    wsprintf( buffOutput, "The destination is: %s",
            pRequest->GetAttribute("DESTINATION") ) ;
    pRequest->Write( buffOutput ) ;
```

CCFXRequest::WriteDebug

Syntax

```
void CCFXRequest::WriteDebug(LPCSTR lpszOutput)
```

Description

Writes text output into the debug stream. The text is only displayed to the end-user if the tag contains the `Debug` attribute. (For more information, see `CCFXRequest::Debug`.)

Parameters

Parameter	Description
lpszOutput	Text to output

Example

The following example checks whether the `Debug` attribute is present; if so, it writes a brief debug message:

```
if ( pRequest->Debug() )
    {
                        pRequest->WriteDebug( "Top secret debug info" ) ;
    }
```

CCFXStringSet class

Abstract class that represents a set of ordered strings. Strings can be added to a set and can be retrieved by a numeric index (index values for strings are 1-based). To create a string set, use `CCFXRequest::CreateStringSet`.

Class methods

`virtual int AddString( LPCSTR lpszString )`	`CCFXStringSet::AddString` adds a string to the end of a list.
`virtual int GetCount()`	`CCFXStringSet::GetCount` gets the number of strings contained in a list.
`virtual LPCSTR GetString( int iIndex )`	`CCFXStringSet::GetString` gets the string located at the passed index.
`virtual int GetIndexForString( LPCSTR lpszString )`	`CCFXStringSet::GetIndexForString` gets the index for the passed string.

CCFXStringSet::AddString

Syntax

```
int CCFXStringSet::AddString(LPCSTR lpszString)
```

Description

Adds a string to the end of the list.

Returns

The index of the string that was added.

Parameters

Parameter	Description
`lpszString`	String to add to the list

Example

The following example demonstrates adding three strings to a string set and saving the indexes of the items that are added:

```
CCFXStringSet* pSet = pRequest->CreateStringSet() ;
    int iRed = pSet->AddString( "Red" ) ;
    int iGreen = pSet->AddString( "Green" ) ;
    int iBlue = pSet->AddString( "Blue" ) ;
```

CCFXStringSet::GetCount

Syntax

```
int CCFXStringSet::GetCount(void)
```

Description

Gets the number of strings in a string set. The value can be used with `CCFXStringSet::GetString` to iterate over the strings in the set (recall that the index values for strings in the list begin at 1).

Returns

Returns the number of strings contained in the string set.

Example

The following example demonstrates using `GetCount` with `CCFXStringSet::GetString` to iterate over a string set and write the contents of the list back to the user:

```
int nNumItems = pStringSet->GetCount() ;
    for ( int i=1; i<=nNumItems; i++ )
    {
                        pRequest->Write( pStringSet->GetString( i ) ) ;
                        pRequest->Write( "<BR>" ) ;
    }
```

CCFXStringSet::GetIndexForString

Syntax

```
int CCFXStringSet::GetIndexForString(LPCSTR lpszString)
```

Description

Searches for a passed string. The search is case-insensitive.

Returns

If the string is found, its index within the string set is returned. If it is not found, the constant `CFX_STRING_NOT_FOUND` is returned.

Parameters

Parameter	Description
`lpszString`	String to search for

Example

The following example demonstrates a search for a string and throwing an exception if it is not found:

```
CCFXStringSet* pAttribs = pRequest->GetAttributeList() ;

    int iDestination =
            pAttribs->GetIndexForString("DESTINATION") ;
    if ( iDestination == CFX_STRING_NOT_FOUND )
    {
                        pRequest->ThrowException(
                        "DESTINATION attribute not found."
                        "The DESTINATION attribute is required "
                        "by this tag." ) ;
    }
```

CCFXStringSet::GetString

Syntax

```
LPCSTR CCFXStringSet::GetString(int iIndex)
```

Description

Retrieves the string located at the passed index (index values are 1-based).

Returns

Returns the string located at the passed index.

Parameters

Parameter	Description
iIndex	Index of string to retrieve

Example

The following example demonstrates `GetString` with `CCFXStringSet::GetCount` to iterate over a string set and write the contents of a list back to the user:

```
int nNumItems = pStringSet->GetCount() ;
    for ( int i=1; i<=nNumItems; i++ )
    {
                        pRequest->Write( pStringSet->GetString( i ) ) ;
                        pRequest->Write( "<BR>" ) ;
    }
```

Chapter 10: ColdFusion Java CFX Reference

ColdFusion includes Java interfaces for building ColdFusion custom CFXs in Java.

Contents

Class libraries overview

The following Java interfaces are available for building ColdFusion custom CFXs in Java:

Interface	Methods
Custom tag interface	processRequest
Query interface	addRow getColumnIndex getColumns getData getName getRowCount setData
Request interface	attributeExists debug getAttribute getAttributeList getIntAttribute getQuery getSetting
Response interface	addQuery setVariable write writeDebug

Custom tag interface

```
public abstract interface CustomTag
```

Interface for implementing custom tags.

Classes that implement this interface can be specified in the `CLASS` attribute of the Java CFX tag. For example, in a class *MyCustomTag*, which implements this interface, the following CFML code calls the `MyCustomTag.processRequest` method:

```
<CFX_MyCustomTag>
```

Other attributes can be passed to the Java CFX tag. Their values are available using the Request object passed to the `processRequest` method.

Methods

Returns	Syntax	Description
void	`processRequest(Request` *request*`, Response` *response)*	Processes a request originating from the `CFX_mycustomtag` tag

processRequest

Description

Processes a request originating from the Java CFX tag.

Category

`Custom tag interface`

Syntax

```
public void processRequest(Request request, Response response)
```

Throws

`Exception` If an unexpected error occurs while processing the request.

Parameters

Parameter	Description
`request`	Parameters (attributes, query, and so on.) for this request
`response`	Interface for generating response to request (output, variables, queries, and so on)

Query interface

```
public abstract interface Query
```

Interface to a query used or created by a custom tag. A query contains tabular data organized by named columns and rows.

Methods

Returns	Method	Description
int	addRow()	Adds a row to the query
int	getColumnIndex(String *name*)	Gets the index of a column given its name
String[]	getColumns()	Gets a list of the column names in a query
String	getData(int iRow, int iCol)	Gets a data element from a row and column of a query
String	getName()	Gets the name of a query
int	getRowCount()	Gets the number of rows in a query
void	setData(int iRow, int iCol, String data)	Sets a data element in a row and column of a query

addRow

Description

Adds a row to a query. Call this method to append a row to a query.

Returns the index of the row that was appended to the query.

Category

Query interface

Syntax

```
public int addRow()
```

See also

setData, getData

Example

The following example demonstrates the addition of two rows to a query that has three columns, *City*, *State*, and *Zip*:

```
// Define column indexes
int iCity = 1, iState = 2, iZip = 3 ;

// First row
int iRow = query.addRow() ;
query.setData( iRow, iCity, "Minneapolis" ) ;
query.setData( iRow, iState, "MN" ) ;
query.setData( iRow, iZip, "55345" ) ;
// Second row
iRow = query.addRow() ;
query.setData( iRow, iCity, "St. Paul" ) ;
query.setData( iRow, iState, "MN" ) ;
query.setData( iRow, iZip, "55105" ) ;
```

getColumnIndex

Description

Returns the index of the column, or 0 if no such column exists.

Category

Query interface

Syntax

```
public int getColumnIndex(String name)
```

See also

getColumns, getData

Parameters

Parameter	Description
name	Name of column to get index of (lookup is case-insensitive)

Example

The following example retrieves the index of the EMAIL column and uses it to output a list of the addresses contained in the column:

```
// Get the index of the EMAIL column
int iEMail = query.getColumnIndex( "EMAIL" ) ;

// Iterate over the query and output list of addresses
int nRows = query.getRowCount() ;
for( int iRow = 1; iRow <= nRows; iRow++ )
{
    response.write( query.getData( iRow, iEMail ) + "<BR>" ) ;
}
```

getColumns

Description

Returns an array of strings containing the names of the columns in the query.

Category

Query interface

Syntax

```
public String[] getColumns()
```

Example

The following example retrieves the array of columns, then iterates over the list, writing each column name back to the user:

```
// Get the list of columns from the query
String[] columns = query.getColumns() ;
int nNumColumns = columns.length ;

// Print the list of columns to the user
response.write( "Columns in query: " ) ;
```

```
for( int i=0; i<nNumColumns; i++ )
{
    response.write( columns[i] + " " ) ;
}
```

getData

Description

Retrieves a data element from a row and column of a query. Row and column indexes begin with 1. You can find the number of rows in a query by calling `getRowCount`. You can find the number of columns in a query by calling `getColumns`.

Returns the value of the requested data element.

Category

Query interface

Syntax

```
public String getData(int iRow, int iCol)
```

Throws

`IndexOutOfBoundsException` if an invalid index is passed to the method.

See also

setData, addRow

Parameters

Parameter	Description
iRow	Row to retrieve data from (1-based)
iCol	Column to retrieve data from (1-based)

Example

The following example iterates over the rows of a query and writes the data back to the user in a simple, space-delimited format:

```
int iRow, iCol ;
int nNumCols = query.getColumns().length ;
int nNumRows = query.getRowCount() ;
for ( iRow = 1; iRow <= nNumRows; iRow++ )
{
    for ( iCol = 1; iCol <= nNumCols; iCol++ )
    {
        response.write( query.getData( iRow, iCol ) + " " ) ;
    }
    response.write( "<BR>" ) ;
}
```

getName

Description

Returns the name of a query.

Category

Query interface

Syntax

```
public String getName()
```

Example

The following example retrieves the name of a query and writes it back to the user:

```
Query query = request.getQuery() ;
    response.write( "The query name is: " + query.getName() ) ;
```

getRowCount

Description

Retrieves the number of rows in a query.

Returns the number of rows contained in a query.

Category

Query interface

Syntax

```
public int getRowCount()
```

Example

The following example retrieves the number of rows in a query and writes it back to the user:

```
Query query = request.getQuery() ;
    int rows = query.getRowCount() ;
    response.write( "The number of rows in the query is "
    + Integer.ToString(rows) ) ;
```

setData

Description

Sets a data element in a row and column of a query. Row and column indexes begin with 1. Before calling `setData` for a given row, call `addRow` and use the return value as the row index for your call to `setData`.

Category

Query interface

Syntax

```
public void setData(int iRow, int iCol, String data)
```

Throws

`IndexOutOfBoundsException` if an invalid index is passed to the method.

See also

`getData`, `addRow`

Parameters

Parameter	Description
iRow	Row of data element to set (1-based)
iCol	Column of data element to set (1-based)
data	New value for data element

Example

The following example demonstrates the addition of two rows to a query that has three columns, *City*, *State*, and *Zip*:

```
// Define column indexes
int iCity = 1, iState = 2, iZip = 3 ;

// First row
int iRow = query.addRow() ;
query.setData( iRow, iCity, "Minneapolis" ) ;
query.setData( iRow, iState, "MN" ) ;
query.setData( iRow, iZip, "55345" ) ;


// Second row
iRow = query.addRow() ;
query.setData( iRow, iCity, "St. Paul" ) ;
query.setData( iRow, iState, "MN" ) ;
query.setData( iRow, iZip, "55105" ) ;
```

Request interface

```
public abstract interface Request
```

Interface to a request made to a CustomTag. The interface includes methods for retrieving attributes passed to the tag (including queries) and reading global tag settings.

Methods

Returns	Syntax	Description
boolean	`attributeExists(String name)`	Checks whether the attribute was passed to this tag.
boolean	`debug()`	Checks whether the tag contains the `debug` attribute.
String	`getAttribute(String name)`	Retrieves the value of the passed attribute.
String[]	`getAttributeList()`	Retrieves a list of attributes passed to the tag.
int	`getIntAttribute(String name)`	Retrieves the value of the passed attribute as an integer.
int	`getIntAttribute(String name, int def)`	Retrieves the value of the passed attribute as an integer (returns default if the attribute does not exist or is not a valid number).
Query	`getQuery()`	Retrieves the query that was passed to this tag.

attributeExists

Description

Checks whether the attribute was passed to this tag.

Returns True if the attribute is available; otherwise returns False.

Category

`Request interface`

Syntax

```
public boolean attributeExists(String name)
```

See also

`getAttribute`, `getAttributeList`

Parameters

Parameter	Description
`name`	Name of the attribute to check (case-insensitive)

Example

The following example checks whether the user passed an attribute named DESTINATION to the tag; if not, it throws an exception:

```
if ( ! request.attributeExists("DESTINATION") )
{
    throw new Exception(
    "Missing DESTINATION parameter",
    "You must pass a DESTINATION parameter in "
```

```
    "order for this tag to work correctly." ) ;
} ;
```

debug

Description

Checks whether the tag contains the `debug` attribute. Use this method to determine whether to write debug information for this request. For more information, see `writeDebug`.

Returns True if the tag contains the `debug` attribute; False, otherwise.

Category

`Request interface`

Syntax

```
public boolean debug()
```

See also

`writeDebug`

Example

The following example checks whether the `debug` attribute is present, and if so, it writes a brief debug message:

```
if ( request.debug() )
{
    response.writeDebug( "debug info" ) ;
}
```

getAttribute

Description

Retrieves the value of a passed attribute. Returns an empty string if the attribute does not exist (use `attributeExists` to test whether an attribute was passed to the tag). Use `getAttribute(String,String)` to return a default value rather than an empty string.

Returns the value of the attribute passed to the tag. If no attribute of that name was passed to the tag, an empty string is returned.

Category

`Request interface`

Syntax

```
public String getAttribute(String name)
```

See also

`attributeExists`, `getAttributeList`, `getIntAttribute`, `getAttribute`

Parameters

Parameter	Description
`name`	The attribute to retrieve (case-insensitive)

Example

The following example retrieves an attribute named `DESTINATION` and writes its value back to the user:

```
String strDestination = request.getAttribute("DESTINATION") ;
response.write( "The destination is: " + strDestination ) ;
```

getAttributeList

Description

Retrieves a list of attributes passed to the tag. To retrieve the value of one attribute, use the `getAttribute` method.

Returns an array of strings containing the names of the attributes passed to the tag.

Category

Request interface

Syntax

```
public String[] getAttributeList()
```

See also

attributeExists, getAttributeList

Example

The following example retrieves the list of attributes, then iterates over the list, writing each attribute and its value back to the user:

```
String[] attribs = request.getAttributeList() ;
int nNumAttribs = attribs.length ;

for( int i = 0; i < nNumAttribs; i++ )
{
    String strName = attribs[i] ;
    String strValue = request.getAttribute( strName ) ;
    response.write( strName + "=" + strValue + "<BR>" ) ;
}
```

getIntAttribute

Description

Retrieves the value of the passed attribute as an integer. Returns -1 if the attribute does not exist. Use `attributeExists` to test whether an attribute was passed to the tag. Use `getIntAttribute(String,int)` to return a default value rather than throwing an exception or returning -1.

Returns the value of the attribute passed to the tag. If no attribute of that name was passed to the tag, -1 is returned.

Category

Request interface

Syntax

```
public int getIntAttribute(String name)
```

Throws

`NumberFormatException` if the attribute is not a valid number.

See also

attributeExists, getAttributeList, getIntAttribute

Parameters

Parameter	Description
name	The attribute to retrieve (case-insensitive)

Example

The following example retrieves an attribute named PORT and writes its value back to the user:

```
int nPort = request.getIntAttribute("PORT") ;
if ( nPort != -1 )
    response.write( "The port is: " + String.valueOf(nPort) ) ;
```

getQuery

Description

Retrieves the query that was passed to this tag.

To pass a query to a custom tag, you use the query attribute. It should be set to the name of a query (created using the cfquery tag). The query attribute is optional and should be used only by tags that process an existing dataset.

Returns the Query that was passed to the tag. If no query was passed, returns null.

Category

Request interface

Syntax

```
public Query getQuery()
```

Example

The following example retrieves a query that was passed to a tag. If no query was passed, an exception is thrown:

```
Query query = request.getQuery() ;
if ( query == null )
{
    throw new Exception(
    "Missing QUERY parameter. " +
    "You must pass a QUERY parameter in "
    "order for this tag to work correctly." ) ;
}
```

getSetting

Description

Retrieves the value of a global custom tag setting. Custom tag settings are stored in the CustomTags section of the ColdFusion Registry key.

Returns the value of the custom tag setting. If no setting of that name exists, an empty string is returned.

Category

Request interface

Syntax

```
public String getSetting(String name)
```

Parameters

Parameter	Description
name	The name of the setting to retrieve (case-insensitive)

Usage

All custom tags implemented in Java share a registry key for storing settings. To avoid name conflicts, preface the names of settings with the name of your custom tag class. For example, the code below retrieves the value of a setting named *VerifyAddress* for a custom tag class named *MyCustomTag*:

```
String strVerify = request.getSetting("MyCustomTag.VerifyAddress") ;
if ( Boolean.valueOf(strVerify) )
{
    // Do address verification...
}
```

Response interface

`public abstract interface Response`

Interface to response generated from a custom tag. This interface includes methods for writing output, generating queries, and setting variables in the calling page.

Methods

Returns	Syntax	Description
Query	`addQuery(String name, String[] columns)`	Adds a query to the calling template.
void	`setVariable(String name, String value)`	Sets a variable in the calling template.
void	`write(String output)`	Outputs text back to the user.
void	`writeDebug(String output)`	Writes text output into the debug stream.

addQuery

Description

Adds a query to the calling template. The query can be accessed by CFML tags in the template. After calling `addQuery`, the query is empty (it has 0 rows). To populate the query with data, call the Query methods `addRow` and `setData`.

Returns the Query that was added to the template.

Category

`Response interface`

Syntax

`public Query addQuery(String name, String[] columns)`

Throws

`IllegalArgumentException` If the name parameter is not a valid CFML variable name.

See also

`addRow`, `setData`

Parameters

Parameter	Description
`name`	The name of the query to add to the template
`columns`	The column names to use in the query

Example

The following example adds a query named *People* to the calling template. The query has two columns (*FirstName* and *LastName*) and two rows:

```
// Create string array with column names (also track columns indexes)
String[] columns = { "FirstName", "LastName" } ;
int iFirstName = 1, iLastName = 2 ;
```

```
// Create a query which contains these columns
Query query = response.addQuery( "People", columns ) ;

// Add data to the query
int iRow = query.addRow() ;
query.setData( iRow, iFirstName, "John" ) ;
query.setData( iRow, iLastName, "Smith" ) ;
iRow = query.addRow() ;
query.setData( iRow, iFirstName, "Jane" ) ;
query.setData( iRow, iLastName, "Doe" ) ;
```

setVariable

Description

Sets a variable in the calling template. If the variable name specified exists in the template, its value is replaced. If it does not exist, a new variable is created.

Category

Response interface

Syntax

`public void setVariable(String name, String value)`

Throws

`IllegalArgumentException` If the `name` parameter is not a valid CFML variable name.

Parameters

Parameter	Description
`name`	The name of the variable to set
`value`	The value to set the variable to

Example

For example, this code sets the value of a variable named *MessageSent* based on the success of an operation performed by the custom tag:

```
boolean bMessageSent ;

    ...attempt to send the message...

    if ( bMessageSent == true )
    {
        response.setVariable( "MessageSent", "Yes" ) ;
    }
    else
    {
        response.setVariable( "MessageSent", "No" ) ;
    }
```

write

Description

Outputs text back to the user.

Category

Response interface

Syntax

```
public void write(String output)
```

Parameters

Parameter	Description
output	Text to output

Example

The following example outputs the value of the DESTINATION attribute:

```
response.write( "DESTINATION = " +
    request.getAttribute("DESTINATION") ) ;
```

writeDebug

Description

Writes text output into the debug stream. This text is displayed to the end-user only if the tag contains the debug attribute (check for this attribute using the Request.debug method).

Category

Response interface

Syntax

```
public void writeDebug(String output)
```

See also

debug

Parameters

Parameter	Description
output	The text to output

Example

The following example checks whether the debug attribute is present; if so, it writes a brief debug message:

```
if ( request.debug() )
{
    response.writeDebug( "debug info" ) ;
}
```

Debugging classes reference

The constructors and methods supported by the `DebugRequest`, `DebugResponse`, and `DebugQuery` classes are as follows. These classes also support the other methods of the `Request`, `Response`, and `Query` interfaces, respectively.

DebugRequest

```
// initialize a debug request with attributes
public DebugRequest( Hashtable attributes ) ;

// initialize a debug request with attributes and a query
public DebugRequest( Hashtable attributes, Query query ) ;

// initialize a debug request with attributes, a query, and settings
public DebugRequest( Hashtable attributes, Query query, Hashtable settings ) ;
```

DebugResponse

```
// initialize a debug response
public DebugResponse() ;

// print the results of processing
public void printResults() ;
```

DebugQuery

```
// initialize a query with name and columns
public DebugQuery( String name, String[] columns )
 throws IllegalArgumentException ;

// initialize a query with name, columns, and data
public DebugQuery( String name, String[] columns, String[][] data )
    throws IllegalArgumentException ;
```

Chapter 11: WDDX JavaScript Objects

You use JavaScript objects and functions to use with WDDX in a ColdFusion application.

Contents

JavaScript object overview

These are the JavaScript objects and functions:

Class	Functions
WddxSerializer object	serialize serializeVariable serializeValue write
WddxRecordset object	addColumn addRows getField getRowCount setField wddxSerialize

WDDX JavaScript objects are defined in the wddx.js file; this file is installed in the CFIDE/scripts directory.

To use these objects, you must put a JavaScript tag before the code that refers to the objects; for example:

```
<script type="text/javascript" src="/CFIDE/scripts/wddx.js"></script>
```

WddxSerializer object

The WddxSerializer object includes functions that serialize any JavaScript data structure. For more information on using this object, see "Using WDDX" on page 896 in the *ColdFusion Developer's Guide*.

Functions

The only function that developers typically call is `serialize`.

Function syntax	Description
object.`serialize`(*rootobj*)	Creates a WDDX packet for a passed WddxRecordset instance.
object.`serializeVariable`(*name*, *obj*)	Serializes a property of a structure. If an object is not a string, number, array, Boolean, or a date, WddxSerializer treats it as a structure.
object.`serializeValue`(*obj*)	Recursively serializes eligible data in a passed instance.
object.`write`(*str*)	Appends data to the serialized data stream.

serialize

Description

Creates a WDDX packet for a passed WddxRecordset instance.

Syntax

```
object.serialize( rootobj )
```

Parameters

Parameter	Description
`object`	Instance name of the WddxSerializer object
`rootobj`	JavaScript data structure to serialize

Return value

Returns a serialized WDDX packet as a string if the function succeeds, or a null value if an error occurs.

Usage

Call this function to serialize the data in a WddxRecordset instance.

Example

This example shows a JavaScript function that you can call to serialize a WddxRecordset instance. It copies serialized data to a form field for display:

```
function serializeData(data, formField)
{
    wddxSerializer = new WddxSerializer();
    wddxPacket = wddxSerializer.serialize(data);
    if (wddxPacket != null)
    {
        formField.value = wddxPacket;
    }
    else
    {
```

```
        alert("Couldn't serialize data");
    }
}
```

serializeVariable

Description

Serializes a property of a structure. If an object is not a string, number, array, Boolean, or date, WddxSerializer treats it as a structure.

Syntax

```
object.serializeVariable( name, obj )
```

Parameters

Parameter	Description
`object`	Instance name of a WddxSerializer object
`name`	Property to serialize
`obj`	Instance name of the value to serialize

Return value

Returns a Boolean True if serialization was successful; False, otherwise.

This is an internal function; you do not typically call it.

Example

This example is from the WddxSerializer `serializeValue` function:

```
...
// Some generic object; treat it as a structure
    this.write("<struct>");
    for (prop in obj)
    {
        bSuccess = this.serializeVariable(prop, obj[prop]);
        if (! bSuccess)
        {
            break;
        }
    }
    this.write("</struct>");
...
```

serializeValue

Description

Recursively serializes eligible data in a passed instance. Eligible data includes:

- String
- Number
- Boolean
- Date
- Array

- Recordset
- Any JavaScript object

This function serializes null values as empty strings.

Syntax

```
object.serializeValue( obj )
```

Parameters

Parameter	Description
object	Instance name of the WddxSerializer object
obj	Instance name of the WddxRecordset object to serialize

Return value

Returns a Boolean True if *obj* was serialized successfully; False, otherwise.

Usage

This is an internal function; you do not typically call it.

Example

This example is from the WddxSerializer `serialize` function:

```
...
this.wddxPacket = "";
this.write("<wddxPacket version='1.0'><header/><data>");
bSuccess = this.serializeValue(rootObj);
this.write("</data></wddxPacket>");
if (bSuccess)
{
    return this.wddxPacket;
}
else
{
    return null;
}
...
```

write

Description

Appends data to a serialized data stream.

Syntax

```
object.write( str )
```

Parameters

Parameter	Description
object	Instance name of the WddxSerializer object
str	String to be copied to the serialized data stream

Return value

Returns an updated serialized data stream as a String.

Usage

This is an internal function; you do not typically call it.

Example

This example is from the WddxSerializer `serializeValue` function:

```
...
else if (typeof(obj) == "number")
{
    // Number value
    this.write("<number>" + obj + "</number>");
}
else if (typeof(obj) == "boolean")
{
    // Boolean value
    this.write("<boolean value='" + obj + "'/>");
}
...
```

WddxRecordset object

Includes functions that you call as needed when constructing a WDDX record set. For more information on using this object, see "Using WDDX" on page 896 in the *ColdFusion Developer's Guide*.

Functions

Function syntax	Description
object.addColumn(*name*)	Adds a column to all rows in a WddxRecordset instance.
object.addRows(*n*)	Adds rows to all columns in a WddxRecordset instance.
object.dump(*escapeStrings*)	Displays WddxRecordset object data.
object.getField(*row*, *col*)	Returns the element in a row/column position.
object.getRowCount()	Indicates the number of rows in a WddxRecordset instance.
object.setField(*row*, *col*, *value*)	Sets the element in a row/column position.
object.wddxSerialize(*serializer*)	Serializes a record set.

Returns

HTML table of the WddxRecordset object data.

Usage

Convenient for debugging and testing record sets. The boolean parameter `escapeStrings` determines whether `<>&` characters in string values are escaped as `<>&` in HTML.

Example

```
<!--- Create a simple query --->
<cfquery name = "q" datasource ="cfdocexamples">
    SELECT Message_Id, Thread_id, Username, Posted
    FROM messages
</cfquery>
<!--- Load the wddx.js file, which includes the dump function --->
<script type="text/javascript" src="/CFIDE/scripts/wddx.js"></script>
<script>
// Use WDDX to move from CFML data to JS
<cfwddx action="cfml2js" input="#q#" topLevelVariable="qj">
// Dump the record set
document.write(qj.dump(true));
</script>
```

addColumn

Description

Adds a column to all rows in a WddxRecordset instance.

Syntax

```
object.addColumn( name )
```

Parameters

Parameter	Description
object	Instance name of the WddxRecordset object
name	Name of the column to add

Return value

None.

Usage

Adds a column to every row of the WDDX record set. Initially the new column's values are set to NULL.

Example

This example calls the addColumn function:

```
// Create a new record set
rs = new WddxRecordset();

// Add a new column
rs.addColumn("NewColumn");

// Extend the record set by 3 rows
rs.addRows(3);

// Set an element in the first row
// newValue is a previously defined variable
rs.setField(0, "NewColumn", newValue);
```

addRows

Description

Adds rows to all columns in a WddxRecordset instance.

Syntax

```
object.addRows( n )
```

Parameters

Parameter	Description
object	Instance name of the WddxRecordset object
n	Integer; number of rows to add

Return value

None.

Usage

This function adds the specified number of rows to every column of a WDDX record set. Initially, the row/column values are set to NULL.

Example

This example calls the addRows function:

```
// Create a new record set
```

```
rs = new WddxRecordset();

// Add a new column
rs.addColumn("NewColumn");

// Extend the record set by 3 rows
rs.addRows(3);

// Set an element in the first row
// newValue is a previously defined variable
rs.setField(0, "NewColumn", newValue);
```

getField

Description

Returns the element in the specified row/column position.

Syntax

```
object.getField( row, col )
```

Parameters

Parameter	Description
object	Instance name of the WddxRecordset object
row	Integer; zero-based row number of the value to return
col	Integer or string; column of the value to be returned.

Return value

Returns the value in the specified row/column position.

Usage

Call this function to access a value in a WDDX record set.

Example

This example calls the `getField` function (the variable *r* is a reference to a WddxRecordset instance):

```
for (row = 0; row < nRows; ++row)
{
    o += "<tr>";
    for (i = 0; i < colNames.length; ++i)
    {
        o += "<td>" + r.getField(row, colNames[i]) + "</td>";
    }
    o += "</tr>";
}
```

getRowCount

Description

Indicates the number of rows in a WddxRecordset instance.

Syntax

```
object.getRowCount( )
```

Parameters

Parameter	Description
object	Instance name of a WddxRecordset object

Return value

Integer. Returns the number of rows in the WddxRecordset instance.

Usage

Call this function before a looping construct to determine the number of rows in a record set.

Example

This example calls the `getRowCount` function:

```
function dumpWddxRecordset(r)
{
// Get row count
    nRows = r.getRowCount();
...
    for (row = 0; row < nRows; ++row)
...
```

setField

Description

Sets the element in the specified row/column position.

Syntax

```
object.setField( row, col, value )
```

Parameters

Parameter	Description
object	Instance name of a WddxRecordset object
row	Integer; row that contains the element to set
col	Integer or string; the column containing the element to set
value	Value to set

Return value

None.

Usage

Call this function to set a value in a WddxRecordset instance.

Example

This example calls the `setField` function:

```
// Create a new recordset
rs = new WddxRecordset();

// Add a new column
rs.addColumn("NewColumn");
```

```
// Extend the record set by 3 rows
rs.addRows(3);

// Set an element in the first row
// newValue is a previously defined variable
rs.setField(0, "NewColumn", newValue);
```

wddxSerialize

Description

Serializes a record set.

Syntax

`object.wddxSerialize( serializer )`

Parameters

Parameter	Description
`object`	Instance name of the WddxRecordset object
`serializer`	WddxSerializer instance

Return value

Returns a Boolean True if serialization was successful; False, otherwise.

Usage

This is an internal function; you do not typically call it.

Example

This example is from the WddxSerializer `serializeValue` function:

```
...
else if (typeof(obj) == "object")
{
if (obj == null)
{
// Null values become empty strings
this.write("<string></string>");
}
else if (typeof(obj.wddxSerialize) == "function")
{
// Object knows how to serialize itself
bSuccess = obj.wddxSerialize(this);
}
...
```

Chapter 12: ColdFusion ActionScript Functions

ColdFusion includes two server-side ActionScript functions, `CF.query` and `CF.http`., including specific syntax and methods.

Contents

CF.query

Description

Performs queries against ColdFusion data sources.

Return value

Returns a RecordSet object.

Syntax

```
CF.query
        ({
            datasource:"data source name",
            sql:"SQL stmts",
            username:"username",
            password:"password",
            maxrows:number,
            timeout:milliseconds
        })
```

Arguments

Arguments	Req/Opt	Description
datasource	Required	Name of the data source from which the query retrieves data.
sql	Required	SQL statement.
username	Optional	Username. Overrides the username specified in the data source setup.
password	Optional	Password. Overrides the password specified in the data source setup.
maxrows	Optional	Maximum number of rows to return in the record set.
timeout	Optional	Maximum number of seconds for the query to execute before returning an error indicating that the query has timed out. Can only be used in named arguments.

Usage

You can code the `CF.query` function using named or positional arguments. You can invoke all supported arguments using the named argument style, as follows:

```
CF.query({datasource:"datasource", sql:"sql stmt",
    username:"username", password:"password", maxrows:"maxrows",
    timeout:"timeout"});
```

Note: *The named argument style uses curly braces {} to surround the function arguments.*

Positional argument style, which is a shorthand coding style, does not support all arguments. Use the following syntax to code the `CF.query` function using positional arguments:

```
CF.query(datasource, sql);
CF.query(datasource, sql, maxrows);
CF.query(datasource, sql, username, password);
CF.query(datasource, sql, username, password, maxrows);
```

Note: *Do not use curly braces {} with positional arguments.*

You can manipulate the record set returned by the `CF.query` function using methods in the RecordSet ActionScript class. The following are some of the methods available in the RecordSet class:

- `RecordSet.getColumnnames`
- `RecordSet.getLength`

- RecordSet.getItemAt
- RecordSet.getItemID
- RecordSet.sortItemsBy
- RecordSet.getNumberAvailable
- RecordSet.filter
- RecordSet.sort

For more information on using server-side ActionScript, see "Using Server-Side ActionScript" on page 708 in the *ColdFusion Developer's Guide*. For more detailed information about the RecordSet ActionScript class, see *Using Flash Remoting*.

Example

```
// Define a function to do a basic query
// Note use of positional arguments
function basicQuery()
{
    result = CF.query("myquery", "cust_data", "SELECT * from tblParks");
    return result;
}

// Example function declaration using named arguments
function basicQuery()
{
    result = CF.query({datasource:"cust_data", sql:"SELECT * from tblParks"});
    return result;
}

// Example of the CF.query function using maxrows argument
function basicQueryWithMaxRows()
{
    result = CF.query("cust_data", "SELECT * from tblParks", 25);
    return result;
}

// Example of the CF.query function with username and password
function basicQueryWithUser()
{
    result = CF.query("cust_data", "SELECT * from tblParks",
        "wsburroughs", "migraine1");
    return result;
}
```

CF.http

Description

Executes HTTP POST and GET operations on files. (POST operations upload MIME file types to a server, or post cookie, formfield, URL, file, or CGI variables directly to a server.)

Return value

Returns an object containing properties that you reference to access data.

Syntax

```
CF.http
    ({
            method:"get or post",
        url:"URL",
        username:"username",
        password:"password",
        resolveurl:"yes or no",
        params:arrayvar,
        path:"path",
        file:"filename"
    })
```

Arguments

Arguments	Req/Opt	Description
method	Required	One of two arguments: • get: downloads a text or binary file or creates a query from the contents of a text file. • post: sends information to the server page or CGI program for processing. Requires the `params` argument.
url	Required	The absolute URL of the host name or IP address of the server on which the file resides. The URL must include the protocol (http or https) and host name.
username	Optional	When required by a server, a username.
password	Optional	When required by a server, a password.

Arguments	Req/Opt	Description

resolveurl	Optional	For `Get` and `Post` methods.
		• Yes or No. Default is No.
		For GET and POST operations, if Yes, the page reference that is returned into the Filecontent property has its internal URLs fully resolved, including port number, so that links remain intact. The following HTML tags, which can contain links, are resolved:
		`-img src`
		`-a href`
		`-form action`
		`-applet code`
		`-script src`
		`-embed src`
		`-embed pluginspace`
		`-body background`
		`-frame src`
		`-bgsound src`
		`-object data`
		`-object classid`
		`-object codebase`
		`-object usemap`

params	Optional	HTTP parameters passed as an array of objects. Supports the following parameter types: • name • type • value `CF.http` params are passed as an array of objects. The `params` argument is required for POST operations.
path	Optional	The path to the directory in which to store files. When using the `path` argument, the `file` argument is required.
file	Optional	Name of the file that is accessed. For GET operations, defaults to the name specified in the `url` argument. Enter path information in the `path` argument. This argument is required if you are using the `path` argument.

Usage

You can write the `CF.http` function using named arguments or positional arguments. You can invoke all supported arguments using the named argument style, as follows:

```
CF.http({method:"method", url:"URL", username:"username", password:"password",
    resolveurl:"yes or no", params:arrayvar,
    path:"path", file:"filename"});
```

Note: *The named argument style uses curly braces {} to surround the function arguments.*

Positional arguments let you use a shorthand coding style. However, not all arguments are supported for the positional argument style. Use the following syntax to code the `CF.http` function using positional arguments:

```
CF.http(url);
CF.http(method, url);
CF.http(method, url, username, password);
```

```
CF.http(method, url, params, username, password);
```

Note: *Do not use curly braces {} with positional arguments.*

The following parameters can only be passed as an array of objects in the `params` argument in the `CF.http` function:

Parameter	**Description**
name	The variable name for data that is passed
type	The transaction type: • URL • FormField • Cookie • CGI • File
value	Value of URL, FormField, Cookie, File, or CGI variables that are passed

The CF.http function returns data as a set of object properties, as described in the following table:

Property	**Description**
Text	A Boolean value that indicates whether the specified URL location contains text data.
Charset	The charset used by the document specified in the URL. HTTP servers normally provide this information, or the charset is specified in the charset parameter of the Content-Type header field of the HTTP protocol. For example, the following HTTP header announces that the character encoding is EUC-JP: Content-Type: text/html; charset=EUC-JP
Header	Raw response header. For example: HTTP/1.1 200 OK Date: Mon, 04 Mar 2002 17:27:44 GMT Server: Apache/1.3.22 (Unix) mod_perl/1.26 Set-Cookie: MM_cookie=207.22.48.162.4731015262864476; path=/; expires=Wed, 03-Mar-04 17:27:44 GMT; domain=..com Connection: close Content-Type: text/html
Filecontent	File contents, for text and MIME files.

Property	Description
Mimetype	MIME type. Examples of MIME types include text/html, image/png, image/gif,video/mpeg, text/css, and audio/basic.
responseHeader	Response header. If there is only one header key, its value can be accessed as simple type. If there are multiple header keys, the values are put in an array in a responseHeader structure.
Statuscode	HTTP error code and associated error string. Common HTTP status codes returned in the response header include: 400: Bad Request 401: Unauthorized 403: Forbidden 404: Not Found 405: Method Not Allowed

You access these attributes using the `get` function:

```
function basicGet()
{
url = "http://localhost:8100/";

// Invoke with just the url. This is an HTTP GET.
result = CF.http(url);
return result.get("Filecontent");
}
```

Note: *For more information on using server-side ActionScript, see "Using Server-Side ActionScript" on page 708 in the ColdFusion Developer's Guide.*

Example

The following examples show a number of the ways to use the `CF.http` function:

```
function postWithNamedArgs()
{
    // Set up the array of Post parameters.
    params = new Array();
    params[1] = {name:"arg1", type:"FormField", value:"value1"};
    params[2] = {name:"arg2", type:"URL", value:"value2"};
    params[3] = {name:"arg3", type:"CGI", value:"value3"};

    url = "http://localhost:8100/";

    path = application.getContext("/").getRealPath("/");
    file = "foo.txt";

    result = CF.http({method:"post", url:url, username:"karl", password:"salsa",
    resolveurl:true, params:params, path:path, file:file});

    if (result)
        return result.get("Statuscode");
    return null;
}

// Example of a basic HTTP GET operation
// Shows that HTTP GET is the default
function basicGet()
{
    url = "http://localhost:8100/";
```

```
    // Invoke with just the url. This is an HTTP GET.
    result = CF.http(url);
    return result.get("Filecontent");
}

// Example showing simple array created to pass params arguments
function postWithParams()
{
    // Set up the array of Post parameters. These are just like cfhttpparam tags.
    params = new Array();
    params[1] = {name:"arg2", type:"URL", value:"value2"};

    url = "http://localhost:8100/";

    // Invoke with the method, url, and params
    result = CF.http("post", url, params);
    return result.get("Filecontent");
}

// Example with username and params arguments
function postWithParamsAndUser()
{
    // Set up the array of Post parameters. These are just like cfhttpparam tags.
    params = new Array();
    params[1] = {name:"arg2", type:"URL", value:"value2"};

    url = "http://localhost:8100/";

    // Invoke with the method, url, params, username, and password
    result = CF.http("post", url, params, "karl", "salsa");
    return result.get("Filecontent");
}
```