CFML Reference
ADOBE® COLDFUSION (2016 release)

Legal notices

For legal notices, see http://help.adobe.com/en_US/legalnotices/index.html.

Contents

Chapter 9: ColdFusion Event Gateway Reference

Chapter 1: What's New

New and changed functions/tags in Adobe ColdFusion (2016 release)

This document lists new and changed functions/tags in Adobe ColdFusion (2016 release).

New functions

The following is a list of new functions in Adobe ColdFusion (2016 release):

ArrayContainsNoCase

ArrayDeleteNoCase

BooleanFormat

Floor

IsPDFArchive

QuerySort

QueryEach

QueryFilter

QueryKeyExists

QueryMap

QueryReduce

ReplaceListNoCase

SpreadsheetGetColumnCount

ValueArray

SpreadsheetAddRows

There is a new parameter, includeColumnNames, in this function. For more information, see SpreadsheetAddRows function description and examples.

ReplaceList

There is a new parameter, includeEmptyFields, in this function. For more information, see ReplaceList function description and examples.

cfapplication

There are two new attributes in the <cfapplication> tag:

1. passArrayByReference
2. searchImplicitScopes

For more information, see cfapplication .

cfsearch

The verity types, simple, explicit, internet, internet_basic, and natural no longer exist. There are two new verity types, **Standard** and **DisMax**, in the tag. For more information, see cfsearch .

cfloop

There is a new attribute, item, in <cfloop>. The attribute, index, is now optional.

For more information, see cfloop .

cfmailparam

There is a new, optional attribute, filename, in cfmailparam.

For more information, see cfmailparam .

cfoutput

There is a new, optional attribute, encodefor in this tag.

For more information, see cfoutput .

cfpdf

In Adobe ColdFusion (2016 release), you can use two additional 256-bit encryption algorithms when protecting a PDF document. The algorithms are:

1. AES_256R5
2. AES_256R6

For more information, see cfpdf .

CacheRemove

The CacheRemove function has an updated syntax:

CacheRemove(Object id, boolean throwOnError, String key, boolean exact)

For more information, see CacheRemove .

TimeFormat

The TimeFormat function has new masks to represent time zones in different formats. For more information, see TimeFormat function description and examples.

DateFormat

The DateFormat function has new masks to represent time zones in different formats. For more information, see DateFormat function description and examples.

Replace

The Replace function has an updated description and syntax. This function can take either string or callback function as an argument instead of the argument to replace the string. For more information, see Replace function description.

StructNew

The function has a new parameter, structType, that represents the type of struct to be created.

For more information, see StructNew .

WriteOutput

The function, WriteOutput, has a new parameter, encodefor. encodefor applies encoding on the input string.

For more information, see WriteOutput .

Elements of CFML

The basic elements of CFML, including tags, functions, constants, variables, expressions, and CFScript, make it a powerful tool for developing interactive web applications.

CFML Basics

Comments

Tags

Functions

ColdFusion components

Constants

Variables

Expressions

Data types

Flow control

Character case

Special characters

Reserved words in ColdFusion

cfscript tag

Elvis operator

Using Arrays and Structures

Adobe ColdFusion supports dynamic multidimensional arrays. Using arrays can enhance your ColdFusion application code. Adobe ColdFusion also supports structures for managing lists of key-value pairs. Because structures can contain other structures or complex data types as it values, they provide a flexible and powerful tool for managing complex data.

About arrays

Basic array techniques

Populating arrays with data

Array functions-Developing guide

About structures

Creating and using structures

Structure examples

Structure functions - Developing guide

Chapter 2: Reserved Words and Variables

Reserved Words and Variables

Adobe ColdFusion language includes reserved words and scope variables.

Reserved words

Scope-specific built-in variables

Custom tag variables

ColdFusion tag-specific variables

CGI environment (CGI Scope) variables

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 in the Developing ColdFusion Applications 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 , and Using Persistent Data and Locking in the Developing ColdFusion Applications.

See also

- Variable scope
- Caller scope
- CGI variables
- Client variables
- Server variables
- Application and session variables

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>
```

See also

- Request variable
- Form variable

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
```

See also

- ColdFusion query variables
- CFCATCH variables
- CFDIRECTORY variables
- CFERROR variables
- CFFILE ACTION=Upload variables
- CFFTP error variables
- CFFTP ReturnValue variable

- CFFTP query object columns
- CFHTTP variables
- CFLDAP variables
- CFPOP variables
- CFQUERY and CFSTOREDPROC variables
- CFREGISTRY variables
- CFSEARCH variables

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. They 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

CGI server variables

CGI client variables

CGI client certificate variables

Chapter 3: ColdFusion Tags

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.

Tags in ColdFusion 10

The following table briefly describes CFML tags added in ColdFusion 10:

CFML tag	Category	Description
cfexchangeconversation	Communications tags	Helps users organize and manage conversations from a Microsoft Exchange account.
cfexchangefolder	Communications tags	Allows you to perform various actions on the mail folder, such as get folder information, find folders, or create, copy, modify, move, delete, and empty the contents of a folder.
cfwebsocket	Web Socket tags	Lets you create the WebSocket object in your CFM template. The tag creates a reference to the WebSocket JavaScript object at the client-side.

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
cfclient		The <cfClient> tag is a marker tag that instructs ColdFusion to generate client-side code (JavaScript) for ColdFusion code.
cfclientsettings		This tag is similar to cfprocessingdirective and acts as a compiler directive to include plugins for various features (device detection and device API).
cfcol	Data output tags	Defines table column header, properties
cfcollection	Extensibility tags	Administers Solr collections
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
cfcontinue	Flow-control tags	Returns processing to the top of a loop; used within a cfloop tag.
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
cfexchangeconversation	Communications tags	Helps users organize and manage conversations from a Microsoft Exchange account.
cfexchangefolder	Communications tags	Allows you to perform various actions on the mail folder, such as get folder information, find folders, or create, copy, modify, move, delete, and empty the contents of a folder.
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
cffileupload	File management tags Forms tags	Displays a dialog for uploading multiple files from the user's system.
cffinally	Exception handling tags	Used inside a cftry tag
cfflush	Data output tags , Page processing tags	Flushes currently available data to client
cfform	Forms tags	Builds input form; performs client-side input validation
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
cfhtmltopdf cfhtmltopdf	Data output tags	<cfhtmltopdf> creates high quality PDF output from a text block containing CFML and HTML using the PDF Service Manager.
cfhtmltopdfitem	Data output tags	The <cfhtmltopdfitem> specifies the action items for a PDF document created by the <cfhtmltopdf>tag.
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.
cfimap	Communications tags , Internet protocol tags	Retrieves and manages e-mails and folders in IMAP servers
cfimapfilter	Communication tags	Specifies filter parameters that control the actions of cfimap, get operations.
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 Solr 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

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
cfmap	Other tags	Embeds a Google map within a ColdFusion web page
cfmapitem	Other tags	Creates markers on the map; a child tag of the cfmap tag
cfmediaplayer	Other tags	Creates an in-built media player that can play FLV files
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.
cfmessagebox	Application framework tags	Defines a control for displaying pop-up messages
cfNTauthenticate	Security tags	Authenticates user information against an NT domain
cfoauth	Communications tags , Internet protocol tags	The <oauth> tag allows you to easily integrate third-party Oauth 2 authentication provider
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.
cfpdfform	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
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
cfprogressbar	Other tags	Defines a progress bar to indicate the progress of an activity
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 Solr 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
cfsharepoint	Extensibility tags	Invokes a SharePoint action from ColdFusion
cfsilent	Data output tags , Page processing tags	Suppresses CFML output within tag scope
cfslider	Forms tags	Creates slider control; used in cfform

cfspreadsheet	Extensibility tags	Manages Excel spreadsheet files
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.
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
cfwebsocket	Web Socket tags	Lets you create the WebSocket object in your CFM template. The tag creates a reference to the WebSocket JavaScript object at the client-side.
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

This section lists tags by their function or purpose.

Application framework tags

Communications tags

Database manipulation tags

Data output tags

Tag changes since ColdFusion 5

The following section lists 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

Deprecated tags, attributes, and values

Obsolete tags, attributes, and values

Tags a-b

This section lists tags starting with alphabets A-B with their description.

cfabort

cfajaximport

cfajaxproxy

cfapplet

cfapplication

cfargument

cfassociate

cfauthenticate

cfbreak

Tags r-s

This section lists tags starting with alphabets R-S with their description.

cfregistry

cfreport

cfreportparam

cfrethrow

cfreturn

cfsavecontent

cfschedule

cfscript

cfsearch

cfselect

cfservlet

cfservletparam

cfset

cfsetting

cfsharepoint

cfsilent

cfslider

cfspreadsheet

cfsprydataset

cfstoredproc

cfswitch

Tags t

This section lists tags starting with alphabet T with their description.

cftable

cftextarea

cftextinput

cfthread

cfthrow

cftimer

cftooltip

cftrace

cftransaction

cftree

cftreeitem

cftry

Tags u-z

This section lists tags starting with alphabets U-Z with their description.

cfupdate

cfwddx

cfwebsocket

cfwindow

cfxml

cfzip

cfzipparam

Tags m-o

This section lists tags starting with alphabets M-O with their description.

cfmail

cfmailparam

cfmailpart

cfmap

cfmapitem

cfmediaplayer

cfmenu

Tags g-h

This section lists tags starting with alphabets G-H with their description.

cfgraph

cfgraphdata

cfgrid

cfgridcolumn

cfgridrow

cfgridupdate

cfheader

cfhtmlhead

cfhtmltopdf

cfhtmltopdfitem

cfhttp

cfhttpparam

Tags f

This section lists tags starting with alphabet F with their description.

cffeed

cffile

cffile action = "append"

cffile action = "copy"

cffile action = "delete"

cffile action = "move"

cffile action = "read"

cffile action = "readBinary"

cffile action = "rename"

cffile action = "upload"

cffile action = "uploadAll"

cffile action = "write"

cffileupload

cffinally

cfflush

cfformgroup

cfformitem

cfftp

cfftp: Opening and closing FTP server connections

cfftp: Opening and closing secure FTP server connections

cfftp: Connection: file and directory operations

cfftp action = "listDir"

cfform

cffunction

Tags d-e

This section lists tags starting with alphabets D-E with their description.

cfdbinfo

cfdefaultcase

cfdirectory

cfdiv

cfdocument

cfdocumentitem

cfdocumentsection

cfdump

cfelse

cfelseif

cferror

cfexchangecalendar

cfexchangeconnection

cfexchangecontact

cfexchangeconversation

cfexchangefilter

cfexchangefolder

cfexchangemail

cfexchangetask

cfexecute

cfexit

Tags p-q

This section lists tags starting with alphabets P-Q with their description.

cfparam

cfpdf

cfpdfform

cfpdfformparam

cfpdfparam

Tags j-l

This section lists tags starting with alphabets J-L with their description.

Tags i

This section lists tags starting with alphabet I with their description.

Tags c

This section lists tags starting with alphabet C with their description.

Chapter 4: ColdFusion Functions

ColdFusion Functions

The following list shows the ColdFusion Markup Language (CFML) functions. New Functions in ColdFusion 10

Functions by category

Function changes since ColdFusion 5

Functions a-b

Functions c-d

Functions e-g

Functions h-im

Functions in-k

Functions l

Functions m-r

Functions s

Functions t-z

New Functions in ColdFusion 10

ArraySlice ArrayEach ArrayFilter ArrayFindAll ArrayFindAllNoCase	ImageMakeTranslucent Invoke IsClosure ListFilter LSDateTimeFormat
CacheIdExists CacheRegionNew CacheRegionRemove CacheRemoveAll Canonicalize CacheRegionExists	ListRemoveDuplicates OnWSAuthenticate ORMIndex ORMIndexPurge ORMSearch ORMSearchOffline
CallStackDump CallStackGet CSRFGenerateToken CSRFVerifyToken DateTimeFormat DecodeForHTML	ReEscape RestInitApplication RemoveCachedQuery RestDeleteApplication RestSetResponse SessionRotate
DecodeFromURL DirectoryCopy EncodeForHTML EncodeForCSS EncodeForHTMLAttribute EncodeForJavaScript	SessionGetMetaData SessionInvalidate StructEach StructFilter WSGetAllChannels WSGetSubscribers
EncodeForURL EncodeForXML FileGetMimeType GetApplicationMetadata GetCPUUsage	WSPublish WSSendMessage
GetTotalSpaceGetFreeSpace GetSystemFreeMemoryGetTotalSpace GetSystemFreeMemory GetSystemTotalMemory HMac ImageCreateCaptcha ImageMakeColorTransparent	

Functions by category

The following section list functions by their category or purpose.

Array functions

Cache functions

Conversion functions

Date and time functions

Data output functions

Debugging functions

Decision functions

Display and formatting functions

Dynamic evaluation functions

Exception handling functions

Extensibility functions

Flow control functions

Full-text search functions

Image functions

International functions

List functions

Mathematical functions

Microsoft Office integration functions

ORM functions

Function changes since ColdFusion 5

The tables in the sub-section lists 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

Deprecated functions, parameters, and values

Obsolete functions, parameters, and values

Functions a-b

Abs

ACos

AddSOAPRequestHeader

AddSOAPResponseHeader

AjaxLink

AjaxOnLoad

ApplicationStop

ArrayAppend

ArrayAvg

ArrayClear

ArrayContains

ArrayDelete

ArrayDeleteAt

ArrayDeleteNoCase

ArrayEach

Functions c-d

DotNetToCFType

Duplicate

Deserialize

DeserializeXML

Functions h-im

Functions m-r

Functions in-k

IsXmlNode

IsXmlRoot

JavaCast

JSStringFormat

Functions l

LCase

Left

Functions t-z

Tan

ThreadJoin

ThreadTerminate

Throw

TimeFormat

ToBase64

ToBinary

YesNoFormat

Functions s

SpreadsheetAddSplitPane

SpreadsheetCreateSheet

SpreadsheetDeleteColumn

SpreadsheetDeleteColumns

SpreadsheetDeleteRow

SpreadsheetDeleteRows

SpreadsheetFormatCell

SpreadsheetFormatColumn

SpreadsheetFormatCellRange

SpreadsheetFormatColumns

SpreadsheetFormatRow

SpreadsheetFormatRows

SpreadsheetGetCellComment

SpreadsheetGetCellFormula

SpreadsheetGetCellValue

SpreadsheetInfo

SpreadsheetMergeCells

SpreadsheetNew

SpreadsheetRead

SpreadsheetReadBinary

SpreadsheetRemoveSheet

SpreadsheetSetActiveSheet

SpreadsheetSetActiveSheetNumber

SpreadsheetSetCellComment

SpreadsheetSetCellFormula

SpreadsheetSetCellValue

SpreadsheetSetColumnWidth

SpreadsheetSetFooter

SpreadsheetSetHeader

SpreadsheetSetRowHeight

SpreadsheetShiftColumns

SpreadsheetShiftRows

SpreadsheetWrite

Sqr

StripCR

StructAppend

StructClear

StructCopy

StructCount

StructDelete

StructEach

StructFilter

StructFind

StructFindKey

StructFindValue

StructGet

StructInsert

StructIsEmpty

StructKeyArray

StructKeyExists

StructKeyList

StructNew

StructSort

StructUpdate

SpreadSheetAddPagebreaks

SpreadSheetAddAutofilter

StructReduce

StructMap

Serialize

SerializeXML

Functions e-g

EncodeForCSS

EncodeForHTML

EncodeForHTMLAttribute

EncodeForXMLAttribute

EncodeForJavaScript

EncodeForURL

EncodeForXML

EncodeForXpath

Encrypt

GetTempDirectory

GetApplicationMetadata

EncryptBinary

EntityDelete

EntityLoad

EntityLoadByExample

EntityLoadByPK

EntityMerge

EntityNew

EntityReload

EntitySave

EntityToQuery

Evaluate

Exp

ExpandPath

FileClose

GetTemplatePath

GetTickCount

GetTimeZoneInfo

GetToken

GetTotalSpace

GetUserRoles

GetVFSMetaData

GetWriteableImageFormats

Chapter 5: Ajax JavaScript Functions

Ajax JavaScript Functions

You can use the JavaScript functions listed below on pages that use ColdFusion Ajax features.

Function summary Ajax

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.Autosuggest.getAutosuggestObject	Lets you access underlying YUI AutoComplete object thereby providing fine-grained control over the object, for example attaching an event.
ColdFusion.FileUpload.cancelUpload	Cancels the file upload at any point during the file upload.
ColdFusion.FileUpload.clearAllFiles	Clears all the files selected for upload.

Coldfusion.fileUpload.setUrl	Sets URL for the fileupload control dynamically.
ColdFusion.FileUpload.startUpload	Starts uploading the selected files.
ColdFusion.getElementValue	Gets the value of an attribute of a bindable ColdFusion control.
ColdFusion.grid.clearSelectedRows	Clears the selected rows in the grid.
ColdFusion.Grid.getBottomToolbar	Gets bottom toolbar that can be used to add a control, for example icon or button.
ColdFusion.Grid.getGridObject	Gets the underlying Ext JS - JavaScript Library object for the specified HTML cfgrid control.
ColdFusion.grid.getSelectedRows	Fetches data for the selected rows in the grid.
ColdFusion.Grid.getTopToolbar	Gets the top toolbar that can be used to add a control, for example icon or button.
ColdFusion.Grid.hideBottomToolbar	Hides the bottom toolbar that can be used to add a control, for example icon or button.
ColdFusion.Grid.hideTopToolbar	Hides the top toolbar that can be used to add a control, for example icon or button.
ColdFusion.Grid.refresh	Manually refreshes a displayed grid.
ColdFusion.Grid.refreshBottomToolbar	Refreshes the bottom toolbar that can be used to add a control, for example icon or button.
ColdFusion.Grid.refreshTopToolbar	Refreshes the top toolbar that can be used to add a control, for example icon or button.
ColdFusion.Grid.showBottomToolbar	Shows bottom toolbar that can be used to add a control, for example icon or button.
ColdFusion.Grid.showTopToolbar	Displays the top toolbar that can be used to add a control, for example icon or button.
ColdFusion.Grid.sort	Sorts the specified HTML grid.
ColdFusion.JSON.decode	Converts a JSON-encoded string into a JavaScript variable
ColdFusion.JSON.encode	converts a JavaScript variable into a JSON string.
ColdFusion.Layout.collapseAccordion	Collapses an area of an accordion layout.
ColdFusion.Layout.collapseArea	Collapses an area of a border layout (cflayout tag with a type attribute of border).
ColdFusion.Layout.createAccordionPanel	Creates a panel in an existing accordion layout (cflayout tag with a type attribute of accordion).
ColdFusion.Layout.createTab	Creates a tab in an existing tabbed layout (cflayout tag with a type attribute of tab).
ColdFusion.Layout.disableSourceBind	Disables the source bind.
ColdFusion.Layout.disableTab	Disables the specified tab so it cannot be selected.
ColdFusion.Layout.enableSourceBind	If disabled, enables the source bind.
ColdFusion.Layout.enableTab	Enables the specified tab so users can select it and display the area contents.
ColdFusion.Layout.expandAccordion	Expands a collapsed area of an accordion layout.
ColdFusion.Layout.expandArea	Expands a collapsed area of a border layout.

ColdFusion.Layout.getAccordionLayout	Gets the underlying Ext JS - JavaScript Library object for the specified accordion type cflayout control.
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.hideAccordion	Hides an accordion.
ColdFusion.Layout.hideArea	Hides a bordered layout area.
ColdFusion.Layout.hideTab	Hides a tab.
ColdFusion.Layout.selectAccordion	Selects an accordion and displays the layout area contents.
ColdFusion.Layout.selectTab	Selects a tab and displays the layout area contents.
ColdFusion.Layout.showAccordion	Shows an accordion that was hidden using the inithide attribute or the hideArea() function.
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.Map.addEvent	Enables event handling in a map.
ColdFusion.Map.addMarker	Adds a marker to the map.
ColdFusion.Map.getLatitudeLongitude	Gets the latitude/longitude coordinates for a given address.
ColdFusion.Map.getMapObject	Gets the Google map component.
ColdFusion.Map.hide	Hides the map if displayed.
ColdFusion.Map.refresh	Reloads the map.
ColdFusion.Map.setCenter	Sets the center of map to the address that you specify.
ColdFusion.Map.setZoomlevel	Sets the zoom level of the map to the new value.
ColdFusion.Map.show	Shows the map if it is hidden.
ColdFusion.Mediaplayer.resize	Changes the current size of the media player.
ColdFusion.Mediaplayer.setMute	Mutes or unmutes the sound of the media player.
ColdFusion.Mediaplayer.setSource	Sets the URL of the FLV file.
ColdFusion.Mediaplayer.setVolume	Sets the volume of sound of the media player.
ColdFusion.Mediaplayer.startPlay	Plays the FLV file.
ColdFusion.Mediaplayer.stopPlay	Stops playing the FLV file.
ColdFusion.MessageBox.create	Creates a ColdFusion message box. Equivalent to the cfmessagebox tag.

ColdFusion.MessageBox.getMessageBoxObject	Gets the underlying Ext JS - JavaScript Library object for the specified HTML cfmessagebox control.
ColdFusion.MessageBox.isMessageBoxDefined	Checks if a message box is defined.
ColdFusion.MessageBox.show	Displays a ColdFusion message box.
ColdFusion.MessageBox.update	Updates message box properties.
ColdFusion.MessageBox.updateMessage	Updates the message property.
ColdFusion.MessageBox.updateTitle	Updates the message box title.
ColdFusion.navigate	Displays the output of a link URL in a specified cfdiv, cflayoutarea, cfpod, or cfwindow container.
ColdFusion.ProgressBar.getProgressBarObject	Gets the progress bar object.
ColdFusion.ProgressBar.hide	Hides the progress bar if it is displayed.
ColdFusion.ProgressBar.reset	Resets the progress.
ColdFusion.ProgressBar.show	Shows the progress bar if it is hidden.
ColdFusion.ProgressBar.start	Stops the underlying progress bar object that is running.
ColdFusion.ProgressBar.stop	Starts the underlying progress bar object.
ColdFusion.ProgressBar.update	Updates the attributes duration, interval, and oncomplete.
ColdFusion.ProgressBar.updatestatus	Lets you manually update the status and message of the progress bar.
ColdFusion.setGlobalErrorHandler	Replaces the global JavaScript error handler for displaying information about ColdFusion Ajax errors.
ColdFusion.Slider.disable	Disables the slider control.
ColdFusion.Slider.enable	Enables the slider control.
ColdFusion.Slider.getSliderObject	Gets the slider control.
ColdFusion.Slider.getValue	Gets the numeric value of the slider control.
ColdFusion.Slider.hide	Hides the slider control.
ColdFusion.Slider.setValue	Sets the numeric value of the slider control.
ColdFusion.Slider.show	Shows the slider control.
ColdFusion.Tree.getTreeObject	Gets the underlying Yahoo YUI Library object for the specified HTML cftree control.
ColdFusion.Tree.refresh	Manually refreshes a displayed HTML tree.
ColdFusion.Window.create	Creates a ColdFusion pop-up window. Equivalent to the cfwindow tag.
ColdFusion.Window.getWindowObject	Gets the underlying Ext JS - JavaScript Library object for the specified HTML 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 in Using Ajax form controls and features in the Developing ColdFusion Applications

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, 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 in Using Ajax form controls and features in the Developing ColdFusion Applications.

ColdFusion.Autosuggest.getAutosuggestObject

Description

Lets you access underlying YUI AutoComplete object thereby providing fine-grained control over the object, for example attaching an event.

Returns

The underlying AutoComplete object.

Function syntax

ColdFusion.Autosuggest.getAutosuggestObject (Id)

Parameters

- Id: Name of the auto-suggest object.

Example

```
<html> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<head> <cfajaximport tags="cfinput-autosuggest"> <script> var init = function() { autosuggestobj =
ColdFusion.Autosuggest.getAutosuggestObject('state'); autosuggestobj.itemSelectEvent.subscribe(foo); } var foo = function(event,args) { var
msg = ""; msg = msg + "Event: " + event + "\n\n"; msg = msg + "Selected Item: " + args[2] + "\n\n"; msg = msg + "Index: " + args[1]._nItemIndex
+ "\n\n"; alert(msg); } var getStates = function(){ return ["California","Connecticut","Colorado","Illinois","Alabama","Iowa","Utah", "Alaska"]; }
</script> </head> <body> <h3>Attaching an event handler to the autosuggest object</h3> <cfform name="mycfform" method="post" >
State:<BR> <cfinput type="text" name="state" autosuggest="javascript:getStates({cfautosuggestvalue})" autosuggestMinLength=1
autosuggestBindDelay=1> <cfset ajaxOnLoad("init")> </cfform> </body> </html>
```

See also

ColdFusion.Chart.getChartHandle

ColdFusion.FileUpload.cancelUpload

ColdFusion.FileUpload.clearAllFiles

Coldfusion.fileUpload.setUrl

ColdFusion.FileUpload.startUpload

ColdFusion.getElementValue

ColdFusion.grid.clearSelectedRows

ColdFusion.Grid.getBottomToolbar

ColdFusion.Grid.getGridObject

ColdFusion.grid.getSelectedRows

ColdFusion.Grid.getTopToolbar

ColdFusion.Grid.hideBottomToolbar

ColdFusion.Grid.hideTopToolbar

ColdFusion.Grid.refresh

ColdFusion.Grid.refreshBottomToolbar

ColdFusion.Grid.refreshTopToolbar

ColdFusion.Grid.showBottomToolbar

ColdFusion.Grid.showTopToolbar

ColdFusion.Grid.sort

ColdFusion.JSON.decode

ColdFusion.JSON.encode

ColdFusion.Layout.createTab

ColdFusion.Layout.collapseArea

ColdFusion.Layout.collapseAccordion

ColdFusion.Layout.createAccordionPanel

ColdFusion.Layout.disableSourceBind

ColdFusion.Layout.enableSourceBind

Description

If disabled, enables the source bind.

Function syntax

ColdFusion.Layout.enableSourceBind(Id)

Parameters

- Id: Name of the layout area.

Usage

See usage in ColdFusion.Layout.disableSourceBind .

Example

See example in ColdFusion.Layout.disableSourceBind .

ColdFusion.FileUpload.getSelectedFiles

Description

Returns an array of objects containing the filename and size of the files selected for upload. The file size is returned in bytes. The function also returns file upload status as YES|NO|Error.

Function syntax

ColdFusion.FileUpload.getSelectedFiles(Id)

Parameters

- Id: Name of the cffileupload control.

Usage

In a real life scenario, you normally use the uploader with other controls. For example, a form with three fields: name, email, and uploader. Assume that you upload the files, but forget to click Submit or you select the files, submit the form, but forget to click Upload. You can use this function to inform the user that there are files that have been selected for upload and provide the following details:

- FILENAME: Name of the file selected for upload.
- SIZE: Size of the file in bytes.
- STATUS: YES|NO|Error; YES indicates a successful upload, NO indicates that the upload is yet to occur, and Error indicates that an exception has occurred during the upload operation.

Example

The following example illustrates a scenario where the user clicks Submit and is informed about the files selected for upload:

```
<html> <head> <script language="javascript"> var formatNumber = function(num){ if(num < 1024) return num + " bytes"; if(num < (1024 * 1024)) return (num/1024).toFixed(2) + " KB"; if(num < (1024 * 1024 *1024)) return (num/(1024 * 1024)).toFixed(2) + " MB"; return (num/(1024 * 1024 * 1024)).toFixed(2) + " GB"; } var getSelectedList = function(id){ var files = ColdFusion.FileUpload.getSelectedFiles(id); var fileslist = ""; if(files.length) fileslist = "You have selected The following files for upload: \n\n"; for(var i=0;i < files.length; i++){ fileslist = fileslist + files[i].FILENAME + " (" + formatNumber(files[i].SIZE) + ")" if(i != files.length-1) fileslist = fileslist + "\r\n"; } if(files.length) { alert(fileslist); } } </script> </head> <body> <br> <cfform name="frmUpload" method="POST"> First Name: <cfinput type="text" name="fname" value=""><br> Last Name: <cfinput type="text" name="lname" value=""><br><br> <cffileupload url="uploadAll.cfm" name="myuploader1" hideUploadButton=false onUploadComplete="foo" /><br><br> <cfinput type="button" name="submitForm2" value="Submit" onClick="getSelectedList('myuploader1')"> </cfform> </body> </html>
```

See also

ColdFusion.Layout.expandAccordion

ColdFusion.Layout.disableTab

ColdFusion.Layout.enableTab

ColdFusion.Layout.expandArea

ColdFusion.Layout.getAccordionLayout

ColdFusion.Layout.getBorderLayout

ColdFusion.Layout.getTabLayout

ColdFusion.Layout.hideAccordion

ColdFusion.Layout.hideArea

ColdFusion.Layout.hideTab

ColdFusion.Layout.selectAccordion

ColdFusion.Layout.selectTab

ColdFusion.Layout.showAccordion

ColdFusion.Layout.showArea

ColdFusion.Layout.showTab

ColdFusion.Log.debug

ColdFusion.Log.dump

ColdFusion.Log.error

ColdFusion.Log.info

ColdFusion.Map.addEvent

ColdFusion.Map.addMarker

ColdFusion.Map.getLatitudeLongitude

ColdFusion.Map.getMapObject

ColdFusion.Map.hide

ColdFusion.Map.refresh

ColdFusion.Map.setCenter

ColdFusion.Map.setZoomlevel

ColdFusion.MessageBox.getMessageBoxObject

Description

Gets the underlying Ext JS - JavaScript Library object for the specified HTML cfmessagebox control.

Function syntax

```
ColdFusion.MessageBox.getMessageBoxObject(name)
```

See also

ColdFusion.MessageBox.create , ColdFusion.MessageBox.isMessageBoxDefined, ColdFusion.MessageBox.update , ColdFusion.MessageBox.updateMessage , ColdFusion.MessageBox.updateTitle

History

ColdFusion 9: Added this function

Parameters

Parameter	Description
name	The name of the message box object.

Returns

A JavaScript object.

Usage

Use this function to get the JavaScript object that contains all the defined properties. ----

ColdFusion.ProgressBar.getProgressBarObject

Description

Gets the progress bar object.

Function syntax

```
ColdFusion.ProgressBar.getProgressBarObject(name)
```

See also

ColdFusion.ProgressBar.start, ColdFusion.ProgressBar.stop

History

ColdFusion 9: Added this function

Parameters

Parameter	Description
name	The name of the progress bar object.

Returns

This function returns the underlying Ext JavaScript progress bar object.

Usage

You call this function to get the progress bar object. ----

See also

ColdFusion.ProgressBar.hide

ColdFusion.ProgressBar.reset

ColdFusion.ProgressBar.show

ColdFusion.ProgressBar.start

ColdFusion.ProgressBar.stop

ColdFusion.ProgressBar.update

ColdFusion.ProgressBar.updatestatus

ColdFusion.MessageBox.isMessageBoxDefined

Description

Checks if a message box is defined.

Function syntax

```
ColdFusion.MessageBox.isMessageBoxDefined(name)
```

See also

ColdFusion.MessageBox.create , ColdFusion.MessageBox.getMessageBoxObject, ColdFusion.MessageBox.update , ColdFusion.MessageBox.updateMessage , ColdFusion.MessageBox.updateTitle

History

ColdFusion 9: Added this function

Parameters

Parameter	Description
name	The name of the message box object.

Returns

A Boolean value, that is, true or false.

Usage

Use this function to check if the message box is defined for a specific name.

See also

- ColdFusion.MessageBox.update

- ColdFusion.MessageBox.updateMessage
- ColdFusion.MessageBox.updateTitle
- ColdFusion.navigate

JavaScriptFunctionsinColdFusion9Update1

The following are the Ajax JavaScript functions added in this release:

ColdFusion.Autosuggest.getAutosuggestObject

Description

Lets you access underlying YUI AutoComplete object thereby providing fine-grained control over the object, for example attaching an event.

Returns

The underlying AutoComplete object.

Function syntax

ColdFusion.Autosuggest.getAutosuggestObject (Id)

Parameters

- Id: Name of the auto-suggest object.

Example

```
<html> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<head> <cfajaximport tags="cfinput-autosuggest"> <script> var init = function() { autosuggestobj =
ColdFusion.Autosuggest.getAutosuggestObject('state'); autosuggestobj.itemSelectEvent.subscribe(foo); } var foo = function(event,args) { var
msg = ""; msg = msg + "Event: " + event + "\n\n"; msg = msg + "Selected Item: " + args[2] + "\n\n"; msg = msg + "Index: " + args[1]._nItemIndex
+ "\n\n"; alert(msg); } var getStates = function(){ return ["California","Connecticut","Colorado","Illinois","Alabama","Iowa","Utah", "Alaska"]; }
</script> </head> <body> <h3>Attaching an event handler to the autosuggest object</h3> <cfform name="mycfform" method="post" >
State:<BR> <cfinput type="text" name="state" autosuggest="javascript:getStates({cfautosuggestvalue})" autosuggestMinLength=1
autosuggestBindDelay=1> <cfset ajaxOnLoad("init")> </cfform> </body> </html>
```

ColdFusion.Layout.disableSourceBind

Description

Disables the source bind.

Function syntax

ColdFusion.Layout.disableSourceBind(Id)

Parameters

- Id: Name of the layout area.

Usage

Assume that you are using Coldfusion.navigate to populate content into tab or accordion panels. You can have instances where content comes from the source bind call if the source attribute is defined for cflayoutarea (and is not from ColdFusion.navigate).In such instances, you might disable the source bind to get content using Coldfusion.navigate.

Example

layout.cfm uses the templates Tab1_Src.cfm, Tab2_Src.cfm, and Tab3_Src.cfm. If you run layout.cfm, you notice that clicking

- navigate populates content of tab2_src.cfm instead of navigate.cfm
- Disable Source bind ensures that the content of navigate.cfm is populated in tab2_src
- Enable Source Bind and then clicking tab2_src would again populate the content of tab2_src**Tab1_Src.cfm**

```
<br><cfdump var="#CGI#" keys="15" label="[CGI scope]"><br>
```

Tab2_Src.cfm

```
<br><cfdump var="#server#" label="[Server scope]"><br>
```

Tab3_Src.cfm

```
<br><cfdump var="#server.coldfusion#" label="[Showing key coldfusion in server scope]"><br>
```

Tab4_Src.cfm__

```
<br><cfdump var="#server.os#" label="[Showing key OS in server scope]"><br>
```

layout.cfm

```
<script> var navigateToTab = function(layoutId,tabId){ alert("Navigating to " + tabId); ColdFusion.Layout.selectTab(layoutId,tabId);
ColdFusion.navigate('navigate.cfm',tabId); } var disableBind = function(tabId){ alert("Disabling binding on source for " + tabId);
ColdFusion.Layout.disableSourceBind(tabId); } var enableBind = function(tabId){ alert("Enabling binding on source for " + tabId);
ColdFusion.Layout.enableSourceBind(tabId); } </script> <cflayout type="tab" name="layout1"> <cflayoutarea name = "tab1" overflow =
"auto" refreshonactivate = "yes" title = "Tab 1" source = "tab1_src.cfm"/> <cflayoutarea name = "tab2" overflow = "auto" refreshonactivate
= "false" title = "Tab 2" source = "tab2_src.cfm" bindonload=false /> <cflayoutarea name = "tab3" overflow = "auto" refreshonactivate =
"yes" title = "Tab 3" source = "tab3_src.cfm" /> </cflayout> <cfform name="myform"> <cfinput type="button" name="disable"
value="Disable Source Bind" onClick="javascript:disableBind('tab2')"> <cfinput type="button" name="b" value="Navigate"
onClick="javascript:navigateToTab('layout1','tab2')"> <cfinput type="button" name="disable" value="Enable Source Bind"
onClick="javascript:enableBind('tab2')"> </cfform>
```

ColdFusion.Layout.enableSourceBind

Description

If disabled, enables the source bind.

Function syntax

ColdFusion.Layout.enableSourceBind(Id)

Parameters

- Id: Name of the layout area.

Usage

See usage in ColdFusion.Layout.disableSourceBind .

Example

See example in ColdFusion.Layout.disableSourceBind .

ColdFusion.FileUpload.getSelectedFiles

Description

Returns an array of objects containing the filename and size of the files selected for upload. The file size is returned in bytes. The function also returns file upload status as YES|NO|Error.

Function syntax

ColdFusion.FileUpload.getSelectedFiles(Id)

Parameters

- Id: Name of the cffileupload control.

Usage

In a real life scenario, you normally use the uploader with other controls. For example, a form with three fields: name, email, and uploader. Assume that you upload the files, but forget to click Submit or you select the files, submit the form, but forget to click Upload. You can use this function to inform the user that there are files that have been selected for upload and provide the following details:

- FILENAME: Name of the file selected for upload.
- SIZE: Size of the file in bytes.
- STATUS: YES|NO|Error; YES indicates a successful upload, NO indicates that the upload is yet to occur, and Error indicates that an exception has occurred during the upload operation.

Example

The following example illustrates a scenario where the user clicks Submit and is informed about the files selected for upload:

```
<html> <head> <script language="javascript"> var formatNumber = function(num){ if(num < 1024) return num + " bytes"; if(num < (1024 * 1024)) return (num/1024).toFixed(2) + " KB"; if(num < (1024 * 1024 *1024)) return (num/(1024 * 1024)).toFixed(2) + " MB"; return (num/(1024 * 1024 * 1024)).toFixed(2) + " GB"; } var getSelectedList = function(id){ var files = ColdFusion.FileUpload.getSelectedFiles(id); var fileslist = ""; if(files.length) fileslist = "You have selected The following files for upload: \n\n"; for(var i=0;i < files.length; i++){ fileslist = fileslist + files[i].FILENAME + " (" + formatNumber(files[i].SIZE) + ")" if(i != files.length-1) fileslist = fileslist + "\r\n"; } if(files.length) { alert(fileslist); } } </script> </head> <body> <br> <cfform name="frmUpload" method="POST"> First Name: <cfinput type="text" name="fname" value=""><br> Last Name: <cfinput type="text" name="lname" value=""><br><br> <cffileupload url="uploadAll.cfm" name="myuploader1" hideUploadButton=false onUploadComplete="foo" /><br><br> <cfinput type="button" name="submitForm2" value="Submit" onClick="getSelectedList('myuploader1')"> </cfform> </body> </html>
```

Coldfusion.fileUpload.setUrl

Description

Used to set URL for the fileupload control dynamically.

Returns

Nothing

Function syntax

ColdFusion.fileUpload.setUrl(id, url)

Parameters

- Id: Name of upload control.
- Url: URL can be an absolute URL, relative URL, or fully qualified URL.

Example

```
<script language="javascript"> var uploadDone = function(result){ alert("File uploaded"); } var setUploadUrl = function(id) { var selectedFiles = ColdFusion.FileUpload.getSelectedFiles(id); var uploadUrl = "/manual/ajaxui/cffileupload/setUrl/includes/_uploadall.cfm"; alert("Upload URL : " + uploadUrl); if(selectedFiles.length){ ColdFusion.FileUpload.setURL(id,uploadUrl); ColdFusion.FileUpload.startUpload(id); } } var callbackhandler = function(obj) { var fileName = obj["FILENAME"]; var status = obj["STATUS"]; var message = obj["MESSAGE"]; var msg = "In callbackhandler()" + "\n\n" + "FILENAME: " + fileName + "\n\n" + "STATUS: " + status + "\n\n" + "MESSAGE: " + message alert(msg); } var errorhandler = function() { alert("In errorhandler()"); } var uploadcompleted = function() { alert("All files have been uploaded successfully"); } </script> <cfform name="frmUpload"> <br> <cffileupload name="uploader" hideuploadbutton="true" onComplete="uploadDone" onError="errorhandler" onUploadComplete="uploadcompleted"> <br> <cfinput type="button" name="submit" value="Click to set URL and Upload Files" onClick="setUploadUrl('uploader')"> </cfform>
```

ColdFusion.grid.getSelectedRows

Description

Used to fetch data for the selected rows in the grid.

Returns

An array of objects that contains row data.

Function syntax

ColdFusion.grid.getSelectedRows(id)

Parameters

- Id: Name of the grid defined using cfgrid.

See also

FileUpload

Usage

See the example in ColdFusion.grid.clearSelectedRows .

Example

See the example in ColdFusion.grid.clearSelectedRows .

ColdFusion.grid.clearSelectedRows

Description

Used to clear the selected rows in the grid.

Returns

Nothing

Function syntax

ColdFusion.grid.clearSelectedRows(id)

Parameters

- Id: Name of the grid defined using cfgrid.

Usage

See the following example.

Example

Employee.cfm

```
<html> <head> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <cfajaxproxy cfc="emp" jsclassname="emputils"> <script language="javascript"> var emp = new emputils(); var deleteAllSelectedRows = function(grid) { emp.setHTTPMethod("POST"); emp.deleteEmployees(getAllSelectedRows(grid,false)); ColdFusion.Grid.refresh(grid); } var getAllSelectedRows = function(grid,showalert) { obj = ColdFusion.Grid.getSelectedRows(grid); jsonbj = ColdFusion.JSON.encode(obj); if(showalert) alert(jsonbj); return obj; } var clearAllSelectedRows = function(grid) { ColdFusion.Grid.clearSelectedRows(grid); } </script> </head> <body> <cfform> <cfgrid format="html" name="empListing" selectmode="edit" bind="cfc:emp.getEmployees({cfgridpage},{cfgridpagesize},{cfgridsortcolumn},{cfgridsortdirection})" onchange="cfc:emp.editEmployees({cfgridaction},{cfgridrow},{cfgridchanged})" autowidth="true" multirowselect=true delete="true" insert="true" title="Employee database" pagesize="25" > <cfgridcolumn name="EMP_ID" header="EMP_ID" select="false" display="false"> <cfgridcolumn name="FIRSTNAME" header="First Name" select="true" /> <cfgridcolumn name="LASTNAME" header="Last Name" select="true" /> <cfgridcolumn name="DEPARTMENT" header="Department" select="true" /> <cfgridcolumn name="EMAIL" header="Email" select="true" /> </cfgrid> <br> <cfinput type="button" onClick="javascript:getAllSelectedRows('empListing',true)" name="getRows" value="Get Selected Rows"> <cfinput type="button" onClick="javascript:clearAllSelectedRows('empListing')" name="clearRows" value="Clear Selected Rows"> <cfinput type="button" onClick="javascript:deleteAllSelectedRows('empListing')" name="deleteRows" value="Delete Selected Rows"> </cfform> </body> </html>
```

Employee.cfc

```
<cfcomponent> <cfscript> empQuery = new query(name="emps", datasource="cfdocexamples"); remote any function getEmployees(page,pageSize,gridsortcolumn="EMP_ID",gridsortdirection="ASC",empName) { var orderBy = "EMP_ID"; var mysql = "SELECT Emp_ID, FirstName, LastName, EMail, Department, Email FROM Employees"; if(isdefined("arguments.empName") and trim(arguments.empName) neq ""){ mysql = mysql & " WHERE " & "firstname = '#arguments.empName#'"; } if(arguments.gridsortcolumn eq ""){ mysql = mysql & " ORDER BY " & orderBy; } mysql = mysql & " " & gridsortdirection; return QueryConvertForGrid(empQuery.execute(sql=mysql).getResult(), page, pageSize); } remote void function editEmployees(gridaction,gridrow,gridchanged) { switch(gridaction) { case "I": { var eFName = gridrow["FIRSTNAME"]; var eLNAme = gridrow["LASTNAME"]; var eDept = gridrow["DEPARTMENT"]; var eEmail = gridrow["EMAIL"]; var insertSql = "insert into Employees(FirstName,LastName,Department,Email) values (" & "'" & eFName & "', '" & eLName & "', '" & eDept & "', '" & eEmail & "')"; empQuery.execute(sql=insertSql); break; } case "U": { var empId = gridrow["EMP_ID"]; var changedCol = structkeylist(gridchanged); var updateSql = "UPDATE Employees SET " & changedCol & "='" & gridchanged[changedCol] & "' WHERE emp_id=" & empId; empQuery.execute(sql=updateSql); break; } case "D": { deleteEmployees(gridrow); } } } remote void function deleteEmployees(empdata) { var i = 1; var emp = {}; if(isArray(empdata) and not ArrayIsEmpty(empdata)){ for(emp in empdata){ if(isStruct(emp) and structkeyexists(emp,"emp_id")){ empid = emp["emp_id"]; writelog("deleting " & empid); //var deleteSql = "delete from Employees where emp_id=" & empid; //empQuery.execute(sql=deleteSql); } } } } </cfscript> </cfcomponent>
```

In this example, setting multirowselect=true enables performing of batch operations on grid data, such as deleting multiple records.In the deleteemployees functions, two lines have been commented out to prevent accidental deletion of data (since it is a batch operation). To see deletion, uncomment the code.The form has a deleteAllSelectedRows button that illustrates how records can be deleted externally. That is, without using the delete button built in to the grid. The same approach can be used to perform other batch operations such as moving multiple files to another folder or batch updates.

Note: Set the httpMethod to POST on the Proxy object carefully to avoid "request URI too large" errors as shown in the deleteAllSelectedRows method in Employee.cfm.

ColdFusion.Map.show

Description

Shows the map if it is hidden.

Function syntax

ColdFusion.Map.show(Id)

Parameters

- Id: Name of the map.

Example

```
<script> function showMap(mapId) { ColdFusion.Map.show(mapId); } function hideMap(mapId) { ColdFusion.Map.hide(mapId); } </script> <a href="##" id="a1" onclick="return showMap('mainMap')">Show Map</a> | <a href="##" id="a1" onclick="return hideMap('mainMap')">Hide Map</a> <cfmap zoomlevel = "12" name = "mainMap" showcentermarker= "true" centeraddress = "The Key Learning centre, Oxford, UK" title="Venue Address" hideborder=false collapsible=true initShow=false/>
```

ColdFusion.Map.hide

Description

If displayed, hides the map.

Function syntax

ColdFusion.Map.hide(Id)

Parameters

- Id: Name of the map.

Example

See example in ColdFusion.Map.show

ColdFusion.Map.refresh

Description

Reloads the map.

Function syntax

ColdFusion.Map.refresh (Id)

Parameters

- Id: Name of the map.

Usage

If the map is embedded within spry collapsible panels or divs that are hidden on display, that is the map container is displayed while the actual map is hidden, use this function to force the map to display.

Example

```
<script type="text/javascript" src="/CFIDE/scripts/ajax/spry/includes_minified/SpryCollapsiblePanel.js" ></script> <link type="text/css" href="/CFIDE/scripts/ajax/spry/widgets/collapsiblepanel/SpryCollapsiblePanel.css" rel="stylesheet"> <div id="cp" class="CollapsiblePanel" style="width:500px;"> <div class="CollapsiblePanelTab" tabindex="0">SHOW MAP</div> <div class="CollapsiblePanelContent"> <cfmap width="500" height="200" zoomlevel="12" name="mainMap" markercolor="333444" showscale="false" typecontrol="none" showcentermarker="true" centeraddress="The Key Learning centre, Oxford, UK" > </cfmap> </div> </div> <script type="text/javascript"> var myTabClick = function() { !cpanel.isOpen() ? cpanel.open() : cpanel.close(); cpanel.focus(); ColdFusion.Map.refresh('mainMap'); } var cpanel = new Spry.Widget.CollapsiblePanel("cp", {contentIsOpen:false}); cpanel.onTabClick = myTabClick; </script>
```

ColdFusion.Grid.getTopToolbar

Description

Gets the top toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.getTopToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar .

ColdFusion.Grid.getBottomToolbar

Description

Gets bottom toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.getBottomToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar .

ColdFusion.Grid.showTopToolbar

Description

Displays the top toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.showTopToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar .

ColdFusion.Grid.hideTopToolbar

Description

Hides the top toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.hideTopToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar .

ColdFusion.Grid.showBottomToolbar

Description

Shows bottom toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.showBottomToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar .

ColdFusion.Grid.hideBottomToolbar

Description

Hides the bottom toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.hideBottomToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar .

ColdFusion.Grid.refreshTopToolbar

Description

Refreshes the top toolbar that can be used to add a control, for example icon or button. This function internally calls the JavaScript function ColdFusion.Grid.showTopToolbar .

Function syntax

ColdFusion.Grid.refreshTopToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar .

ColdFusion.Grid.refreshBottomToolbar

Description

Refreshes the bottom toolbar that can be used to add a control, for example icon or button. This function internally calls the JavaScript function ColdFusion.Grid.showBottomToolbar .

Function syntax

ColdFusion.Grid.refresheBottomToolbar(Id)

Parameters

- Id: Name of the grid control.

Example

grid.cfc

```
<cfcomponent> <cfscript> remote any function getEmployees(page,pageSize,gridsortcolumn="EMP_ID",gridsortdirection="ASC"){ var
startRow = (page-1)*pageSize; var endRow = page*pageSize; if(!isdefined("arguments.gridsortcolumn") or
isdefined("arguments.gridsortcolumn") and trim(arguments.gridsortcolumn) eq "") gridsortcolumn = "EMP_ID";
if(!isdefined("arguments.gridsortdirection") or isdefined("arguments.gridsortdirection") and arguments.gridsortdirection eq "")
gridsortdirection = "ASC"; var mysql = "SELECT Emp_ID, FirstName, EMail, Department FROM Employees";
if(isdefined("arguments.gridsortcolumn") and arguments.gridsortcolumn neq "") mysql = mysql & " ORDER BY " & gridsortcolumn;
if(isdefined("arguments.gridsortdirection") and arguments.gridsortdirection neq "") mysql = mysql & " " & gridsortdirection ; rs1 = new
query(name="team", datasource="cfdocexamples", sql=mysql).execute(); return QueryConvertForGrid(rs1.getResult(), page, pageSize); }
remote any function editEmployees(gridaction,gridrow,gridchanged){ writelog("edit employee info"); } </cfscript> </cfcomponent>
```

grid.cfm

```
<script> var refreshToolbar = function(id,type){ type == "top" ? ColdFusion.Grid.refreshTopToolbar(id) :
ColdFusion.Grid.refreshBottomToolbar(id); } var hideToolbar = function(id,type){ type == "top" ? ColdFusion.Grid.hideTopToolbar(id) :
ColdFusion.Grid.hideBottomToolbar(id); } var showToolbar = function(id,type){ (type == "top") ? ColdFusion.Grid.showTopToolbar(id) :
ColdFusion.Grid.showBottomToolbar(id); } var handleToolbar = function(id,type){ if(type == "top"){ tbar = ColdFusion.Grid.getTopToolbar(id);
tbar.addButton({ text: "Add User Account", tooltip: "Add a user account", handler: addUserAccount }); } else{ bbar =
ColdFusion.Grid.getBottomToolbar(id); bbar.add(new Ext.Toolbar.Separator()); bbar.addButton({ text: "Delete User Account", tooltip: "Delete a
user account", handler: deleteUserAccount }); } } var GetUserInfo = function(){ alert("Retrieving user account"); } var addUserAccount =
function(){ alert("Adding new user account") } var deleteUserAccount = function(){ alert("Deleting user account") } </script> <cfform> <br>
<cfinput type="button" onClick="showToolbar('empGrid','top')" name="btn1" value="Show Top Toolbar"> <cfinput type="button"
onClick="handleToolbar('empGrid','top')" name="btn2" value="Add button to Top Toolbar"> <cfinput type="button"
onClick="refreshToolbar('empGrid','top')" name="btn3" value="Refresh Top Toolbar"> <cfinput type="button"
onClick="hideToolbar('empGrid','top')" name="btn4" value="Hide Top Toolbar"> <br><br> <cfgrid format="html" name="empGrid"
width="800" pagesize=5 sort=true title="Employee database" collapsible="true" insert="yes" delete="yes"
bind="cfc:grid.getEmployees({cfgridpage},{cfgridpagesize},{cfgridsortcolumn},{cfgridsortdirection})"
onChange="cfc:grid.editEmployees({cfgridaction},{cfgridrow},{cfgridchanged})" selectMode="edit" > <cfgridcolumn name="Emp_ID"
display=false header="ID" /> <cfgridcolumn name="FirstName" display=true header="First Name"/> <cfgridcolumn name="Email"
display=true header="Email"/> <cfgridcolumn name="Department" display=true header="Department" /> </cfgrid> <br><br> <cfinput
type="button" onClick="hideToolbar('empGrid','bottom')" name="btn5" value="Hide Bottom Toolbar"> <cfinput type="button"
onClick="showToolbar('empGrid','bottom')" name="btn6" value="Show Bottom Toolbar"> <cfinput type="button"
onClick="handleToolbar('empGrid','bottom')" name="btn7" value="Add button to Bottom Toolbar"> <cfinput type="button"
onClick="refreshToolbar('empGrid','bottom')" name="btn8" value="Refresh Bottom Toolbar"> </cfform>
```

Chapter 6: Script Functions Implemented as CFCs

Script Functions Implemented as CFCs

Script functions were added in ColdFusion 9. They are implemented as ColdFusion Components. These functions extend the usage of the tags cfmail, cfpdf, cfquery, cfhttp, cfstoredproc, and cfftp to CFScript.

Accessing the functions

Script functions are available in the following location: cf_root\CustomTags\com\adobe\coldfusion. Ensure that you do not delete the default custom tag mapping listed in the ColdFusion Administrator (Extensions > Custom Tag Paths > Custom tag mappings). Script functions work if they are either in the default location or web root. If you have the functions in any other location, add a /com mapping in the ColdFusion Administrator that points to the new location (for example C:\com).

Note: Values of the attributes set in a service action, for example, mail.send(body="test mail") are transient in nature. They are not accessible after the action completes. Accessing the attributes using implicit getters results in error whereas any attributes set using either implicit setters or the init method call are retained and can be accessed using implicit getters.

Function summary

The following table lists the script functions and the equivalent ColdFusion tag.

Function	Equivalent ColdFusion Tag
ftp	cfftp
http	cfhttp
mail	cfmail
pdf	cfpdf
query	cfquery
storedproc	cfstoredproc

ftp

Description

Used to implement File Transfer Protocol (FTP) operations using CFScript.

Syntax

Mode	Syntax
Creating the service	new ftp() or createObject("component","ftp")
Initializing the attributes	Any one of the following: • ftpService=new ftp(attribute-value pair) • ftpService.setAttributes(_attribute-value pair_) • ftpService.setA_ttributeName_(attribute_value) • ftpService.action_method(attribute-value_pair)
Executing the service action	ftpService.action_method(attribute-value_pair)

Properties

actionparam	buffersize	connection	passive
password	port	proxyserver	retrycount
server	stoponerror	timeout	username
fingerprint	key	passphrase	secure
ASCIIExtensionList	directory	existing	failifexists
item	localfile	name	new
remotefile	result	transfermode	allosize

All attributes supported by the tag cfftp can be used as attribute-value pairs. For example,

```
<cfftp userName="myUserName">
```

can be used as

```
ftpService.setUserName("myUserName");
```

For details, see the Attributes section for the cfftp tag.

See also

cfftp , Function summary

History

ColdFusion 9: Added this function.

Methods

The following FTP actions are available as methods. All methods have similar arguments and syntax.

•

open	close	quote	site
allo	acct	changeDir	createDir
listDir	removeDir	getFile	putFile
rename	remove	getCurrentDir	getCurrentUrl
existDir	existsFile	exists	

•

Description	All methods correspond to the FTP actions supported by the tag cfftp. For details of each method, refer to the relevant section for the tag cfftp .
Returns	All methods return a component with the following properties set:

- prefix: Equivalent to the result attribute or cfftp scope
- result: Applicable only for action="listdir"

Syntax	ftpService.methodName(attribute-value pair)
Arguments	All attributes supported by the tag cfftp.

- setAttributes

Description	Sets attributes for the ftp function.
Returns	Nothing
Syntax	ftpService.setAttributes (attribute-value pair)
Arguments	All attributes supported by the tag cfftp.

- getAttributes

Description	Gets the attributes that were set for the ftp function.
Returns	Returns a struct with all or some attribute values.
Syntax	ftpService.get_Attributes_ (attributelist)
Arguments	A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

- clear

Description	Removes all attributes added for the ftp function.
Returns	Nothing
Syntax	ftpService.clear()
Arguments	None

Usage

This function corresponds to the cfftp tag. For details, see the Usage section for the tag cfftp .

Example

```
<cfscript> /* Create a new ftp Service*/ ftpService = new ftp(); /* Set attributes using implicit setters */ ftpService.setUsername("myUsername"); ftpService.setPassword("myPassword"); ftpService.setServer("myFtpServer"); ftpService.setStopOnError("true"); ftpService.setConnection("conn"); /* Open connection to ftp server */ WriteOutput("<h4>Open a connection</h4>"); result = ftpService.open(); WriteOutput("<p>Did it succeed? " & result.getPrefix().succeeded & "<br></p>"); /* Get current directory */ WriteOutput("<h4>Get current directory</h4>"); result = ftpService.getcurrentdir(); WriteOutput("<p>Current Directory: " & """ & result.getPrefix().returnvalue & """ & "<br></p>"); /* List contents of the current directory */ WriteOutput("<h4>List directory contents</h4>"); result = ftpService.listdir(directory = "/",name="listDirs"); displayListing(result.getResult()); /* Move a file to the ftp server */ WriteOutput("<h4>Move File to Remote Server</h4>"); lFile = "C:\temp\artifacts.xml"; rFile = "artifacts.xml"; result = ftpService.putFile(transferMode="binary", localfile=lFile, remoteFile=rFile); WriteOutput("<p>Did it succeed? " & result.getPrefix().succeeded & "<br></p>"); /* Close connection to the ftp server */ WriteOutput("<h4>Close the connection</h4>"); ftpService.close(connection="conn"); WriteOutput("<p>Did it succeed? " & result.getPrefix().succeeded & "<br></p>"); </cfscript> <cffunction name="displayListing" hint="display ftp files"> <cfargument name="filesToList" required="true"> <cftable query = "filesToList" HTMLTable = "Yes" colHeaders = "Yes" border="1" maxrows="10"> <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> </cffunction>
```

http

Description

Used in CFScript to generate an HTTP request and handle the response from the server.

Syntax

Mode	Syntax
Creating the service	new http() or createObject("component","http")
Initializing the attributes	Any one of the following: • httpService=new http(_attribute-value_pair_) • httpService.setAttributes(_attribute-value_pair_) • httpService.set_AttributeName_(attribute_value) • httpService.send(attribute-value_pair)
Executing the service action	httpService.send(_attribute-value_pair_)

Properties

url	charset	clientcert	clientcertpassword
columns	delimiter	file	firstrowasheaders
getasbinary	method	multipart	multiparttype
name	password	path	port
proxyserver	proxyport	proxyuser	proxypassword
redirect	resolveurl	result	textqualifier
throwonerror	timeout	useragent	username

All attributes supported by the tag cfhttp can be used as attribute-value pairs. For example,

```
<cfhttp name="onerow">
```

can be used as

```
httpService.setName("onerow");
```

For details of the attributes, see the Attributes section for the tag cfhttp .

See also

cfhttp __, Function summary

History

ColdFusion 9: Added this function.

Methods

- addParam

Description	Used to add cfhttpparam tags. For example, to specify http POST operations in CFScript. Specifies parameters to build an HTTP request.
Syntax	httpService.addParam(attribute-value pair)
Returns	Nothing
Arguments	All attributes supported by cfhttpparam tag can be used as attribute-value pairs.

- send

Description	Used to generate an HTTP request and handle the response from the server.
Returns	A component on which the following methods can be invoked:

- GetResult(): To access the query object returned by the server if a name attribute is specified.
- GetPrefix(): To access the cfhttp scope. This is equivalent to the result attribute of the cfhttptag.

Syntax	httpService.send(attribute_value pair)
Arguments	All attributes supported by the cfhttpparam tag.

- setAttributes

Description	Sets attributes for the http function.
Returns	Nothing
Syntax	httpService.setAttributes (attribute-value pair)
Arguments	All arguments supported by the cfhttp tag.

- getAttributes

Description	Gets attributes that were set for the http function.
Returns	Returns a struct with all or some of the service tag attribute values.
Syntax	httpService.get_Attributes_(attribute_list)
Arguments	A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

- clearAttributes

Description	Removes all attributes added for the http function.
Returns	Nothing
Syntax	httpService.clearAttributes(attribute_list)
Arguments	A comma-separated list of attributes.

- clearParams

Description	Removes cfhttpparam tags that were added using the addParam method.
Returns	Nothing
Syntax	httpService.clearParams()
Arguments	None

- clear

Description	Removes all attributes and cfhttpparam tags that were added using the addParam method.
Returns	Nothing
Syntax	httpService.clear()
Arguments	None

Usage

This function corresponds to the cfhttp tag. For usage details, see the Usage section for cfhttp in the CFML Reference.

Example

```
<!--- Get Video ---> <!--- <cfset videoName = "<video path>\hello.wmv"> <cfset videoFileName = "hello.wmv"> ---> <!--- Set User Account Data ---> <!--- <cfset clientKey = "enter client key from google"/> <cfset devKey = "ebtdev key from google>"/> ---> <cfscript> /* youtube uplaod url */ youTubeUploadURL = "http://uploads.gdata.youtube.com/feeds/api/users/default/uploads"; /* video to upload */ videoName = ExpandPath('./hello.wmv'); videoFileName = "hello.wmv"; /* set user account data */ clientKey = "enter client key from google"; devKey = "ewnter dev key from google"; /* create new http service */ httpService = new http(); /* set attributes using implicit setters */ httpService.setMethod("post"); httpService.setCharset("utf-8"); httpService.setUrl("https://www.google.com/accounts/ClientLogin"); /* add httpparams using addParam() */ httpService.addParam(type="formfield",name="accountType",value="HOSTED_OR_ GOOGLE");
httpService.addParam(type="formfield",name="Email",value="enter gmail id");
httpService.addParam(type="formfield",name="Passwd",value="enter password");
httpService.addParam(type="formfield",name="service",value="youtube");
httpService.addParam(type="formfield",name="source",value="youtubecode"); /* make the http call to the URL using send() */ result = httpService.send().getPrefix(); /* process the filecontent returned */ content = listtoarray(result.filecontent,chr(10)); for(i=1;i lte arraylen(content);i++) { item = content[i]; authdata[listFirst(item, "=")] = listRest(item, "="); } </cfscript> <!--- Create ATOM XML and save to a file to be sent with video ---> <cfsavecontent variable="meta"> <cfoutput> <entry xmlns="http://www.w3.org/2005/Atom" xmlns:media="http://search.yahoo.com/mrss/" xmlns:yt="http://gdata.youtube.com/schemas/2007"> <media:group> <media:title type="plain">WithOutQuotes</media:title> <media:description type="plain">Test Description</media:description> <media:category scheme="http://gdata.youtube.com/schemas/2007/categories.cat">People </media:category>
<media:keywords>yourvideo</media:keywords> </media:group> </entry> </cfoutput> </cfsavecontent> <cfscript> tmpfile = expandPath("./meta.xml"); FileWrite(tmpfile,trim(meta)); /* use the httpService created above */
httpService.setUrl("http://uploads.gdata.youtube.com/feeds/api/users/default/ uploads"); httpService.setTimeOut(450);
httpService.setMultipartType("related"); /* clear params first */ httpService.clearParams(); /* add httpparams using addParam() */
httpService.addParam(type="header",name="Authorization", value="GoogleLogin auth=#authdata.auth#");
httpService.addParam(type="header",name="X-GData-Client", value="#variables.clientkey#");
httpService.addParam(type="header",name="X-GData-Key", value="key=#variables.devkey#");
httpService.addParam(type="header",name="Slug",value="#videoFileName#");
httpService.addParam(type="file",name="API_XML_Request",file="#tmpfile#", mimetype= "application/atom+xml");
httpService.addParam(type="file",name="file",file="#videoName#",mimetype ="video/*"); /* make the http call to the URL using send() */ result = httpService.send().getPrefix(); if(result.statuscode contains "201") { WriteOutput("Your video has been successfully uploaded to YouTube"); } else { WriteOutput("There was a problem uploading the video. Status code returned was " & result.statuscode); } </cfscript>
```

mail

Description

Used to sends an e-mail message, that optionally contains query output, using an SMTP server.

Syntax

Mode	Syntax
Creating the service	new mail() or createObject("component", "mail")
Initializing the attributes	Any one of the following: • mailService=new mail(_attribute-value_pair_) • mailService.setAttributes(_attribute-value_pair_) • mailService.set_AttributeName_(attribute_value) • mailService.send(attribute-value_pair)
Executing the service action	mailService.send(_attribute-value_pair_)

Properties

from	to	subject	bcc
cc	charset	debug	failto
group	groupcasesensitive	mailerid	maxrows
mimeattach	password	port	priority
query	replyto	server	spoolenable
startrow	timeout	type	username
useSSL	useTLS	wraptext	remove
body			

All attributes supported by the tag cfmail can be used as attribute-value pairs. For example,

```
<cfmail from="#form.mailFrom#">
```

can be used as

```
mailerService.setFrom(form.mailFrom);
```

See also

cfmail , Function summary

History

ColdFusion 9: Added this function.

Methods

- addParam

Description	Used to add cfmailparam tags. For example, to attach a file or add a header to an e-mail message.
Syntax	mailService.addParam(attribute-value pair)
Returns	Nothing
Arguments	All attributes supported by the cfmailparam tag can be used as attribute-value pairs.

- addPart

Description	Used to add cfmailpart tags. For example, one part of a multipart e-mail message.
Syntax	mailService.addPart(attribute-value pair)
Returns	Nothing
Arguments	All attributes supported by the cfmailpart tag can be used as attribute-value pairs.

- send

Description	Used to invoke the mail service to send an e-mail message.
Returns	Nothing
Syntax	mailService.send(attribute-value pair)
Arguments	All attributes supported by the cfmail tag.

- setAttributes

Description	Sets attributes for the mail function.
Returns	Nothing
Syntax	mailService.setAttributes (attribute-value pair)
Arguments	All attributes supported by the cfmail tag.

- getAttributes

Description	Gets attributes that were set for the mail function.
Returns	Returns a struct with all or some of the attribute values.
Syntax	mailService.get_Attributes_ (attributelist)
Arguments	A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

- clearAttributes

Description	Removes all attributes added for the mail function.
Returns	Nothing
Syntax	mailService.clearAttributes(attribute_list)
Arguments	A comma-separated list of attributes.

- clearParams

Description	Removes cfmailparam tags that were added using the addParam method.
Returns	Nothing
Syntax	mailService.clearParams()
Arguments	None

- clearParts

Description	Removes cfmailpart tags that were added using the addPart method.
Returns	Nothing
Syntax	mailService.clearProcResults()
Arguments	None

- clear

Description	Removes all attributes, cfmailparam tags, and cfmailpart tags that were added using the methods addParam and addPart.
Returns	Nothing
Syntax	mailService.clear()
Arguments	None

Usage

This function corresponds to the tag cfmail. For usage details, see the Usage section for cfmail .

Example

```
<h3>Sending mail in cfscript</h3> <cfscript> /* create mailer service */ mailerService = new mail(); if(IsDefined("form.mailto")) { if(form.mailto is not "" AND form.mailfrom is not "" AND form.Subject is not "" and form.attachment is not "") { savecontent variable="mailBody"{ WriteOutput("This message was sent by an automatic mailer built with cfmail:==========" & "<br><br>" & form.body); } /* set mail attributes using implicit setters provided */ mailerService.setTo(form.mailto); mailerService.setFrom(form.mailFrom); mailerService.setSubject(form.subject); mailerService.setType("html"); /* add mailparams */ mailerService.addParam(file=expandpath(form.attachment),type="text/plain",remove =false); /* send mail using send(). Attribute values specified in an end action like "send" will not persist after the action is performed */ mailerService.send(body=mailBody); writeoutput("<h3>Thank you</h3>" & "<p>Thank you, " & mailfrom & "<br>" & "Your message, " & subject & ", has been sent to " & mailto & "</p>"); } } </cfscript> <p> <form action = "mail1.cfm" method="POST"> <table> <tr> <td>TO</td> <td><input type = "Text" name = "MailTo"></td> </tr> <tr> <td>FROM</td> <td><input type = "Text" name = "MailFrom"></td> </tr> <tr> <td>SUBJECT</td> <td><input type = "Text" name = "Subject"></td> </tr> <tr> <td>ATTACHMENT</td> <td><input type = "file" name = "attachment"></td> </tr> </table> <hr> MESSAGE BODY: <br> <textarea name ="body" cols="40" rows="5" wrap="virtual"></textarea> <!--- 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"> <input type = "hidden" name = "attachment_required" value = "You must select a file"> <p><input type = "Submit" name = ""></p> </p> </form>
```

pdf

Description

Provides services to manipulate existing PDF documents in CFScript.

Syntax

Mode	Syntax
Creating the service	new pdf()}}or {{createObject("component", "pdf")
Initializing the attributes	Any one of the following: • pdfService=new pdf(attribute-value pair) • pdfService.setAttributes(attribute-value pair) • pdfService.set_AttributeName(attribute_value_) • pdfService.action_method(attribute-value pair)
Executing the service action	pdfService.action_method(attribute-value pair)

Properties

addQuads	algo	align	ascending
bottomMargin	compressTiffs	copyFrom	ddxfile
destination	directory	encodeAll	encrypt
flatten	foreground	format	height
hires	honourSpaces	hScale	image
imagePrefix	info	inputFiles	isBase64
jpgDpi	keepBookmark	leftMargin	maxBreadth
maxLength	maxScale	name	newOwnerPassword
newUserPassword	noAttachments	noBookmarks	noComments
noJavascripts	noLinks	noMetadata	noThumbnails
numberFormat	opacity	order	outputFiles
overridePage	overwrite	package	pages
password	permissions	position	resolution
rightMargin	rotation	saveOption	scale
showOnPrint	source	stopOnError	text
topMargin	transparent	type	useStructure
version	vscale	width	

All attributes supported by the tag cfpdf can be used as attribute-value pairs. For example,

```
<cfpdf action="getInfo" source="myBook.pdf" name="PDFInfo">
```

can be used as

```
pdfInfo = pdfService.getPdfInfo(source="myBook.pdf", name="pdfinfo");
```

For details, see the Attributes section for the cfpdf tag__.

Methods

- addParam

Description	Used in CFScript to add cfpdfparam tags. Applicable only to action="merge".
Returns	Nothing
Syntax	pdfService.addParam(attribute-value pair)
Arguments	All attributes supported by the cfpdfparam tag can be used as attribute-value pairs.

- The following PDF actions are available as methods. All these methods have similar arguments and syntax.

addWatermark	removeWatermark	deletePages	getPDFInfo
setPDFInfo	merge	processDDX	protect
read	write	thumbnail	transform
optimize	extractImage	extractText	addHeader
addFooter	removeHeaderFooter		

Note: In the list, setPDFInfo and getPDFInfo do not have identical actions in cfpdf. cfpdf action="setinfo" and cfpdf action="getinfo" represent them respectively.

Description	All methods correspond to the PDF actions specified for the tag cfpdf. For details of each method, refer to the corresponding section for cfpdf .
Returns	Depends on the action. If the name attribute is specified, the result of the pdf operation is returned. Else, an empty string. For example, the following code returns a structure containing the pdf information for "book.pdf": pdfinfo = pdfService.getPDFInfo(source="book.pdf",name="var")}}}PDF manipulation is done using the {{cfpdf tag. This is why, you must specify the name attribute. Accessing "var" directly does not work since "var" does not exist in the page variables scope.
Syntax	serviceName.methodName(attribute-value pair)
Arguments	All attributes supported by the cfpdf tag for a given action are supported.

- setAttributes

Description	Sets attributes for the pdf function.
Returns	Nothing
Syntax	pdfService.setAttributes (attribute-value pair)
Arguments	All attributes supported by the cfpdf tag.

- getAttributes

Description	Gets the attributes that were set for the pdf function.
Returns	Returns a struct with all or some of the attribute values.
Syntax	pdfService.get_Attributes_ (attributelist)
Arguments	A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

- clearAttributes

Description	Removes all attributes added for the pdf function.
Returns	Nothing
Syntax	pdfService.clearAttributes(attribute_list)
Arguments	A comma-separated list of attributes that must be removed.

- clearParams

Description	Removes params that were added using addParam method.
Returns	Nothing
Syntax	pdfService.clearParams()
Arguments	None

- clear

Description	Removes all attributes and params added using the addParam method.
Returns	Nothing
Syntax	pdfService.clear()
Arguments	None

See also

cfpdf , Function summary

History

ColdFusion 9: Added this function.

Usage

This function corresponds to the cfpdf tag. For usage details, refer to the Usage section for cfpdf .

Example

```
<h3>PDF Thumbnail</h3> <cfscript> // Create a variable for the name of the PDF document. mypdf = "book"; thumbnailsDirectory = ExpandPath(".") & "\" & "#mypdf#_thumbnails"; //create new PDF service pdfService = new pdf(); //set attributes using implicit setters pdfService.setSource(expandpath('./#mypdf#.pdf')); //Use the getPdfInfo action to retrieve the total page count for the PDF document. PDFInfo = pdfService.getPdfInfo(name="pdfinfo"); pageCount = PDFInfo.TotalPages; WriteOutput("pageCount=" & pageCount); //Generate a thumbnail image for each page in the PDF source document, //create a directory (if it does not 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. pdfService.thumbnail(destination=thumbnailsDirectory, scale=60, overwrite=true); //Loop through the images in the thumbnail directory and generate a link //from each image to the corresponding page in the PDF document. for(i="1";i lte pageCount;i++) { //Click the thumbnail image to navigate to the page in the PDF document. WriteOutput("<a href='#mypdf#.pdf##page=#i#' target='_blank'><img src='#mypdf#_thumbnails/#mypdf#_page_#i#.jpg'></a>"); } </cfscript
```

query

Description

Used to execute a query passing SQL statements to a data source using CFScript.

Syntax

Mode	Syntax
Creating the service	new query() or createObject("component", "query")
Initializing the attributes	Any one of the following: • queryService=new query(attribute-value_pair) • queryService.setAttributes(attribute-value_pair) • queryService.set_AttributeName_(attribute_value) • queryService.execute(attribute-value_pair)
Executing the service action	queryService.execute(attribute-value_pair)

Properties

name	blockfactor	cachedafter	cachedwithin
dataSource	dbtype	debug	maxRows
password	result	timeout	username
sql			

All attributes supported by the tag cfquery can be used as attribute-value pairs. For example,

```
<cfquery Name="myName"> </cfquery>
```

can be used as

```
queryService.setName("myName");
```

See also

cfquery , Function summary

History

ColdFusion 9: Added this function.

Methods

- addParam

Description	Used in CFScript to add cfqueryparam tags to:

- Verify the data type of a query parameter
- For DBMSs that support bind variables, to enable ColdFusion to use bind variables in the SQL statement

Syntax	serviceName.addParam(attribute-value pair)
Returns	Nothing
Arguments	All attributes supported by cfqueryparam tag can be used as attribute-value pairs.

- execute

Description	Used to execute SQL statements.
Returns	A component with the following properties set:

- Result: For SQL queries that return a result set, for example, a "SELECT" SQL query.
- Prefix: Equivalent to the result attribute for the cfquery tag.

Syntax	queryService.execute(attribute-value pair)
Arguments	All attributes supported by the cfquery tag.

- setAttributes

Description	Sets attributes for the query function.
Returns	Nothing
Syntax	queryService.setAttributes (attribute-value pair)
Arguments	All attributes supported by the cfquery tag.

- getAttributes

Description	Gets attributes that were set for the query function.

Returns	Returns a struct with all or some of the attribute values.
Syntax	queryService.get_Attributes_ (attributelist)
Arguments	A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

- clearAttributes

Description	Removes all attributes added for the query function.
Returns	Nothing
Syntax	queryService.clearAttributes(attribute_list)
Arguments	A comma-separated list of attributes.

- clearParams

Description	Removes queryparams that were added using the addParam method.
Returns	Nothing
Syntax	queryService.clearParams()
Arguments	None

- clear

Description	Removes all attributes and queryparms that were added using the addParam method.
Returns	Nothing
Syntax	queryService.clear()
Arguments	None

Usage

This function corresponds to the cfquery tag. For usage information, see Usage details for cfquery .

Example

```
<cfscript> /* This example shows how to create a query service in cfscript, set/get attributes using implict setters/getters, and also how to execute the query and access the resultset */ param MaxRows="10"; param StartRow="1"; /* Query database for information if cached database information has not been updated in the last six hours; otherwise, use cached data. */ /* create a quey service */ queryService = new query(); /* set properties using implict setters */ queryService.setDatasource("cfdocexamples"); queryService.setName("GetParks"); queryService.setcachedwithin(CreateTimeSpan(0, 6, 0, 0)); /* Add sql queryparams using named and positional notation */ queryService.addParam(name="state",value="MD",cfsqltype="cf_sql_varchar"); queryService.addParam(value="National Capital Region",cfsqltype="cf_sql_varchar"); /* invoke execute() on the query object to execute the query and return a component with properties result and prefix (which can be accessed as implcit getters) */ result = queryService.execute(sql="SELECT PARKNAME, REGION, STATE FROM Parks WHERE STATE = :state and REGION = ? ORDER BY ParkName, State "); GetParks = result.getResult(); /* getPrefix() returns information like recordcount,sql etc (typically whatever one gets if one uses the result attribute of the cfquery tag */ metaInfo = result.getPrefix(); </cfscript> <cfoutput> <h4>Found #metaInfo.recordcount# records for '#metainfo.sqlparameters[2]#' in the state '#metainfo.sqlparameters[1]#' </h4> </cfoutput> <!--- 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>
```

Script functions implemented as CFCs in ColdFusion 9 Update 1

Function summary

The following table lists the script functions and the equivalent ColdFusion tag.

Function	Equivalent ColdFusion Tag
dbinfo	cfdbinfo
imap	cfimap
pop	cfpop
ldap	cfldap
feed	cffeed

dbinfo

Description

Used in CFScript to retrieve information about a data source such as database details, tables, queries, procedures, foreign keys, indexes, and version information about the database, driver, and JDBC.

Syntax

Mode	Syntax
Creating the service	new dbinfo() or createObject("component", "dbinfo");
Executing the service action	dbinfoService.action_method(attribute-value_pair);

Initializing the attributes	See Initializing the attributes below.
Getting the CFC properties	See Getting the CFC Properties below.
Working with the data returned	data=dbinfoService.action_method(attribute-value_pair);writedump(data);

Properties

datasource	dbname	name	password
pattern	table	username	

All attributes supported by the tag cfdbinfo can be used as attribute-value pairs. For example,

```
<cfdbinfo userName="myUserName">
```

can be used as

```
dbinfoService.setUserName("myUserName");
```

For details, see the Attributes section for the cfdbinfo tag.

See also

Function summary

History

ColdFusion 9.0.1: Added this function.

Methods

The following dbinfo types are available as methods. All methods have similar arguments and syntax.

•

dbnames	tables	columns	version
procedures	foriegnkeys	index	

Description	All methods correspond to the type of information supported by the tag cfdbinfo. For details of each method, see the relevant section for the tag cfdbinfo in ColdFusion 9 CFML Reference.
Returns	All methods return a query object.
Syntax	dbinfoService.methodName(attribute-value pair);
Arguments	All attributes supported by the tag cfdbinfo.

- setAttributes, getAttributes, clearAttributes, clear, setProperties, getProperties, clearProperties, and clearProperties. For details, see Methods common to all functions.

Usage

This function corresponds to the tag cfdbinfo. For details, see the Usage section for the tag cfdbinfo.

Example

```
<cfscript> d = new dbinfo(datasource=" cfartgallery ").dbnames(datasource="ajax"); writedump(d); d = new dbinfo(datasource="
ajax").dbnames(); writedump(d); </cfscript>
```

imap

Description

Used in CFScript to query an IMAP server to retrieve and manage mails within multiple folders.

Syntax

Mode	Syntax
Creating the service	new imap(); or createObject("component", "imap");
Initializing the attributes	See Initializing the attributes below.
Executing the service action	imapService.methodName(_attribute-value_pair_)
Getting the CFC properties	See Getting the CFC Properties below.
Working with returned data	imapResult=imapService.action_method(_attribute-value_pair_);

Properties

attachmentpath	connection	folder	generateuniquefilenames
maxrows	messagenumber	name	newfolder
password	port	recurse	secure
server	startrow	stoponerror	timeout
uid	username		

All attributes supported by the tag cfimap can be used as attribute-value pairs. For example,

```
<cfimap action="open" connection = "myconnection">
```

can be used as

```
imapService = new imap(server="myimapserver",username="myusername",password="mypassword",port= "myport",secure="yes");
imapService.open();
```

Note: If connection properties such as server, username, password, port, and secure are specified either during initialization or when open method is called, a connection is created implicitly. Therefore, you need not specify the properties for further actions. If sandbox security is turned on, the directory referred to by the property attachmentPath must be given the required permission. By default, the temp directory is used.

For details of the attributes, see the Attributes section for the tag cfimap.

See also

Function summary

History

ColdFusion 9.0.1: Added this function.

Methods

The following imap actions are available as methods. All methods have similar arguments and syntax.

•

getAll	delete	open	close
markRead	createFolder	deleteFolder	renameFolder
listAllFolders	moveMail	getHeaderOnly	

Description	All methods correspond to the type of information supported by the tag cfimap. For details of each method, see the relevant section of cfimap in the ColdFusion 9 CFML Reference.
Returns	A query object for methods getAll, getHeaderOnly, and listAllFolders. Else, nothing.
Syntax	imapService.methodName(attribute-value pair);
Arguments	All attributes supported by the tag cfimap.

- setAttributes, getAttributes, clearAttributes, clear, setProperties, getProperties, and clearProperties. For details, see Methods common to all functions.

Usage

This function corresponds to the tag cfimap. See the Usage section for cfimap in the ColdFusion 9 CFML Reference for details.

Example

```
<cfscript> m = new imap(); m.setAttributes(server="#REQUEST.server#",username="#REQUEST.username#",
password="#REQUEST.password#",secure="#REQUEST.secure#",
connection="#REQUEST.connectionname#",stoponerror="#REQUEST.stoponerror#"); m.open(); master = m.getAll(connection =
"#REQUEST.connectionname#",name = "queryname", stoponerror = "#REQUEST.stoponerror#" ); writedump(master); </cfscript>
```

pop

Description

Used in CFScript to retrieve or delete e-mail messages from a POP mail server.

Syntax

Mode	Syntax
Creating the service	new pop(); or createObject("component", "pop");
Initializing the attributes	See Initializing the attributes below.

Executing the service action	popService.action_method(_attribute-value_pair_);
Getting the CFC properties	See Getting the CFC Properties below.
Working with data returned	popresult = popService.action_method (attribute-value pair); where popresult is a query object if the action_method is getAll or getHeaderOnly. For any other method, nothing is returned.

Properties

server attachmentPath	debug	
generateUniqueFilenames name	maxRows	messageNumber
password timeout	port	startRow
uid	username	

All attributes supported by the tag cfpop can be used as attribute-value pairs. For example,

```
<cfpop server = "#form.popserver# " action = "getHeaderOnly" name = "GetHeaders">
```

can be used as

```
popHeaders = popService.getHeaderOnly(server="#form.popserver#");
```

Note: name is a required attribute in cfpop, but not in CFScript.

See also

Function summary

History

ColdFusion 9.0.1: Added this function.

Methods

The following pop actions are available as methods. All methods have similar arguments and syntax.

-

getHeaderOnly	getAll	delete

-

Description	All methods correspond to the type of information supported by the tag cfpop. For details of each method, see the relevant section of cfpop in the ColdFusion 9 CFML Reference.
Returns	All methods except delete returns a query object.
Syntax	popService.methodName(attribute-value pair)
Arguments	All attributes supported by the tag cfpop.

- setAttributes, getAttributes, clearAttributes, clear, setProperties, getProperties, and clearProperties. For details, see Methods common to all functions.

Usage

This function corresponds to the tag cfpop. For usage details, see the Usage section for cfpop__.

Example

```
<cfscript> p = createObject("component",""pop"); p.setAttributes(server="#popServer#",username="failoveruser",password="
#popPassword#"); r = p.GetAll(name="results",maxRows = "2"); writeoutput("getAll Passed<br>"); r = p.GetAll(messageNumber = "2");
writeoutput(#r.FROM# & "<br>"); r= p.GETHEADERONLY(messageNumber = "1"); writeoutput(#r.subject# & "<br>"); </cfscript>
```

ldap

Description

Used in CFScript to provide an interface to a Lightweight Directory Access Protocol (LDAP) directory server, such as the Netscape Directory Server.

Syntax

Mode	Syntax
Creating the service	new ldap(); or createObject("component", "ldap");
Initializing the attributes	See Initializing the attributes below.
Executing the service action	ldapService.action_method(attribute-value pair);
Getting the CFC properties	See Getting the CFC Properties below.
Working with data	ldapresult = ldapService.query(attribute-value pair).For other methods, nothing is returned.

Properties

server attributes	delimiter	
dn modifyType	filter	maxRows
name rebind	password	port

referral secure	returnAsBinary	scope
separator start	sort	sortcontrol
startRow	timeout	userName

All attributes supported by the tag cfldap can be used as attribute-value pairs. For example,

```
<cfldap action="add" server="ldap.uconn.edu">
```

can be used as

```
ldapService.add(server="ldap.uconn.edu");
```

For details, see the Attributes section for the tag cfldap.

Methods

The following ldap actions are available as methods. All methods have similar arguments and syntax.

•

query	add	modify	modifyDn
delete			

•

Description	All methods correspond to the actions supported by the tag cfldap. For details of each method, see the relevant section of cfldap in the ColdFusion 9 CFML Reference.
Returns	If method is query, returns a query object. Else, none.
Syntax	ldapService.methodName(attribute-value pair)
Arguments	All attributes supported by the tag cfldap.

- setAttributes. For details, see Methods common to all functions
- getAttributes, clearAttributes, clear, setProperties, getProperties, and clearProperties. For details, see Methods common to all functions.
- setLdapAttributes

Description	Sets the property attributes.

Returns	Nothing
Syntax	ldapService.setLdapAttributes(attribute-value);
Arguments	A string that contains the value of the property attributes.

- getLdapAttributes

Description	Gets the property attributes.
Returns	A string that contains the value of the property attributes.
Syntax	myattributes = ldapService.getLdapAttributes(__);

See also

Function summary

History

ColdFusion 9.0.1: Added this function.

Usage

This function corresponds to the tag cfldap. For usage details, see the Usage section for cfldap.

Example

```
<cfscript> l = new ldap(); l.setLdapAttributes("objectclass=top, person, organizationalPerson, inetOrgPerson;cn=Joe Smith; sn=Smith; mail=spenella@allaire.com; telephonenumber=(617) 761 - 2128"); l.setUsername("uid=admin,ou=system"); l.setPassword("administrator"); l.setPort(port); l.setServer(ldapserver); l.setdn("ou=People+o=aribus.com,dc=example,dc=com"); l.add();- l.clearAttributes(); result = l.query(name="apache", attributes="dn,cn,o,ou,c,mail,telephonenumber", start="dc=example,dc=com", scope="SUBTREE", filter="(&(cn=Joe Smith)(ou=people))"; writeoutput("<b>Adding and Querying a LDAP entry : </b>" & "CN = " & result.CN & " DN = " & result.DN & "<br> "); l.clearAttributes(); l.delete( DN="ou=People+o=aribus.com,dc=example,dc=com", ); </cfscript>
```

feed

Description

Used in CFScript to read or create an RSS or Atom syndication feed. This service reads 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.

Syntax

Mode	Syntax
Creating the service	new feed() or createObject("component" "feed")
Initializing the attributes	See Initializing the attributes below.

Executing the service action	feedService.action_method(attribute-value_pair)
Getting the CFC properties	See Getting the CFC Properties below.
Working with the data returned	• feedresult = feedService.read(attribute-value_pair) where feedresult is a struct with the keys name, query, properties, and xmlvar. • feedresult = feedService.create(attribute-value_pair) where feedresult is a string that contains the xmlvar.

Properties

columnMap	enclosureDir	escapeChar	ignoreEnclosureError
name (optional in CFScript)	outputFile	overwrite	overwriteEnclosure
properties (optional in CFScript)	proxyPassword	proxyPort	proxyServer
proxyUser	query (optional in CFScript)	source	timeout
useragent	xmlvar (optional in CFScript)		

All attributes supported by the tag cffeed can be used as attribute-value pairs. For example,

```
<cffeed action="read" source="http://googleblog.blogspot.com/atom.xml" query="feedQuery" properties="feedMetadata" >
```

can be used as

```
feedservice.read(source="http://googleblog.blogspot.com/atom.xml", query="feedQuery", properties="feedMetadata");
```

See also

Function summary

History

ColdFusion 9.0.1: Added this function.

Methods

- create

Description	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.
Returns	String representing the xmlvar
Syntax	feedService.create (attribute-value pair);
Arguments	All attributes supported by the tag cffeed.

- read

Description	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.
Returns	Struct with the following keys:

- name
- query
- properties
- xmlvar

Syntax	feedService.read (attribute-value pair);
Arguments	All attributes supported by the tag cffeed.

- setAttributes, getAttributes, clearAttributes, clear, setProperties, getProperties, and clearProperties. For details, see Methods common to all functions Script functions implemented as CFCs in ColdFusion 9 Update 1.
- getFeedProperties

Description	Returns the value of the property properties.
Returns	Struct or error (if property is not set)
Syntax	feedService.getFeedProeprties()
Arguments	None

- setFeedProperties

Description	Sets the value of the property properties.
Returns	Nothing
Syntax	feedService.setFeedProperties()
Arguments	properties struct

Usage

This service corresponds to the tag cffeed. For usage, see Usage section for cffeed.

Example

```
<cfscript> f = new feed(); r = f.read(source=feedpath); writeoutput("Name : " & r.name.title & "<br>"); writeoutput("Properties : " & r.properties.version & "<br>"); writeoutput("Query : " & r.query.recordcount & "<br>"); writeoutput("XMLVar : " & r.xmlvar.length() & "<br>"); </cfscript>
```

Methods common to all functions

The following methods are common to all script functions:

- setAttributes

Description	Sets attributes for the function.
Returns	Nothing
Syntax	service_name.setAttributes (attribute-value pair);
Arguments	All attributes supported by the equivalent tag.

- getAttributes

Description	Gets the attributes set for the function.
Returns	Returns a struct with all or some attribute values.
Syntax	service_name.get_Attributes_ (attributelist);
Arguments	A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

- clearAttributes

Description	Removes all attributes added for the function.
Returns	Nothing
Syntax	_service_name.clearAttributes(_attribute_list);
Arguments	A comma-separated list of attributes.

- clear

Description	Removes all attributes added for the function.
Returns	Nothing
Syntax	service_name.clear();
Arguments	None

- clearProperties

Description	Removes all properties added for the function.
Returns	Nothing
Syntax	service_name.clearProperties(attribute_list);
Arguments	If nothing is specified, all properties are cleared.

- setProperties

Description	Sets properties for the function.
Returns	Nothing
Syntax	service_name.setproperties (attribute-value pair);
Arguments	All attributes supported by the equivalent tag.

- getProperties

Description	Gets the properties set for the function.
Returns	Returns a struct with all or some attribute values.
Syntax	service_name.getproperties (attributelist);
Arguments	A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

Initializing the attributes

You can initialize the attributes using one of the following ways:

- service_name=new dbinfo(attribute-value pair)
- service_name=new dbinfo().init(attribute-value pair)
- service_name.setAttributes(_attribute-value pair_)
- service_name.setA_ttributeName_(attribute_value)
- service_name.action_method(attribute-value_pair)
- service_name.setProperties (attribute_value)

Getting the CFC Properties

Get the CFC properties using one of the following ways:

- service_name.getAttributeName(attributelist)
- service_name.getProperties (attributelist)
- service_name.getAttributes(attributelist)

storedproc

Description

Used to execute a stored procedure in a server database using CFScript. It specifies database connection information and identifies the stored procedure.

Syntax

Mode	Syntax
Creating the service	new storedProc() or createObject("component", "storedproc")
Initializing the attributes	Any one of the following: • storedProcService=new storedproc(attribute-value_pair) • storedprocService.setAttributes(attribute-value_pair) • storedProcService.set_AttributeName_(attribute_value) • storedProcService.execute(attribute-value_pair)
Executing the service action	storedProcService.execute(_attribute-value_pair_)

Properties

datasource	procedure	debug	cachedafter
cachedwithin	blockfactor	password	result
returncode	username		

All attributes supported by the tag cfstoredproc are supported as attribute-value pairs. For example,

```
<cfstoredproc procedure= "sp_proc">
```

can be used as

```
spService.setProcedure("sp_proc");
```

For details of the cfstoredproc tag attributes, see the Attributes section for cfstoredproc __.

See also

cfstoredproc , Function summary

History

ColdFusion 9: Added this function.

Methods

- addParam

Description	Used to add cfprocparam tags.
Syntax	storedprocService.addParam(attribute-value pair)
Returns	Nothing
Arguments	All attributes supported by cfprocparam tag can be used as attribute-value pairs.

- addProcResult

Description	Used to add cfprocresult tags to associate a query object with a result set returned by a stored procedure.
Syntax	storedprocService.addProcResult(attribute-value pair)
Returns	Nothing
Arguments	All attributes supported by the cfprocresult tag can be used as attribute-value pairs.

- execute

Description	Used to execute a stored procedure.
Returns	A component on which the following methods can be invoked:

- getProcResultSets(): To access result sets returned by the procedure.
- getProcOutVariables(): To access OUT or INOUTvariables returned by the procedure.

Syntax	storedprocService.execute(attribute-value pair)
Arguments	All attributes supported by the cfstoredproc tag.

- setAttributes

Description	Sets attributes for the storedproc function.
Returns	Nothing
Syntax	storedProcService.setAttributes (attribute-value pair)
Arguments	All attributes supported by the cfstoredproc tag.

- getAttributes

Description	Gets attributes that were set for the storedproc function.
Returns	Returns a struct with all or some of the attribute values.
Syntax	storedProcService.get_Attributes_ (attributelist)
Arguments	A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

- clearAttributes

Description	Removes all attributes added for the storedProc function.

Returns	Nothing
Syntax	storedProcService.clearAttributes(attribute_list)
Arguments	A comma-separated list of attributes.

- clearParams

Description	Removes cfprocparam tags added using the addParam method.
Returns	Nothing
Syntax	storedProcService.clearParams()
Arguments	None

- clearProcResults

Description	Removes cfprocresult tags added using the addProcResults method.
Returns	Nothing
Syntax	storedProcService.clearProcResults()
Arguments	None

- clear

Description	Removes all attributes and params that were added using the methods addProcResults and addParam.
Returns	Nothing
Syntax	storedProcService.clear()
Arguments	None

Usage

This function corresponds to the cfstoredproc tag. For usage details, refer to the Usage section for cfstoredproc .

Example

```
<cfscript> //If submitting a new book, insert the record and display confirmation if(isDefined("form.title")) { //create a new storedproc service spService = new storedproc(); //set attributes using implicit setters spService.setDatasource("books"); spService.setProcedure("Insert_Book"); //add procparams using addParam spService.addParam(cfsqltype="cf_sql_varchar", type="in",value=form.title); spService.addParam(cfsqltype="cf_sql_numeric",type="in",value=form.price); spService.addParam(cfsqltype="cf_sql_date", type="in",value=form.publishDate); spService.addParam(cfsqltype="cf_sql_numeric",type="out",variable="bookId"); //add procresults using addProcResult spService.addProcResult(name="rs1",resultset=1); //execute the stored procedure result = spService.execute(); //getprocOutVariables() returns any OUT or INOUT varibles added using addParams() bookId = result.getprocOutVariables().bookId; //getProcResultSets() returns resultsets added using addProcresult() listOfBooks = result.getProcResultSets().rs1; WriteOutput("<h3>List of Books</h3>"); writeDump(listOfBooks); //output data WriteOutput("<h3>" & "'" & form.title & "'" & " inserted into database. The ID is " & bookId & ".</h3>"); } </cfscript> <cfform action="#CGI.SCRIPT_NAME#" method="post"> <h3>Insert a new book</h3> <table> <tr> <td>Title:</td> <td><cfinput type="text" size="20" required="yes" name="title"/></td> </tr> <tr> <td>Price:</td> <td><cfinput type="text" size="20" required="yes" name="price" validate="float" /></td> </tr> <tr> <td>Publish Date:</td> <td> <cfinput type="datefield" name="publishdate" mask="mm/dd/yyyy" size="20" ></td> </tr> </table> <input type="submit" value="Insert Book"/> </cfform>
```

Chapter 7: ColdFusion Flash Form Style Reference

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 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. 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.
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 are 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 attributes

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, 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.

ColdFusion Flash Form Style Reference

You can specify styles in ColdFusion forms tags when you display the form or form element in Flash format.

Note: The column labeled **Inh** indicates whether a style is inherited by child controls, such as the form controls in a vbox.

Chapter 8: Application.CFC Reference

Application.CFC Reference

You implement methods in Application.cfc to handle ColdFusion application events and set variables in the CFC to configure application characteristics.

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.

Note: Although Windows is case-insensitive, you must 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

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 Integrating JSP and servlets in a ColdFusion application in Interoperating with JSP pages and servlets in the Developing ColdFusion Applications.
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.
authcookie.disableupdate	False	Disable update of cfauthorization cookie using cfcookie or cfheader tag
authcookie.timeout	-1	Auth Cookie age in days.
cache.useInternalQueryCache	false	If true, ColdFusion will store cached queries in the old non-cool non-Ehcache version of the cache.
cache.querysize	Administrator value	Maximum number of queries that can be cached. To be clear, this refers to **automatic** caching via cachedWithin and cachedAfter in the cfquery/queryExecute tag/function. You can store as many queries as you would like using cachePut. Well, as many as your RAM will allow. Be sensible, people.
chartStyleDirectory		Application specific chart styles directory.
clientManagement	no	• yes: enables client variables. • no

clientStorage	Administrator value	Where Client variables are stored; can be cookie, registry, or the name of a data source.
customtagpaths	Administrator value	Contains ColdFusion custom tag paths. It is a comma delimited list with absolute path.To use this variable, select the Enable Per App Settings option in the Administrator **Server** > **Settings** page.The settings that you define here take precedence over the custom tag paths defined in the Administrator **Server Settings** > **Mappings** page for the current application.
googleMapKey		The Google Maps API key required to embed Google Maps in your web pages.
datasource		Name of the data source from which the query retrieves data.
loginStorage	cookie	Whether to store login information in the Cookie scope or the Session scope.
mappings	Administrator value	A structure that contains ColdFusion mappings. Each element in the structure consists of a key and a value. The logical path is the key and the absolute path is the value. To use this variable, select the Enable Per App Settings option in the Administrator Server Settings > Settings page.The mappings that you define here take precedence over the mappings defined in the Administrator Server Settings > Mappings page for the current application.
restSettings.cfclocation		To publish the CFCs only in a particular location, provide comma-separated list of directories where the REST CFCs are located. The directory paths can be absolute or relative.If not set, all the CFCs from the application root are published.
restSettings.skipCFCWithError		When an error occurs, continue publishing, ignoring the CFC that has caused the exception.If true, the CFC with error is ignored and the rest of the CFCs are published. By default it is false.If set to false, in case of an error, the application itself is not published. But other registered application are published.If an error occurs during application startup, the error is printed in console.Each application has separate log files for logging the issues.
sessioncookie.httponly	True	Specify whether session cookies have to be set as httponly or not. i.e. accessible only to Http requests
sessioncookie.secure	False	Specify whether session cookies have to be set as secure or not. i.e. returned on any type of connection or only secured (https) connections
sessioncookie.domain		Domain for which the cookie should be set. This should match exactly with the domain, with which application would be accessed

sessioncookie.timeout	30 years	Session Cookie age in days
sessioncookie.disableupdate	False	Disable update of cfid and cftoken cookie using cfcookie or cfheader tag
serverSideFormValidation	yes	Whether to enable validation on cfform fields when the form is submitted.
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 fromcross-site scripting attacks.
security.antisamypolicy		Specify the location of antisamy file to be used when no antisamy policy file is passed to the getSafeHTML or isSafeHTML functions. The policy file can be relative to the Application CFC path or an absolute path can be provided.
compileextforinclude		Specify the list of allowed file extensions as a comma-separated list for the cfinclude tag. Specifying a wildcard * in the list makes any file included using the cfinclude tag to be compiled. If any file included using the cfinclude tag is not found in this list, their content will be statically included. By default, files with the cfm and cfml extensions are always compiled irrespective of this setting.
strictnumbervalidation		True/False. Default is true. The IsValid function for the integer and numeric types allowed the currency symbols at the start and commas inside the number. In ColdFusion 11, the isValid function behaves in a different way. Setting strictnumbervalidation to false makes the isValid function to behave in a way just like in the previous versions (ColdFusion 10 or earlier). Note that this setting also changes the behavior of the following tags: • cfargument • cfparam • cfform

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 in Ajax programming rules and techniques in the Developing ColdFusion Applications.
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 in Ajax programming rules and techniques in the Developing ColdFusion Applications.
serialization.preservecaseforstructkey	False	Boolean that determines if case for struct keys should be preserved when serializing a struct to JSON.
serialization.serializequeryas	row	Determines how queries should be serialized to JSON. Possible values are row, column, and struct.
serialization.preserveCaseForQueryColumn	false	If true, column case will be preserved.

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 (for example, 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 must have 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. The welcomeFileList variable has to be used with most integrated web and application servers.
smtpServersettings		A struct that contains the following values: server, username, and password.If no value is specified, takes the value in the administrator.
sameformfieldsasarray	false	If the form fields have the same name, ColdFusion converts the form fields as an array instead of a list. To do this, in the Application.cfc, specify the following: this.sameformfieldsasarray = "true". Note: The empty string values will be preserved only if this is set to true.
timeout		This number represents how long an individual request can take. Timeout set using <cfsetting requesttimeout=""> overrides the timeout in the Application.cfc using this.timeout="".

debuggingIPAddresses		A list of IP addresses that need debugging.
enablerobustexception		Overrides the default administrator settings. It does not report compile-time exceptions.
javaSettings		A structure allowing you to specify Java class paths to be made available to your code. Valid keys are: loadPaths (an array of paths to include when searching for Java libraries), loadColdFusionClassPath (a boolean indicating if the default class path should be used, defaults to false), and reloadOnChange (a boolean indicating if the classpaths should be reloaded when they change, the default is false).

Form fields with same name

Assume that the form fields have same name. In this case, ColdFusion converts the form fields as an array instead of a list.To do this, in the Application.cfc, specify the following: this.sameformfieldsasarray = "true".The default value is false.

Enhancements made in ColdFusion 11

In ColdFusion 11, you can register application-specific datasources in Application.cfc. These datasources will be specific to that application and will not be available through the Administrator. If there is a name clash with a server-wide datasource, the one specific to the application will be given the priority.

```
this.datasources.dsn1={"database"="regression""host"="localhost\MSSQL2008",
"driver"="MSSQLServer""username"="sa""password"="password"};
this.datasources.dsn2={"driver"="MSSQLServer",url="jdbc:macromedia:sqlserver:
//localhost\MSSQL2008;databaseName=regression;;sendStringParametersAsUnicode=
false;querytimeout=0;MaxPooledStatements=1000""username"="sa""password"= "pass"};
```

With a custom driver:

```
this.datasources.dsn3 = { "driver" = "other", "url" = "jdbc:sqlserver://localhost\MSSQL2008;databaseName=pubs;sendStringParameters
AsUnicode=false;querytimeout=0;MaxPooledStatements=1000", "username" = "sa", "password" = "S33N0Ev!l",
"class"="com.microsoft.sqlserver.jdbc.SQLServerDriver"};
```

The following drivers are supported:

- MSSQLServer
- Oracle
- Sybase
- DB2
- Informix
- MySQL_DD
- PostgreSQL
- MSAccess
- Apache Derby Embedded
- Apache Derby Client
- MySQL5

- ODBCSocket
- Other (for custom driver)

Enhancements made in ColdFusion 9.0.1

Application.cfc lets you specify data source authentication details for the data source. The data source settings can now be a string or a struct. When string, it is considered to be the data source name and authentication information is taken from the data source defined in the ColdFusion Administrator. You can specify the authentication information using a struct value for data source. The following are the key names:

- name: data source name
- username: Username for the data source
- password: Password for the data source**Example**

```
<this.datasource={name='cfartgallery', username="user", password="passwd"}>
```

or

```
<this.datasource="cfartgallery">
```

Note: The same convention is used for ORM default data source where you can specify the data source authentication information in the ormsettings.

The following application-specific attributes have been added for Amazon S3 integration:

- accessKeyId: ID for Amazon S3 account.
- awsSecretKey: Secret key for S3 account.
- defaultLocation:}}The default location of Amazon S3 bucket creation. A bucket on S3 storage can be in one of the following regions: {{US, EU, or US-WEST.The defaultLocation provided in the Application.cfc defines the default location for the bucket that you create. The default value is US.**Example**

```
<cfscript> this.s3.accessKeyId = "key_ID"; this.s3.awsSecretKey = "secret_key"; this.s3.defaultLocation="location"; </cfscript>
```

Application-specific In-memory file system

You can use in-memory file system specific to applications. This enables application isolation for your virtual file system. That is, the file created in the in-memory file system by one application will not be accessible to another application. The settings can be specified in the Application.cfc as follows:

Variable	Description
this.inmemoryfilesystem.enabled	Set the value to true to enable in-memory file system for application. This is the default setting.
this.inmemoryfilesystem.size	Specify the memory limit in MB for the in-memory file system.You can also specify the value in the ColdFusion Administrator (Server Settings > Settings > Memory Limit per Application for In-Memory Virtual File System).The lesser value is considered.

Method summary

The following table briefly describes the application event methods that you can implement in Application.CFC:

Method name	Method runs when
onAbort	Runs when you execute the tag cfabort
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.
onCFCRequest	HTTP or AMF calls are made to an application.
onError	An exception that is not caught by a try/catch block occurs.
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
onServerStart	A ColdFusion server 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/onCFCRequest
5. onRequestEnd The onApplicationEnd, onSessionEnd, and onError CFCs are triggered by specific events.

onAbort

Description

Runs when you execute the tag cfabort.

Note: If showError attribute is specified in cfabort, onError method is executed instead of OnAbort.

Note: When using cfabort, cflocation, or cfcontent tags, the OnAbort method is invoked instead on OnRequestEnd.

Returns

Nothing

Syntax

```
<cffunction name="onAbort" returnType="boolean"> <cfargument type="string" name="targetPage" required=true/> ... <cfreturn BooleanValue /> </cffunction>
```

Parameters

Parameter	Description
targetPage	The path from the web root to the requested CFML page.

Example

Application.cfc

```
<cfcomponent> <cffunction name="onAbort" access="public" returntype="void" hint="Hanldes Aborted request"> <cfargument type="String" name="targetPage" required=true/> <cfoutput> Target Page: #targetPage#</cfoutput> <!--- do stuff ---> </cffunction> </cfcomponent>
```

Test.cfm

```
<cfabort>
```

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 in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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.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.

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

, Method summary, Managing the application with Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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 runs 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 you 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>
```

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 in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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 must 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 must use try/catch blocks in your missing template handler and, if the catch block cannot handle the error, it must 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.</h3> <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>
```

Note: *When OnMissingTemplate is configued in ColdFusion Admin and when user try to access some non-existing page, OnMissingTemplate gets invoked and Tomcat returns that content. But in case of IIS, instead of showing that content IIS displays its own error page / 404 page. To fix this issue, a new property is introduced in isapi_redirect.properties, named iis_skip_custom_errors_enable. If set to true, it will skip IIS custom errors. Default value is false.*

onCFCRequest

Description

Intercepts any HTTP or AMF calls to an application based on CFC request.

Syntax

```
<cffunction name="oncfcRequest" returnType="void"> <cfargument type="string" name="cfcname"> <cfargument type="string" name="method"> <cfargument type="struct" name="args"> </cffunction>
```

See also

Method summary , Handling errors in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

Parameter	Description
cfcname	Fully qualified dotted path to the CFC.
method	The name of the method invoked.
args	The arguments (struct) with which the method is invoked.

Usage

Whereas onRequest handles only requests made to ColdFusion templates, this function controls Ajax, Web Service, and Flash Remoting requests.

Example

Create a folder onCFCRequest in your web root. Place test.cfc and Application.cfc in this directory and make an HTTP call to the CFC using the following URL:http://localhost:8500/onCFCRequest/test.cfc?method=foo&arg1=1&arg2=2&arg3=3When you run the URL, the method onCFCRequest is called and the function name foo is passed along with the arguments arg1, arg2, and arg3. You can then invoke the test.cfc as shown in the following example:

```
<!--- Application.cfc ---> <cfcomponent> <cfset this.name = "oncfcrequest"> <cffunction name="onCFCRequest"> <cfargument type="string" name="cfcname" required=true> <cfargument type="string" name="method" required=true> <cfargument type="struct" name="args" required=true> <cflog text="oncfcRequest()"> <cfdump var="#arguments#" output="console" format="text"> <cfinvoke component = "arguments.cfcname" method = "arguments.method" returnVariable = "result" argumentCollection = "#arguments.args#"> <cfdump var="#result#" output="console" format="text"> </cffunction> <cffunction name="onRequest" output="yes" access="remote"> <cfargument type="string" name="targetpage"> <cflog text="onRequest()"> </cffunction> </cfcomponent> <!--- test.cfc ---> <cfcomponent> <cffunction name="foo"> <cfargument name="arg1" type="string" > <cfargument name="arg2" type="string" > <cfargument name="arg3" type="string" > <cfreturn arguments> </cffunction> </cfcomponent>
```

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 in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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, EventName 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>
```

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 in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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.

Note: When using cfabort, cflocation, or cfcontent tags, the OnAbort method is invoked instead on OnRequestEnd.

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 simple authuserfooter.cfm page consists of the following code:

```
<cfoutput> <h3>Thank you for shopping at our store, #theAuthUser#!</h3> </cfoutput>
```

A 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">
```

onRequest

Description

Runs when a request starts, after the onRequestStartevent 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 in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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. The affected directory and subdirectories do not contain any CFC files that are intended to be accessed as web services, AJAX bind, 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 does 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>
```

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 in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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. 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>
```

onServerStart

Note: Despite being documented in this section of the manual, this onServerStart method is NOT a method of Application.cfc. See below for more details on where, why, and how to use this method.

ColdFusion now supports a CFC with an onServerStart method that runs only when the server starts. The onServerStart method takes no parameters, and is the only function in the CFC. The function is useful for application-independent tasks, such as instantiating the applications, configuring logging, or setting up the scheduler. By default, ColdFusion looks for the onServerStart method in cf_webroot/Server.cfc. To specify a different filepath:

1. Launch ColdFusion Administrator.
2. Click ColdFusion Administrator Server Settings > Settings.
3. Specify the absolute filepath under the web root on the Settings page such as c:\Server.cfc. Alternatively, you can use a dot-delimited path under the web root, such as a.b.Server.

Note: If you use an absolute path, the filename must end with .cfc. If you use a relative path or dotted path, do not end the name with the .cfc suffix.

You select an option on the Settings page to enable and disable the onServerStart method. By default, the method is disabled. You can also specify a timeout limit (in seconds) for the onServerStart method. The timeout limit determines the duration for which the method would be allowed to run during server start up. This setting can be specified in server.cfc. The onServerStart method can use most CFML features, but not any features that require full server start. For example, the method cannot use a cfhttp tag with a URL that specifies a location on the same server. You also cannot use Application or Request scope variables in the method. By default, all errors, including any serverCFC errors, are logged in <ColdFusion_home>/WEB-INF/cfusion/logs directory for standalone and <appserver_root>/logs directory for J2EE configurations. You can also specify a different location for logging by configuring the log directory setting in ColdFusion Administrator > Debugging and Logging > Logging Settings. The server.log file contains server startup information. So, any server.CFC startup errors are logged in it, but for details about the error, you have to see the exception.log file. In addition, server startup information is logged in {appserver_root}/logs directory. For WebSphere, it is logged in the SystemOut.log file.

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 requests in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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:

- 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 requests in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

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 need not 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 9:ColdFusion Event Gateway Reference

ColdFusion Event Gateway Reference

Java interfaces are available for building ColdFusion custom CFXs in Java.

Note: The following CFML functions also apply to gateway application development: GetGatewayHelper , SendGatewayMessage .

addEvent

CFEvent

CFEventclass

Constructor

Gateway development interfaces and classes

GatewayHelper interface

Gateway interface

GatewayServices class

getCFCMethod

getCFCPath

setCFCTimeout

getCFCTimeout

getData

getGatewayID

getGatewayID_1

getGatewayServices

getGatewayType

getHelper

getLogger

getMaxQueueSize

getOriginatorID

setCFCMethod

setCFCPath

getQueueSize

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 in Building an event gateway in the Developing ColdFusion Applications

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); }
```

CFEvent

Description

CFEvent constructor.

Category

Event Gateway Development

Syntax

CFEvent(String gatewayID)

See also

getGatewayID , CFML CFEvent structure, CFEvent class in the Developing ColdFusion Applications

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(); }
```

CFEventclass

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 structurethat 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 outgoingMessagemethod. 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.

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 in Building an event gateway in the Developing ColdFusion Applications.

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(); }
```

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:

getStatus

Description

Returns the gateway status.

Category

Event Gateway Development

Syntax

```
public int getStatus()
```

See also

getStatus in the Developing ColdFusion Applications

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; }
```

setCFCPath

Description

Specifies the listener CFC that processes this event.

Category

Event Gateway Development

Syntax

```
void setCFCPath(String path)
```

See also

getCFCPath , setCFCMethod, setCFCTimeout, CFEventclassin the Developing ColdFusion Applications

Parameters

Parameter	Description
path	An absolute path to the application listener CFC that processes 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 setCFCListenersmethod, 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); }
```

setCFCMethod

Description

Sets the name of the CFC method that processes an incoming message.

Category

Event Gateway Development

Syntax

```
void setCFCMethod(String method)
```

See also

getCFCMethod , setCFCPath, setCFCTimeout, CFML CFEvent structure, CFEventclassin the Developing ColdFusion Applications

Parameters

Parameter	Description
method	The method in the listener CFC that ColdFusion calls 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 handles 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.
```

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, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

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; }
```

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 in Building an event gateway the Developing ColdFusion Applications

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); . . .
```

getBuddyList

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 in the Developing ColdFusion Applications

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) • NA (displays as NOT AVAILABLE) • FREE_TO_CHAT (displays as FREE TO CHAT) • IDLE**XMPP only** • NA (displays as NOT AVAILABLE) • FREE_TO_CHAT (displays as FREE TO CHAT) • IDLE**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.
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 Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods. For an example of using this method to get the buddy custom away message, see onBuddyStatus.----

getBuddyInfo

Description

Gets the buddy list for the gateway's user ID.

Syntax

```
array = getBuddyList()
```

See also

addBuddy , getBuddyInfo, removeBuddy, Using the GatewayHelper object in the Developing ColdFusion Applications

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 in Using the GatewayHelper object in the Developing ColdFusion Applications which uses all GatewayHelper class methods.

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 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	

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 in the Developing ColdFusion Applications

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>
```

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>
```

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 • NOT AVAILABLE • FREE TO CHAT • IDLE**XMPP only** • NOT AVAILABLE • FREE TO CHAT • IDLE**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.

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>
```

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>
```

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>
```

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

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.

CFML CFEvent structure

The CFML listener CFC methods receive messages in the form of a CFEvent structure that corresponds to the CFEvent class in Event gateway elements in the Developing ColdFusion Applications 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.

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 in Building an event gateway in the Developing ColdFusion Applications

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); }
```

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 in Building an event gateway in the Developing ColdFusion Applications

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 do 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);
```

setOriginatorID

Description

Identifies the originator of an incoming message.

Category

Event Gateway Development

Syntax

```
void setOriginatorID(String originatorID)
```

See also

getOriginatorID , CFML CFEvent structure, CFEventclassin Developing ColdFusion Applications

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."); } }
```

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 in the Developing ColdFusion Applications.

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: " &amp; message))>
<cfreturn retValue>
</cffunction>
```

submit Multi 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.
shortMessageormessagePayload	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 in the Developing ColdFusion Applications.

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: " &amp; message>
<cfreturn retValue>
</cffunction>
```

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.
shortMessageormessagePayload	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 in the Developing ColdFusion Applications.

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: " &amp; CFEvent.Data.MESSAGE>
<!--- Send the message back. --->
<cfreturn retValue>
```

setGatewayType

Description

Identifies the type of event gateway.

Category

Event Gateway Development

Syntax

void setGatewayType(String gatewayType)

See also

getGatewayType , CFML CFEvent structure, CFEventclassin Developing ColdFusion Applications

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."); } }
```

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, setGatewayIDin the Developing ColdFusion Applications

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; }
```

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, CFEventclassin the Developing ColdFusion Applications

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."); } }
```

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 in the Developing ColdFusion Applications

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 instances 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; } }
```

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 in the Developing ColdFusion Applications

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; }
```

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 in the Developing ColdFusion Applications

Returns

The date and time that the gateway changed its online status, normally by calling the setStatus gatewayHelper object method.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

Returns

The number of messages received by the gateway since it was started.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 GatewayHelper object in the Developing ColdFusion Applications

Returns

The number of messages sent by the gateway since it was started.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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 in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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: If the XMPP 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 in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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: If the XMPP 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 in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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 in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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: If the XMPP server does not support permission management, this function returns False to all values except PERMIT_ALL.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

Parameters

Parameter	Description
status	The gateway's online presence status; one of the following: • ONLINE • AWAY • DND (Do Not Disturb) • NA (Not Available) • FREE_TO_CHAT • IDLE **XMPP only** • NA (Not Available) • FREE_TO_CHAT • IDLE **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 in Using the GatewayHelper object in the Developing ColdFusion Applications, 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

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 NotificationFor 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 onlyPlus0: 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 requestedPlus0: No Intermediate notification requested _or_16: Intermediate notification requested
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 Alphabet00000001--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 Encoding00001010--ISO-2022-JP (Music Codes)00001101--Extended Kanji JIS(X 0212-1990)00001110--KS C 560111xxxxxx--GSM control use only; see the GSM 03.38 specificationFor 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 in the Developing ColdFusion Applications.

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 in the Developing ColdFusion Applications

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**
- IDLE

Usage

The DO NOT DISTURB, NOT AVAILABLE, and FREE TO CHAT strings differ from the status values that you use in the setStatusmethod, which does not allow spaces in the status names.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

Returns

The gateway's protocol, as determined by the gateway type; one of the following values:

- JABBER (for XMPP)
- SAMETIME

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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: If the XMPP server does not support permission management, this function always returns PERMIT_ALL.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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: If the XMPP server does not support permission management, this function always returns False.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

Returns

The gateway's nickname, if any; empty string, otherwise.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 GatewayHelper object in the Developing ColdFusion Applications

Returns

The gateway's user name, as specified in gateway configuration file.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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.

If the XMPP server does not support permission management, this function always returns False.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

Returns

The gateway's custom away message if it has been set by the GatewayHelper object setStatus method.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications , which uses all GatewayHelper class methods.

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 gatewayService.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()))); }
```

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 does 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 gatewayService.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())); }
```

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 in Building an event gateway in the Developing ColdFusion Applications.

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(); }
```

getGatewayType

Description

Returns the gateway type field of the CFEvent object.

Category

Event Gateway Development

Syntax

```
String getGatewayType()
```

See also

setGatewayType , CFML CFEvent structure, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

Returns

The gateway type of the CFEvent object, or null.

Usage

Most gateways do not need to use this method.----

getGatewayServices

Description

Static method that returns the GatewayServices object. Gateway code can call this method at any time, if necessary.

Category

Event Gateway Development

Syntax

```
GatewayServices getGatewayServices()
```

See also

GatewayServices class in the Developing ColdFusion Applications

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);
```

getGatewayID_1

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, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

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.----

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 in Building an event gateway in the Developing ColdFusion Applications.

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; }
```

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, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

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; }
```

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, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

Returns

The listener CFC time-out, in seconds, as set by the setCFCTimeoutmethod, 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.

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 in Event gateway elements in the Developing ColdFusion Applications

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); }
```

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, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

Returns

An absolute path to the application listener CFC that processes the event, as set by the setCFCPathmethod. 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.

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, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

Returns

For incoming messages, the name of the method that gateway services call in the listener CFC, as set by the setCFCMethodmethod. 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.

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(CFEventmsg)	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.
intgetMaxQueueSize ()	Returns the current size of the ColdFusion event queue that handles all messages for all gateways.

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:

Signaturestart	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.

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 getHelpermethod that returns the helper class, or null if you do not implement the interface. For information on GatewayHelper classes, see GatewayHelper class.

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 in the Developing ColdFusion Applications

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.

If the XMPP server does not support permission management, this function always returns False.

Example

See GatewayHelper example, in Using the GatewayHelper object in the Developing ColdFusion Applications, 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 in the Developing ColdFusion Applications

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: If the XMPP server does not support permission management, this function always returns False.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

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 in the Developing ColdFusion Applications

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 onAddBuddyResponsemethod 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 in Using the GatewayHelper object the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

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 in Building an event gateway in the Developing ColdFusion Applications

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"; }
```

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 in Building an event gateway in the Developing ColdFusion Applications

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());
```

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.

stop

Description

Stops a gateway if it is running.

Category

Event Gateway Development

Syntax

```
public void stop()
```

See also

restart , start, Starting, stopping, and restarting the event gateway in Building an event gateway in the Developing ColdFusion Applications

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; }
```

start

Description

starts a gateway running.

Category

Event Gateway Development

Syntax

```
public void start()
```

See also

restart , stop, Starting, stopping, and restarting the event gateway in Building an event gateway in the Developing ColdFusion Applications

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; }
```

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 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>
```

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(); }
```

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 in Building an event gateway in the Developing ColdFusion Applications

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 writes message with a severity of "fatal" to the log file specified in the getLogger method that returned the Logger instance (or the default log file).

SMS gateway message sending commands

ColdFusion applications that use gateways of the Short Message Service (SMS) type can send the commands (described in this section) to the event gateway in an outgoing message.

Chapter 10: ColdFusion C++ CFX Reference

C++ class overview

The following table lists the CFXAPI classes and methods:

Class	Methods
CCFXException class	:GetError :GetDiagnostics
CCFXQuery class	:AddRow :GetColumns :GetData :GetName :GetRowCount :SetData
CCFXRequest class	:AddQuery :AttributeExists :CreateStringSet :Debug :GetAttribute :GetAttributeList :GetCustomData :GetQuery :ReThrowException :SetCustomData :SetVariable :ThrowException :Write :WriteDebug
CCFXStringSet class	:AddString :GetCount :GetIndexForString :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::SetQueryStringCCFXQuery::SetTotalTime	ColdFusion MXColdFusion 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. The CCFXRequest class, CCFXQuery class, and CCFXStringSet classcan throw exceptions of this type. Your ColdFusion Extension code must be written to handle exceptions of this type.

Class methods

virtual LPCSTR GetError()	The :GetError function returns a general error message.
virtual LPCSTR GetDiagnostics()	The :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()	:AddRow adds a row to a query.
virtual CCFXStringSet* GetColumns	:GetColumns retrieves a list of a query's column names.
virtual LPCSTR GetData(int iRow, int iColumn)	:GetData retrieves a data element from a row and column of a query.
virtual LPCSTR GetName()	:GetName retrieves the name of a query.
virtual int GetRowCount()	:GetRowCount retrieves the number of rows in a query.
virtual void SetData(int iRow, int iColumn, LPCSTR lpszData)	: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 classthat 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 :GetRowCount. You can determine the number of columns in a query by retrieving the list of columns using :GetColumns, and then calling :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 :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)	:AttributeExists checks whether the attribute was passed to the tag.
virtual LPCSTR GetAttribute(LPCSTR lpszName)	:GetAttribute gets the value of the passed attribute.
virtual CCFXStringSet* GetAttributeList()	:GetAttributeList gets an array of attribute names passed to the tag.
virtual CCFXQuery* GetQuery()	: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)	:Write writes text output back to the user.
virtual void SetVariable(LPCSTR lpszName, LPCSTR lpszValue)	:SetVariable sets a variable in the template that contains this tag.
virtual CCFXQuery* AddQuery(LPCSTR lpszName, CCFXStringSet* pColumns)	:AddQuery adds a query to the template that contains this tag.
virtual BOOL Debug()	:Debug checks whether the tag contains the Debug attribute.
virtual void WriteDebug(LPCSTR lpszOutput)	:WriteDebug writes text output into the debug stream.
virtual CCFXStringSet* CreateStringSet()	:CreateStringSet allocates and returns a CCFXStringSet instance.
virtual void ThrowException(LPCSTR lpszError, LPCSTR lpszDiagnostics)	:ThrowException throws an exception and ends processing of this request.
virtual void ReThrowException(CCFXException* e)	:ReThrowException rethrows an exception that has been caught.
virtual void SetCustomData(LPVOID lpvData)	:SetCustomData sets custom (tag specific) data to carry with a request.
virtual LPVOID GetCustomData()	: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 :AddRow and :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 :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 :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 :GetAttribute.

Returns

Returns an object of class CCFXStringSet classthat 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 :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. Set the attribute to the name of a query (created using the cfquery tag or another custom tag). The QUERY attribute is optional and must be used only by tags that process an existing data set.

Returns

Returns an object of the CCFXQuery classthat 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

Rethrows 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 :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 :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 :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. You can add strings to a set and retrieve them by a numeric index (index values for strings are 1-based). To create a string set, use :CreateStringSet.

Class methods

virtual int AddString(LPCSTR lpszString)	:AddString adds a string to the end of a list.
virtual int GetCount()	:GetCount gets the number of strings contained in a list.
virtual LPCSTR GetString(int iIndex)	:GetString gets the string located at the passed index.
virtual int GetIndexForString(LPCSTR lpszString)	: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 :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 :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 :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>" ) ; }
```

ColdFusion C++ CFX Reference

ColdFusion includes Java interfaces for building ColdFusion custom CFXs in Java.

C++ class overview

Deprecated class methods

CCFXException class

CCFXQuery class

CCFXRequest class

CCFXStringSet class

Chapter 11: ColdFusion Java CFX Reference

ColdFusion Java CFX Reference

ColdFusion includes Java interfaces for building ColdFusion custom CFXs in Java. Class libraries overview
Custom tag interface
Query interface
Request interface
Response interface
Debugging classes reference

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()(in CCFXQuery class)	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

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

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

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 12: WDDX JavaScript Objects

WDDX JavaScript Objects

You use JavaScript objects and functions to use with WDDX in a ColdFusion application. JavaScript object overview

WddxRecordset object

WddxSerializer object

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>
```

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 in the Developing ColdFusion Applications.

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 parameterescapeStringsdetermines 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 theaddColumnfunction:

```
// 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 theaddRowsfunction:

```
// 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 thegetFieldfunction (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 thegetRowCountfunction:

```
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 thesetFieldfunction:

```
// 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 WddxSerializerserializeValuefunction:

```
...
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);
}
...
```

WddxSerializer object

The WddxSerializer object includes functions that serialize any JavaScript data structure. For more information on using this object, see Using WDDX in the Developing ColdFusion Applications.

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 serialize(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 objectThis 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 + "'/>");
}
...
```

Chapter 13: ColdFusion ActionScript Functions

ColdFusion ActionScript Functions

BOTH ActionScript FUNCTIONS HAVE BEEN DEPRECATED

For a full list of deprecated features, refer to Deprecated features .

ColdFusion includes two server-side ActionScript functions,CF.queryandCF.http., including specific syntax and methods.

CF.http

CF.query

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 theparamsargument.
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.

resolveurl	Optional	ForGetandPostmethods. • 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.httpparams are passed as an array of objects. Theparamsargument is required for POST operations.
path	Optional	The path to the directory in which to store files. When using thepathargument, thefileargument is required.
file	Optional	Name of the file that is accessed. For GET operations, defaults to the name specified in theurlargument. Enter path information in thepathargument. This argument is required if you are using thepathargument.

Usage

You can write theCF.httpfunction 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 theCF.httpfunction using positional arguments:

```
CF.http(url); CF.http(method, url); CF.http(method, url, username, password); CF.http(method, url, params, username, password);
```

Do not use curly braces {} with positional arguments.

The following parameters can only be passed as an array of objects in theparamsargument in theCF.httpfunction:

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=adobe.com Connection: close Content-Type: text/html
Filecontent	File contents, for text and MIME files.
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 thegetfunction:

```
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 in the Developing ColdFusion Applications.

Example

The following examples show a number of the ways to use theCF.httpfunction:

```
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"); }
```

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 theCF.queryfunction 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 theCF.queryfunction 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);
```

Do not use curly braces {} with positional arguments.

You can manipulate the record set returned by theCF.queryfunction 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 in the Developing ColdFusion Applications. 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; }
```

Chapter 14: ColdFusion Mobile Functions

ColdFusion Mobile Functions

Accelerometer Functions

The Accelerometer API allows you to capture the device motion in the X, Y, and Z direction. The ColdFusion Accelerometer API dispatches certain events based on the activity detected by the device's motion sensor (accelerometer). The data returned by the API represents the device's location or movement along a 3-dimensional axis. When the device moves, the sensor detects this movement and returns acceleration data.

cfclient.accelerometer.clearWatch

cfclient.accelerometer.getOptions

cfclient.accelerometer.setOptions

cfclient.accelerometer.watch

Camera Functions

ColdFusion allows you to access the camera of the mobile device through simple CFML code.

The following sections describe how you can invoke the camera using <cfclient>.

Note: The image quality of pictures taken using the camera on newer devices is quite good, and images from the Photo Album will not be downscaled to a lower quality, even if a quality parameter is specified. Encoding such images using Base64 can cause memory issues on many newer devices. Therefore, using FILE_URI for images captured is highly recommended.

cfclient.camera.cleanup

cfclient.camera.getOptions

cfclient.camera.getPicture

cfclient.camera.getPictureFromAlbum

cfclient.camera.getPictureFromPhotoLibrary

cfclient.camera.setOptions

options Object

Connection Functions

The connection API allows you to detect the type of phone connection. Using these APIs, you can find out the connection type (2G, 3G, or 4G) and register for events when the phone goes online or offline.

cfclient.connection.getType

cfclient.connection.onOffline

cfclient.connection.onOnline

Contact Functions

ColdFusion allows you to access the contacts of the mobile device through simple CFML code.

The following sections describe how you can work with phone contacts using <cfclient>.

Contact Creation Functions

Contacts Searching Functions

Event Functions

You can let your applications listen to the device events by adding specific event listeners. The following sections show you how to handle device events.

cfclient.events.onBackButton

cfclient.events.onBatteryCritical

cfclient.events.onBatteryLow

cfclient.events.onBatteryStatusChange

cfclient.events.onMenuButton

cfclient.events.onPause

cfclient.events.onResume

cfclient.events.onSearchButton

File System Functions

ColdFusion allows you to access the file system of the mobile device through simple CFML code.

The following sections describe how you can manage the native file system using <cfclient>.

Note: All the file system functions support file URL (file path starting with file://) apart from supporting absolute and relative paths.

cfclient.file.append

cfclient.file.copy

cfclient.file.copyDirectory

cfclient.file.createDirectory

cfclient.file.directoryExists

cfclient.file.download

cfclient.file.exists

cfclient.file.get

cfclient.file.getDirectory

cfclient.file.getWorkingDirectory

cfclient.file.listDirectory

cfclient.file.move

cfclient.file.moveDirectory

cfclient.file.read

cfclient.file.readAsBase64

cfclient.file.remove

cfclient.file.removeDirectory

cfclient.file.renameDirectory

cfclient.file.setFileSystem

cfclient.file.setWorkingDirectory

cfclient.file.upload

cfclient.file.write

DirectoryEntry Object

FileEntry Object

rename Function

Geolocation Functions

The Geolocation APIs allow your mobile application to connect and get details from the location sensor.

cfclient.geolocation.clearWatch

cfclient.geolocation.getCurrentPosition

cfclient.geolocation.getOptions

cfclient.geolocation.setOptions

cfclient.geolocation.watchPosition

Media and Capture Functions

ColdFusion enables you to build mobile applications capable of capturing audio and video. APIs are made available for playback and control of audio files.

Audio Functions

Video Functions

Notification Functions

Visual, audible and tactile device notification support

cfclient.notification.alert

cfclient.notification.beep

cfclient.notification.confirm

cfclient.notification.vibrate

Splash Screen Functions

You can show or hide a splash screen image that you have configured as part of your PhoneGap build using the show() and hide() functions.

cfclient.splashscreen.hide

cfclient.splashscreen.show

Storage Functions

ColdFusion supports PhoneGap storage APIs that are based on the Web storage specification. Web storage is an important aspect of any client application running on a browser or a device. This section describes how you can access and manage web storage through ColdFusion.

Support for local storage is provided through key-value pairs.

cfclient.localstorage.clear

cfclient.localstorage.getItem

cfclient.localstorage.removeItem

cfclient.localstorage.setItem