macromedia®
COLDFUSION®
MX

CFML Quick Reference

Part Number ZCF60M300

Acknowledgments

Project Management: Stephen M. Gilson

Writing: Christina Lamkin

Editing: Linda Adler

First Edition: May 2002

Macromedia® Incorporated

CONTENTS

CFML tags

cfabort

```
<cfabort
  showError = "error_message">
```

cfapplet

```
<cfapplet
  appletSource = "applet_name"
  name = "form_variable_name"
  height = "height_in_pixels"
  width = "width_in_pixels"
  vSpace = "space_above_and_below_in_pixels"
  hSpace = "space_on_each_side_in_pixels"
  align = "alignment_option"
  notSupported = "message_to_display_for_nonJava_browser"
  param_1 = "applet_parameter_name"
  param_2 = "applet_parameter_name"
  param_n = "applet_parameter_name">
```

cfapplication

```
<cfapplication
  name = "application_name"
  clientManagement = "Yes" or "No"
  clientStorage = "datasource_name" or "Registry" or
    "Cookie"
  setClientCookies = "Yes" or "No"
  sessionManagement = "Yes" or "No"
  sessionTimeout = #CreateTimeSpan(days, hours,
    minutes, seconds)#
  applicationTimeout=#CreateTimeSpan(days,hours,
    minutes, seconds)#
  setDomainCookies = "Yes" or "No">
```

cfargument

```
<cfargument
  name=”...”
  type=”...”
  required=”...”
  default=”...”
  ...>
```

cfassociate

```
<cfassociate
  baseTag = "base_tag_name"
  dataCollection = "collection_name">
```

cfbreak

```
<cfbreak>
```

cfcache

```
<cfcache
  action = "action"
  directory = "directory_name"
  timespan = "value"
  expireURL = "wildcarded_URL_reference"
  username = "username"
  password = "password"
  port = "port_number"
  protocol = "protocol">
```

cfcase See cfswitch

cfcatch See cftry

cfchart

```
<cfchart
  format = “flash, jpg, png”
  chartHeight = "integer number of pixels"
  chartWidth = "integer number of pixels"
  scaleFrom = "integer minimum value"
  scaleTo = "integer maximum value"
  showXGridlines = “yes" or "no”
  showYGridlines = “yes" or "no”
  gridlines = "integer number of lines"
  seriesPlacement = "default, cluster, stacked, percent"
  foregroundColor = "Hex value or Web color"
  dataBackgroundColor = "Hex value or Web color"
  borderBackgroundColor = “Hex value or Web color”
  showBorder = "yes" or "no”
  font = “font name”
  fontSize = “integer font size”
  fontBold = “yes" or "no”
  fontItalic = “yes" or "no”
  labelFormat = “number, currency, percent, date”
  xAxisTitle = "title text"
  yAxisTitle = "title text"
  sortXAxis = "yes/no"
  show3D = “yes" or "no”
  xOffset = "number between -1 and 1"
  yOffset = "number between -1 and 1"
  rotated = "yes/no"
  showLegend = "yes/no"
  tipStyle = "MouseDown, MouseOver, Off"
  tipBGColor = "hex value or web color"
  showMarkers = ”yes" or "no”
  markerSize = "integer number of pixels"
  pieSliceStyle = "solid, sliced"
  url = “onClick destination page"
  name = "String"
  </cfchart>
```

cfchartdata

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

cfchartseries

```
<cfchartseries
  type="type"
  query="queryName"
  itemColumn="queryColumn"
  valueColumn="queryColumn"
  seriesLabel="Label Text"
  seriesColor="Hex value or Web color"
  paintStyle="plain, raise, shade, light"
  markerStyle="style"
  colorlist = “list”>
</cfchartseries>
```

cfcol

```
<cfcol
  header = "column_header_text"
  width = "number_indicating_width_of_column"
  align = "Left" or "Right" or "Center"
  text = "column_text">
```

cfcollection

```
<cfcollection
  action = "action"
  collection = "collection_name"
  path = "path_to_verity_collection"
  language = "language"
  name = "queryname" >
```

cfcomponent

```
<cfcomponent
  extends ="anotherComponent">
  <cffunction ...>
    ...
  </cffunction>

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

cfcontent

```
<cfcontent
  type = "file_type"
  deleteFile = "Yes" or "No"
  file = "filename"
  reset = "Yes" or "No">
```

cfcookie

```
<cfcookie
  name = "cookie_name"
  value = "text"
  expires = "period"
  secure = "Yes" or "No"
  path = "url"
  domain = ".domain">
```

cfdefaultcase See cfswitch

cfdirectory

```
<cfdirectory
  action = "directory action"
  directory = "directory name"
  name = "query name"
  filter = "list filter"
```

```
    mode = "permission"
    sort = "sort specification"
    newDirectory = "new directory name">
```

cfdump

```
  <cfdump
    var = #variable#
    expand = "Yes or No"
    label = "text">
```

cfelse See cfif

cfelseif See cfif

cferror

```
  <cferror
    type = "a type"
    template = "template_path"
    mailTo = "email_address"
    exception = "exception_type">
```

cfexecute

```
  <cfexecute
    name = " ApplicationName "
    arguments = "CommandLine Arguments"
    outputFile = "Output file name"
    timeout = "Timeout interval">
    ...
  </cfexecute>
```

cfexit

```
  <cfexit
    method = "method">
```

cffile

```
  <cffile
    action = "upload"
    fileField = "formfield"
    destination = "full_path_name"
    nameConflict = "behavior"
    accept = "mime_type/file_type"
    mode = "permission"
    attributes = "file_attribute_or_list">
  <cffile
    action = "move"
    source = "full_path_name"
    destination = "full_path_name"
    mode = "mode"
    attributes = "file_attributes_list"
    charset = "charset_option">
  <cffile
    action = "rename"
    source = "full_path_name"
    destination = "full_path_name"
    mode = "mode"
    attributes = "file_attributes_list">
  <cffile
    action = "copy"
    source = "full_path_name"
    destination = "full_path_name"
    mode = "mode"
    attributes = "file_attributes_list">
  <cffile
    action = "delete"
    file = "full_path_name">
  <cffile
    action = "read"
    file = "full_path_name"
    variable = "var_name"
    charset = "charset_option" >
  <cffile
    action = "readBinary"
    file = "full_path_name"
    variable = "var_name">
  <cffile
    action = "write"
    file = "full_path_name"
    output = "content"
    mode = "permission"
    addNewLine = "Yes" or "No"
    attributes = "file_attributes_list"
    charset = "charset_option" >
  <cffile
    action = "append"
    file = "full_path_name"
    output = "string"
    addNewLine = "Yes" or "No"
    attributes = "file_attributes_list">
    mode = "mode"
    charset = "charset_option" >
```

cfflush

```
  <cfflush
    interval = "integer number of bytes">
```

cfform

```
  <cfform
    name = "name"
    action = "form_action"
    preserveData = "Yes" or "No"
    onSubmit = "javascript"
    target = "window_name"
    encType = "type"
    passThrough = "HTML_attribute(s)"
    codeBase = "URL"
    archive = "URL"
    scriptSrc = "path">
    ...
  </cfform>
```

cfftp

cfftp: connecting to an FTP server

```
  <cfftp
    action = "action"
    username = "name"
    password = "password"
    server = "server"
    timeout = "timeout in seconds"
    port = "port"
    connection = "name"
    proxyServer = "proxyserver"
    retryCount = "number"
    stopOnError = "Yes" or "No"
    passive = "Yes" or "No">
```

cfftp: connection: file and directory operations

```
  <cfftp
    action = "action"
    username = "name"
    password = "password"
    name = "query_name"
    server = "server"
    ASCIIExtensionList = "extensions"
    transferMode = "mode"
    failIfExists = "Yes" or "No"
    directory = "directory name"
    localFile = "filename"
    remoteFile = "filename"
    item = "directory or file"
    existing = "file or directory name"
    new = "file or directory name"
    proxyServer = "proxyserver"
    passive = "Yes" or "No">
```

cffunction

```
  <cffunction
    name = "methodName"
    returnType = "dataType"
    roles = "securityRoles"
    access = "methodAccess"
    output = "yes" or "no" >
```

cfgrid

```
  <cfgrid
    name = "name"
    height = "integer"
    width = "integer"
    autoWidth = "Yes" or "No"
    vSpace = "integer"
    hSpace = "integer"
    align = "value"
    query = "query_name"
    insert = "Yes" or "No"
    delete = "Yes" or "No"
    sort = "Yes" or "No"
    font = "column_font"
    fontSize = "size"
    italic = "Yes" or "No"
    bold = "Yes" or "No"
    textColor = "web color"
    href = "URL"
    hrefKey = "column_name"
    target = "URL_target"
    appendKey = "Yes" or "No"
    highlightHref = "Yes" or "No"
    onValidate = "javascript_function"
    onError = "text"
    gridDataAlign = "position"
    gridLines = "Yes" or "No"
    rowHeight = "pixels"
    rowHeaders = "Yes" or "No"
    rowHeaderAlign = "position"
    rowHeaderFont = "font_name"
    rowHeaderFontSize = "size"
    rowHeaderItalic = "Yes" or "No"
    rowHeaderBold = "Yes" or "No"
    rowHeaderTextColor = "web color"
    colHeaders = "Yes" or "No"
    colHeaderAlign = "position"
    colHeaderFont = "font_name"
    colHeaderFontSize = "size"
    colHeaderItalic = "Yes" or "No"
    colHeaderBold = "Yes" or "No"
    colHeaderTextColor = "web color"
    bgColor = "web color"
    selectColor = "web color"
    selectMode = "mode"
    maxRows = "number"
    notSupported = "text"
    pictureBar = "Yes" or "No"
    insertButton = "text"
    deleteButton = "text"
    sortAscendingButton = "text"
    sortDescendingButton = "text">
  </cfgrid>
```

cfgridcolumn

```
  <cfgridcolumn
    name = "column_name"
    header = "header"
    width = "column_width"
    font = "column_font"
    fontSize = "size"
    italic = "Yes" or "No"
    bold = "Yes" or "No"
    textColor = "web color" or "expression"
    bgColor = "web color" or "expression"
    href = "URL"
    hrefKey = "column_name"
    target = "URL_target"
    select = "Yes" or "No"
    display = "Yes" or "No"
    type = "type"
    headerFont = "font_name"
    headerFontSize = "size"
    headerItalic = "Yes" or "No"
    headerBold = "Yes" or "No"
    headerTextColor = "web color"
    dataAlign = "position"
    headerAlign = "position"
    numberFormat = "format"
    values = "Comma-delimited strings and/or numeric range"
    valuesDisplay="Comma-delimited strings and numeric range"
    valuesDelimiter = "delimiter character">
```

cfgridrow

```
  <cfgridrow
    data = "col1, col2, ...">
```

cfgridupdate

```
  <cfgridupdate
    grid = "gridname"
    dataSource = "data source name"
    tableName = "table name"
    username = "data source username"
    password = "data source password"
    tableOwner = "table owner"
    tableQualifier = "qualifier"
    keyOnly = "Yes" or "No">
```

cfheader

```
<cfheader
  name = "header_name"
  value = "header_value">
or
<cfheader
  statusCode = "status_code"
  statusText = "status_text">
```

cfhtmlhead

```
<cfhtmlhead
  text = "text">
```

cfhttp

```
<cfhttp
  url = "hostname"
  port = "port_number"
  method = "get_or_post"
  username = "username"
  password = "password"
  name = "queryname"
  columns = "query_columns"
  firstrowasheaders = "yes" or "no"
  path = "path"
  file = "filename"
  delimiter = "character"
  textQualifier = "character"
  resolveURL = "yes" or "no"
  proxyServer = "hostname"
  proxyPort = "port_number"
  userAgent = "user_agent"
  throwOnError = "yes" or "no"
  redirect = "yes" or "no"
  timeout = "timeout_period"
  charset = "character set">
</cfhttp>
```

cfhttpparam

```
<cfhttpparam
  name = "name"
  type = "type"
  value = "transaction type"
  file = "filename">
```

cfif

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

cfimport

```
<cfimport
  taglib = "taglib-location"
  prefix = "custom"
  webservice = "URL">
```

cfinclude

```
<cfinclude
  template = "template_name">
```

cfindex

```
<cfindex
  collection = "collection_name"
  action = "action"
  type = "type"
  title = "title"
  key = "ID"
  body = "body"
  custom1 = "custom_value"
  custom2 = "custom_value"
  URLpath = "URL"
  extensions = "file_extensions"
  query = "query_name"
  recurse = "Yes" or "No"
  language = "language">
```

cfinput

```
<cfinput
  type = "input_type"
  name = "name"
  value = "initial_value"
  required = "Yes" or "No"
  range = "min_value, max_value"
  validate = "data_type"
  onValidate = "javascript_function"
  pattern = "regexp"
  message = "validation_msg"
  onError = "text"
  size = "integer"
  maxLength = "integer"
  checked
  passThrough = "HTML_attributes">
```

cfinsert

```
<cfinsert
  dataSource = "ds_name"
  tableName = "tbl_name"
  tableOwner = "owner"
  tableQualifier = "tbl_qualifier"
  username = "username"
  password = "password"
  formFields = "formfield1, formfield2, ...">
```

cfinvoke

```
<!--- Syntax 1 - this invokes a method of a component --->
<cfinvoke
  component = "component name or reference"
  method = "method name"
  returnVariable = "variable name"
  argumentCollection = "argument collection"
  ...>
OR
<!--- Syntax 2 - this can invoke a method of a component only
  from within the component. --->
<cfinvoke
  method = "method name"
  returnVariable = "variable name"
  argumentCollection = "argument collection"
  ...>
OR
<!--- Syntax 3 - this syntax invokes a web service --->
<cfinvoke
  webservice = "URLtoWSDL_location"
  method = "operation_name"
  username = user name"
  password = "password"
  inputParam1 = "value1"
  inputParam2 = "value2"
  ...
  returnVariable = "var_name"
  ...>
OR
<!--- Syntax 4A - this syntax invokes a component.
This syntax shows instantiation with the cfobject tag.
This cfinvoke syntax applies to instantiating a component
with the cfobject tag and to instantiating a component
with the createobject function. --->
<cfobject
  component = "component name"
  name = "mystringname for instantiated object">
<cfinvoke
  <!--- value is object name, within pound signs --->
  component = "#mystringname for instantiated component#">
OR
<!--- Syntax 4B - this syntax invokes a web service.
This syntax shows instantiation with the cfobject tag.
This cfinvoke syntax applies to instantiating a web service
with the cfobject tag and to instantiating a web service
with the createobject function. --->
<cfobject
  webservice = "web service name"
  name = "mystringname for instantiated object"
  method = "operation_name">
<cfinvoke
  <!--- value is object name, within pound signs --->
  webservice="#mystringname for instantiated web service#">
```

cfinvokeargument

```
<cfinvokeargument
  name="argument name"
  value="argument value">
```

cfldap

```
<cfldap
  server = "server_name"
  port = "port_number"
  username = "name"
  password = "password"
  action = "action"
  name = "name"
  timeout = "seconds"
  maxRows = "number"
  start = "distinguished_name"
  scope = "scope"
  attributes = "attribute, attribute"
  filter = "filter"
  sort = "attribute[, attribute]..."
  sortControl = "nocase" and/or "desc" or "asc"
  dn = "distinguished_name"
  startRow = "row_number"
  modifyType = "replace" or "add" or "delete"
  rebind = "Yes" or "No"
  referral = "number_of_allowed_hops"
  secure = "multi_field_security_string"
  separator = "separator_character"
  delimiter = "delimiter_character">
```

cflocation

```
<cflocation
  url = "url"
  addToken = "Yes" or "No">
```

cflock

```
<cflock
  timeout = "timeout in seconds "
  scope = "Application" or "Server" or "Session"
  name = "lockname"
  throwOnTimeout = "Yes" or "No"
  type = "readOnly" or "exclusive ">
  <!--- CFML to be synchronized --->
</cflock>
```

cflog

```
<cflog
  text = "text"
  log = "log type"
  file = "filename"
  type = "message type"
  application = "application name yes or no">
```

cflogin

```
<cflogin
  idletimeout = "value"
  applicationToken = "token"
  cookieDomain = "domain"
  ...
  <cfloginuser
    name = "name"
    password = "password-string"
    roles = "roles">
  ...>
</cflogin>
```

cfloginuser

```
<cfloginuser
  name = "name"
  password = "password-string"
  roles = "roles">
```

cflogout

```
<cflogout>
```

cfloop

```
cfloop: index loop
<cfloop
  index = "parameter_name"
  from = "beginning_value"
  to = "ending_value"
  step = "increment">
  ... HTML or CFML code ...
</cfloop>
cfloop: conditional loop
<cfloop
  condition = "expression">
  ...
</cfloop>
```

```
cfloop: looping over a query
 <cfloop
   query = "query_name"
   startRow = "row_num"
   endRow = "row_num">
 </cfloop>
cfloop: looping over a list or file
 <cfloop
   index = "index_name"
   list = "list_items"
   delimiters = "item_delimiter">
   ...
 </cfloop>
```

cfmail

```
 <cfmail
   to = "recipient"
   from = "sender"
   cc = "copy_to"
   bcc = "blind_copy_to"
   subject = "msg_subject"
   type = "msg_type"
   maxrows = "max_msgs"
   mimeattach = "path"
   query = "query_name"
   group = "query_column"
   groupcasesensitive = "yes" or "no"
   startrow = "query_row"
   server = "servername"
   port = "port_id"
   mailerid = "headerid"
   timeout = "seconds"
   spoolenable = "yes" or "no">
```

cfmailparam

```
 <cfmail
   to = "recipient"
   subject = "msg_subject"
   from = "sender"
   ...more attributes... >

   <cfmailparam
     file = "file-name" >
   OR
   <cfmailparam
     name = "header-name"
     value = "header-value" >
   ...
 </cfmail>
```

cfmodule

```
 <cfmodule
   template = "path"
   name = "tag_name"
   attributeCollection = "collection_structure"
   attribute_name1 = "valuea"
   attribute_name2 = "valueb"
   ...>
```

cfobject

```
 <cfobject
   type = "com"
   action = "action"
   class = "program_ID"
   name = "text"
   context = "context"
   server = "server_name">
 <cfobject
   name = "variable name"
   component = "component name">
 <cfobject
   type = "corba"
   context = "context"
   class = "file or naming service"
   name = "text"
   locale = "type-value arguments">
 <cfobject
   type = "Java"
   action = "Create"
   class = "Java class"
   name = "object name">
```

```
<cfobject
  webservice="http://...?wsdl" or "name in Administrator"
  name = "myobjectname">
```

cfobjectcache

```
<cfobjectcache
  action = "clear">
```

cfoutput

```
<cfoutput
  query = "query_name"
  group = "query_column"
  groupCaseSensitive = "Yes" or "No"
  startRow = "start_row"
  maxRows = "max_rows_output">
</cfoutput>
```

cfparam

```
<cfparam
  name = "param_name"
  type = "data_type"
  default = "value">
```

cfpop

```
<cfpop
  server = "servername"
  port = "port_number"
  username = "username"
  password = "password"
  action = "action"
  name = "queryname"
  messageNumber = "number"
  uid = "number"
  attachmentPath = "path"
  timeout = "seconds"
  maxRows = "number"
  startRow = "number"
  generateUniqueFilenames = "boolean">
```

cfprocessingdirective

```
<cfprocessingdirective
  pageencoding = "page-encoding literal string">
OR
<cfprocessingdirective
  suppressWhiteSpace = "Yes" or "No"
  pageEncoding = "page-encoding literal string">
  CFML tags
</cfprocessingdirective>
```

cfprocparam

```
<cfprocparam
  type = "in" or "out" or "inout"
  variable = "variable name"
  dbVarName = "DB variable name"
  value = "parameter value"
  CFSQLType = "parameter datatype"
  maxLength = "length"
  scale = "decimal places"
  null = "Yes" or "No">
```

cfprocresult

```
<cfprocresult
  name = "query_name"
  resultSet = "1-n"
  maxRows = "maxrows">
```

cfproperty

```
<cfproperty
  name=”name”
  type=”type”
  ...>
```

cfquery

```
<cfquery
  name = "query_name"
  dataSource = "ds_name"
  dbtype = "query"
  username = "username"
  password = "password"
  maxRows = "number"
  blockFactor = "blocksize"
  timeout = "seconds"
  cachedAfter = "date"
  cachedWithin = "timespan"
```

```
    debug = "Yes" or "No"
  or:
    debug

  SQL statement(s)>
</cfquery>
```

cfqueryparam

```
<cfquery
  name = "query_name"
  dataSource = "ds_name"
  ...other attributes...
  SELECT STATEMENT WHERE column_name =
  <cfqueryparam value = "parameter value"
    CFSQLType = "parameter type"
    maxLength = "maximum parameter length"
    scale = "number of decimal places"
    null = "Yes" or "No"
    list = "Yes" or "No"
    separator = "separator character">
  AND/OR ...additional criteria of the WHERE clause...
  </cfquery>
```

cfregistry

```
<cfregistry
  action = "getAll"
  branch = "branch"
  type = "data type"
  name = "query name"
  sort = "criteria">
<cfregistry
  action = "get"
  branch = "branch"
  entry = "key or value"
  variable = "variable"
  type = "data type">
<cfregistry
  action = "set"
  branch = "branch"
  entry = "key or value"
  type = "value type"
  value = "data">
<cfregistry
  action = "delete"
  branch = "branch"
  entry = "keyorvalue">
```

cfreport

```
<cfreport
  report = "report_path"
  dataSource = "ds_name"
  type = "type"
  timeout = "number of seconds"
  orderBy = "result_order"
  username = "username"
  password = "password"
  formula = "formula">
</cfreport>
```

cfrethrow

```
<cfrethrow>
```

cfreturn

```
<cfreturn
  expr>
```

cfsavecontent

```
<cfsavecontent
  variable = "variable name">
  the content
</cfsavecontent>
```

cfschedule

```
<cfschedule
  action = "update"
  task = "taskname"
  operation = "HTTPRequest"
  file = "filename"
  path = "path_to_file"
  startDate = "date"
  startTime = "time"
  url = "URL"
  publish = "Yes" or "No"
  endDate = "date"
  endTime = "time"
```

```
    interval = "seconds"
    requestTimeOut = "seconds"
    username = "username"
    password = "password"
    resolveURL = "Yes" or "No"
    proxyServer = "hostname"
    port = "port_number"
    proxyPort = "port_number">

  <cfschedule
    action = "delete"
    task = "TaskName">

  <cfschedule
    action = "run"
    task = "TaskName">
```

cfscript

```
  <cfscript>
    cfscript code here
  </cfscript>
```

cfsearch

```
  <cfsearch
    name = "search_name"
    collection = "collection_name"
    type = "criteria"
    criteria = "search_expression"
    maxRows = "number"
    startRow = "row_number"
    language = "language">
```

cfselect

```
  <cfselect
    name = "name"
    required = "Yes" or "No"
    message = "text"
    onError = "text"
    size = "integer"
    multiple = "Yes" or "No"
    query = "queryname"
    selected = "column_value"
    value = "text"
    display = "text"
    passThrough = "HTML_attributes">
  </cfselect>
```

cfset

```
  <cfset
    variable_name = expression>
```

cfsetting

```
  <cfsetting
    enableCFoutputOnly = "Yes" or "No"
    showDebugOutput = "Yes" or "No"
    requestTimeOut = "value in seconds">
```

cfsilent

```
  <cfsilent>
  ...
  </cfsilent>
```

cfslider

```
  <cfslider
  name = "name"
    label = "text"
    refreshLabel = "Yes" or "No"
    range = "min_value, max_value"
    scale = "uinteger"
    value = "integer"
    onValidate = "script_name"
    message = "text"
    onError = "text"
    height = "integer"
    width = "integer"
    vSpace = "integer"
    hSpace = "integer"
    align = "alignment"
    tickMarkMajor = "Yes" or "No"
    tickMarkMinor = "Yes" or "No"
    tickMarkImages = "URL1, URL2, URLn"
    tickMarkLabels = "Yes" or "No" or "list"
    lookAndFeel = "motif" or "windows" or "metal"
    vertical = "Yes" or "No"
    bgColor = "color"
    textColor = "color"
    font = "font_name"
    fontSize = "integer"
    italic = "Yes" or "No"
    bold = "Yes" or "No"
    notSupported = "text">
```

cfstoredproc

```
  <cfstoredproc
    procedure = "procedure name"
    dataSource = "ds_name"
    username = "username"
    password = "password"
    blockFactor = "blocksize"
    debug = "Yes" or "No"
    returnCode = "Yes" or "No">
```

cfswitch

```
  <cfswitch
    expression = "expression">
    <cfcase
      value = "value"
      delimiters = "delimiters">
      HTML and CFML tags
    </cfcase>
    additional <cfcase></cfcase> tags
    <cfdefaultcase>
      HTML and CFML tags
    </cfdefaultcase>
  </cfswitch>
```

cftable

```
  <cftable
    query = "query_name"
    maxRows = "maxrows_table"
    colSpacing = "number_of_spaces"
    headerLines = "number_of_lines"
    HTMLTable
    border
    colHeaders
    startRow = "row_number">
    ...
  </cftable>
```

cftextinput

```
  <cftextinput
    name = "name"
    value = "text"
    required = "Yes" or "No"
    range = "min_value, max_value"
    validate = "data_type"
    onValidate = "script_name"
    message = "text"
    onError = "text"
    size = "integer"
    font = "font_name"
    fontSize = "integer"
    italic = "Yes" or "No"
    bold = "Yes" or "No"
    height = "integer"
    width = "integer"
    vSpace = "integer"
    hSpace = "integer"
    align = "alignment"
    bgColor = "color"
    textColor = "color"
    maxLength = "integer"
    notSupported = "text">
```

cfthrow

```
  <cfthrow
    type = "exception_type "
    message = "message"
    detail = "detail_description "
    errorCode = "error_code "
    extendedInfo = "additional_information"
    object = "java_except_object">
  <cfthrow
    object = #object_name#>
```

cftrace

```
  <cftrace
    abort = "Yes or No"
    category = "string"
    inline = "Yes or No"
    text = "string"
    type = "format"
    var = "variable_name"
  </cftrace>
```

cftransaction

```
  <cftransaction
    action = "begin" or "commit" or "rollback"
    isolation = "read_uncommitted" or "read_committed" or
      "repeatable_read" >
  </cftransaction>
```

cftree

```
  <cftree
    name = "name"
    required = "Yes" or "No"
    delimiter = "delimiter"
    completePath = "Yes" or "No"
    appendKey = "Yes" or "No"
    highlightHref = "Yes" or "No"
    onValidate = "script_name"
    message = "text"
    onError = "text"
    lookAndFeel = "motif" or "windows" or "metal"
    font = "font"
    fontSize = "size"
    italic = "Yes" or "No"
    bold = "Yes" or "No"
    height = "integer"
    width = "integer"
    vSpace = "integer"
    hSpace = "integer"
    align = "alignment"
    border = "Yes" or "No"
    hScroll = "Yes" or "No"
    vScroll = "Yes" or "No"
    notSupported = "text">
  </cftree>
```

cftreeitem

```
  <cftreeitem
    value = "text"
    display = "text"
    parent = "parent_name"
    img = "filename"
    imgopen = "filename"
    href = "URL"
    target = "URL_target"
    query = "queryname"
    queryAsRoot = "Yes" or "No"
    expand = "Yes" or "No">
```

cftry

```
  <cftry>
    code here
  <cfcatch type = "exceptiontype">
    Exception processing code here
  </cfcatch>
    Optional: More cfcatch blocks here
  </cftry>
```

cfupdate

```
  <cfupdate
    dataSource = "ds_name"
    tableName = "table_name"
    tableOwner = "name"
    tableQualifier = "qualifier"
    username = "username"
    password = "password"
    formFields = "field_names">
```

cfwddx

```
  <cfwddx
    action = "action"
    input = "inputdata"
    output = "resultvariablename"
    topLevelVariable = "toplevelvariablenameforjavascript"
    useTimeZoneInfo = "Yes" or "No"
    validate = "Yes" or "No" >
```

cfxml

```
  <CFXML
    variable="xmlVarName"
    caseSensitive="yes" or "no">
```

CFML functions

Array functions

```
ArrayAppend(array, value)
ArrayAvg(array)
ArrayClear(array)
ArrayDeleteAt(array, position)
ArrayInsertAt(array, position, value)
ArrayIsEmpty(array)
ArrayLen(array)
ArrayMax(array)
ArrayMin(array)
ArrayNew(dimension)
ArrayPrepend(array, value)
ArrayResize(array, minimum_size)
ArraySet(array, start_pos, end_pos, value)
ArraySort(array, sort_type [, sort_order ])
ArraySum(array)
ArraySwap(array, position1, position2)
ArrayToList(array [, delimiter ])
IsArray(value [, number ])
ListToArray(list [, delimiters ])
```

Authentication functions

```
GetAuthUser()
IsUserInRole("role_name")
```

Conversion functions

```
ArrayToList(array [, delimiter ])
Hash(string)
LCase(string)
ListToArray(list [, delimiters ])
ToBase64(string or binary_object[, encoding])
ToBinary(string_in_Base64 or binary_value)
ToString(any_value[, encoding])
URLDecode(urlEncodedString[, charset])
URLEncodedFormat(string)
Val(string)
XmlFormat(string)
XmlParse(xmlString [, caseSensitive ] )
XmlTransform(xmlString | xmlObj, xslString)
```

Date and time functions

```
CreateDate(year, month, day)
CreateDateTime(year, month, day, hour, minute, second)
CreateODBCDate(date)
CreateODBCDateTime(date)
CreateODBCTime(date)
CreateTime(hour, minute, second)
CreateTimeSpan(days, hours, minutes, seconds)
DateAdd("datepart", number, "date")
DateCompare("date1", "date2" [, "datePart"])
DateConvert("conversion-type", "date")
DateDiff("datepart", "date1", "date2")
DateFormat("date" [, "mask" ])
DatePart("datepart", "date")
Day("date")
DayOfWeek("date")
```

DayOfWeekAsString*(day_of_week)*
DayOfYear*("date")*
DaysInMonth*("date")*
DaysInYear*("date")*
FirstDayOfMonth*(date)*
GetHttpTimeString*(date_time_object)*
GetTickCount*()*
GetTimeZoneInfo*()*
Hour*(date)*
IsDate*(string)*
IsLeapYear*(year)*
IsNumericDate*(number)*
LSDateFormat*(date [, mask])*
LSIsDate*(string)*
LSParseDateTime*(date/time-string)*
LSTimeFormat*(time [, mask])*
Minute*(date)*
Month*(date)*
MonthAsString*(month_number)*
Now*()*
ParseDateTime*(date/time-string [, pop-conversion])*
Quarter*(date)*
Second*(date)*
TimeFormat*(time [, mask])*
Week*(date)*
Year*(date)*

Decision functions

DirectoryExists*(absolute_path)*
FileExists*(absolute_path)*
IIf*(condition, string_expression1, string_expression2)*
IsArray*(value [, number])*
IsBinary*(value)*
IsBoolean*(value)*
IsCustomFunction*("name")*
IsDate*(string)*
IsDebugMode*()*
IsDefined*("variable_name")*
IsK2ServerABroker*()*
IsK2ServerDocCountExceeded*()*
IsK2ServerOnline*()*
IsLeapYear*(year)*
IsNumeric*(string)*
IsNumericDate*(number)*
IsObject*(value [, type [, …]])*
IsQuery*(value)*
IsSimpleValue*(value)*
IsStruct*(variable)*
IsUserInRole*("role_name")*
IsWDDX*(value)*
IsXmlDoc*(value)*
IsXmlElement*(value)*
IsXmlRoot*(value)*
LSIsCurrency*(string)*
LSIsDate*(string)*
LSIsNumeric*(string)*
StructIsEmpty*(structure)*
StructKeyExists*(structure, "key")*

YesNoFormat*(value)*

Display and formatting functions

Cjustify*(string, length)*
DateFormat*("date" [, "mask"])*
DecimalFormat*(number)*
DollarFormat*(number)*
FormatBaseN*(number, radix)*
GetLocale*()*
HTMLCodeFormat*(string [, version])*
HTMLEditFormat*(string [, version])*

Dynamic evaluation functions

DE*(string)*
Evaluate*(string_expression1 [, string_expression2 [, …]])*
IIf*(condition, string_expression1, string_expression2)*
SetVariable*(name, value)*

Extensibility functions

CreateObject
CreateObject: COM object
CreateObject*(type, class, context, serverName)*
CreateObject: component object
CreateObject*(type, component-name)*
CreateObject: CORBA object
CreateObject*(type, context, class, locale)*
CreateObject: Java or EJB object
CreateObject*(type, class)*
CreateObject: web service object
CreateObject*(type, urltowsdl)*
XmlChildPos*(elem, childName, N)*
XmlElemNew*(xmlObj, childName)*
XmlFormat*(string)*
XmlNew*([caseSensitive])*
XmlParse*(xmlString [, caseSensitive])*
XmlSearch*(xmlDoc, xPathString)*
XmlTransform*(xmlString | xmlObj, xslString)*

Full-text search functions

GetK2ServerDocCount*()*
GetK2ServerDocCountLimit*()*
IsK2ServerABroker*()*
IsK2ServerDocCountExceeded*()*
IsK2ServerOnline*()*

International functions

DateConvert*("conversion-type", "date")*
GetHttpTimeString*(date_time_object)*
GetTimeZoneInfo*()*
GetLocale*()*
LSCurrencyFormat*(number [, type])*
LSDateFormat*(date [, mask])*
LSEuroCurrencyFormat*(currency-number [, type])*
LSIsCurrency*(string)*
LSIsDate*(string)*
LSIsNumeric*(string)*
LSNumberFormat*(number [, mask])*
LSParseCurrency*(string)*

LSParseDateTime*(date/time-string)*
LSParseEuroCurrency*(currency-string)*
LSParseNumber*(string)*
LSTimeFormat*(time [, mask])*
SetLocale*(new_locale)*

List functions

ArraySort*(array, sort_type [, sort_order])*
ArrayToList*(array [, delimiter])*
Asc*(string)*
Chr*(number)*
Cjustify*(string, length)*
Compare*(string1, string2)*
CompareNoCase*(string1, string2)*
Decrypt*(encrypted_string, seed)*
Encrypt*(string, seed)*
Find*(substring, string [, start])*
FindNoCase*(substring, string [, start])*
FindOneOf*(set, string [, start])*
FormatBaseN*(number, radix)*
GetClientVariablesList*()*
ListContains*(list, substring [, delimiters])*
ListContainsNoCase*(list, substring [, delimiters])*
ListDeleteAt*(list, position [, delimiters])*
ListFind*(list, value [, delimiters])*
ListFindNoCase*(list, value [, delimiters])*
ListFirst*(list [, delimiters])*
ListGetAt*(list, position [, delimiters])*
ListInsertAt*(list, position, value [, delimiters])*
ListLast*(list [, delimiters])*
ListLen*(list [, delimiters])*
ListPrepend*(list, value [, delimiters])*
ListQualify*(list, qualifier [, delimiters] [, elements])*
ListRest*(list [, delimiters])*
ListSetAt*(list, position, value [, delimiters])*
ListSort*(list, sort_type [, sort_order] [, delimiters])*
ListToArray*(list [, delimiters])*
ListValueCount*(list, value [, delimiters])*
ListValueCountNoCase*(list, value [, delimiters])*
LJustify*(string, length)*
ReplaceList*(string, list1, list2)*
RJustify*(string, length)*

Mathematical functions

Abs*(number)*
ACos*(number)*
ArrayAvg*(array)*
ArraySum*(array)*
ASin*(number)*
Atn*(number)*
BitAnd*(number1, number2)*
BitMaskClear*(number, start, length)*
BitMaskRead*(number, start, length)*
BitMaskSet*(number, mask, start, length)*
BitNot*(number)*
BitOr*(number1, number2)*
BitSHLN*(number, count)*
BitSHRN*(number, count)*

`BitXor`*(number1, number2)*
`Ceiling`*(number)*
`Cos`*(number)*
`DecrementValue`*(number)*
`Exp`*(number)*
`Fix`*(number)*
`FormatBaseN`*(number, radix)*
`IncrementValue`*(number)*
`InputBaseN`*(string, radix)*
`Int`*(number)*
`Log`*(number)*
`Log10`*(number)*
`Max`*(number1, number2)*
`Min`*(number1, number2)*
`Pi`*()*
`Rand`*()*
`Randomize`*(number)*
`RandRange`*(number1, number2)*
`Round`*(number)*
`Sgn`*(number)*
`Sin`*(number)*
`Sqr`*(number)*
`Tan`*(number)*

Other functions

`CreateUUID`*()*
`Decrypt`*(encrypted_string, seed)*
`Encrypt`*(string, seed)*
`GetBaseTagData`*(tagname [, instancenumber])*
`GetBaseTagList`*()*
`GetBaseTemplatePath`*()*
`GetClientVariablesList`*()*
`GetTickCount`*()*
`Hash`*(string)*
`PreserveSingleQuotes`*(variable)*
`QuotedValueList`*(query.column [, delimiter])*
`StripCR`*(string)*
`ToBase64`*(string or binary_object[, encoding])*
`ToBinary`*(string_in_Base64 or binary_value)*
`ToString`*(any_value[, encoding])*
`URLDecode`*(urlEncodedString[, charset])*
`URLEncodedFormat`*(string)*
`URLSessionFormat`*(request_URL)*
`ValueList`*(query.column [, delimiter])*
`WriteOutput`*(string)*

Query functions

`IsQuery`*(value)*
`QueryAddColumn`*(query, column-name, array-name)*
`QueryAddRow`*(query [, number])*
`QueryNew`*(columnlist)*
`QuerySetCell`*(query, column_name, value [, row_number])*
`ValueList`*(query.column [, delimiter])*

String functions

`Asc`*(string)*
`Chr`*(number)*
`Cjustify`*(string, length)*

```
Compare(string1, string2)
CompareNoCase(string1, string2)
DayOfWeekAsString(day_of_week)
Decrypt(encrypted_string, seed)
Encrypt(string, seed)
Find(substring, string [, start ])
FindNoCase(substring, string [, start ])
FindOneOf(set, string [, start ])
GetToken(string, index [, delimiters ])
Hash(string)
Insert(substring, string, position)
JavaCast(type, variable)
JSStringFormat(string)
LCase(string)
Left(string, count)
Len(string or binary object)
ListValueCount(list, value [, delimiters ])
ListValueCountNoCase(list, value [, delimiters ])
LJustify(string, length)
LSIsCurrency(string)
LSIsDate(string)
LSIsNumeric(string)
LSParseCurrency(string)
LSParseDateTime(date/time-string)
LSParseEuroCurrency(currency-string)
LSParseNumber(string)
LTrim(string)
Mid(string, start, count)
MonthAsString(month_number)
ParagraphFormat(string)
ParseDateTime(date/time-string [, pop-conversion ] )
REFind(reg_expression, string [, start]
     [, returnsubexpressions ])
REFindNoCase(reg_expression, string [, start]
     [, returnsubexpressions] )
RemoveChars(string, start, count)
RepeatString(string, count)
Replace(string, substring1, substring2 [, scope ])
ReplaceList(string, list1, list2)
ReplaceNoCase(string, substring1, substring2 [, scope ])
REReplace(string, reg_expression, substring [, scope ])
REReplaceNoCase(string, reg_expression, substring [, scope ])
Reverse(string)
Right(string, count)
RJustify(string, length)
RTrim(string)
SpanExcluding(string, set)
SpanIncluding(string, set)
StripCR(string)
ToBase64(string or binary_object[, encoding])
ToBinary(string_in_Base64 or binary_value)
ToString(any_value[, encoding])
Trim(string)
UCase(string)
URLDecode(urlEncodedString[, charset])
URLEncodedFormat(string)
Val(string)
XmlFormat(string)
```

Structure functions

Duplicate*(variable_name)*
IsStruct*(variable)*
StructAppend*(struct1, struct2, overwriteFlag)*
StructClear*(structure)*
StructCopy*(structure)*
StructCount*(structure)*
StructDelete*(structure, key [, indicatenotexisting])*
StructFind*(structure, key)*
StructFindKey*(top, value, scope)*
StructFindValue*(top, value [, scope])*
StructGet*(pathDesired)*
StructInsert(*structure*, *key*, *value [, allowoverwrite]*)
StructIsEmpty*(structure)*
StructKeyArray*(structure)*
StructKeyExists*(structure, "key")*
StructKeyList*(structure [, delimiter])*
StructNew*()*
StructSort*(base, sortType, sortOrder, pathToSubElement)*
StructUpdate*(structure, key, value)*

System functions

DirectoryExists*(absolute_path)*
ExpandPath*(relative_path)*
FileExists*(absolute_path)*
GetBaseTemplatePath*()*
GetCurrentTemplatePath*()*
GetDirectoryFromPath*(path)*
GetException*(object)*
GetFileFromPath*(path)*
GetFunctionList*()*
GetHttpRequestData*()*
GetLocale*()*
GetMetaData*(object)*
  or, if used within a ColdFusion component:
GetMetaData*(this)*
GetMetricData*(mode)*
GetPageContext*()*
GetProfileSections*(iniFile)*
GetProfileString*(iniPath, section, entry)*
GetServiceSettings*()*
GetTempDirectory*()*
GetTempFile*(dir, prefix)*
GetTimeZoneInfo*()*

XML functions

IsWDDX*(value)*
IsXmlDoc*(value)*
IsXmlElement*(value)*
IsXmlRoot*(value)*
XmlChildPos*(elem, childName, N)*
XmlElemNew*(xmlObj, childName)*
XmlFormat*(string)*
XmlNew*([caseSensitive])*
XmlParse*(xmlString [, caseSensitive])*
XmlSearch*(xmlDoc, xPathString)*
XmlTransform*(xmlString | xmlObj, xslString)*

ColdFusion 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.

You use the `cflock` tag to limit the scope of CFML constructs that modify shared shared data structures, files, and CFXs, to ensure that modifications occur sequentially. See Developing ColdFusion MX Applications with CFML.

Variable scope

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

Client variables

The following client variables are read-only:

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

Server variables

To reference the variables, use the Server prefix, as follows:

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

Application and session variables

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

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

To ensure that modifications to shared data occur in the intended sequence, use the `cflock` tag.

Predefined application and session variables are as follows:

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

Custom tag variables

A ColdFusion custom tag returns the following variables:

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

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

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

The calling page can access the variable as follows:

```
<cfoutput>#Caller.variable_name#</cfoutput>
```

Request variables

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

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

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

A nested tag can access the variable as follows:

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

Form variable

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

```
Form.FieldNames
```

ColdFusion tag-specific variables

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

The following tags return data that can be referenced in variables:

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

ColdFusion query variables

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

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

cfcatch variables

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

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

cfdirectory variables

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

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

cferror variables

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

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

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

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

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

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

Note: Iif `type = "Exception"` or `"Monitor"`, you can substitute the prefix CFERROR for Error; for example, CFERROR.Diagnostics, CFERROR.Mailto or CFERROR.DateTime.

cffile action=upload variables

File variables are read-only. To reference file variables, use the CFFILE prefix; for example, CFFILE.ClientDirectory. (The FILE prefix is deprecated; use the CFFILE prefix.)

```
CFFILE.AttemptedServerFile
CFFILE.ClientDirectory
CFFILE.ClientFile
CFFILE.ClientFileExt
CFFILE.ClientFileName
CFFILE.ContentSubType
CFFILE.ContentType
CFFILE.DateLastAccessed
CFFILE.FileExisted
CFFILE.FileSize
CFFILE.FileWasAppended
CFFILE.FileWasOverwritten
CFFILE.FileWasRenamed
CFFILE.FileWasSaved
CFFILE.OldFileSize
```

```
    CFFILE.ServerDirectory
    CFFILE.ServerFile
    CFFILE.ServerFileExt
    CFFILE.ServerFileName
    CFFILE.TimeCreated
    CFFILE.TimeLastModified
```

cfftp error variables

If you use the `cfftp stoponerror` attribute, these variables are populated:

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

cfftp ReturnValue variable

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

```
    GetCurrentDir
    GetCurrentURL
    ExistsDir
    ExistsFile
    Exists
```

cfftp query object columns

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

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

cfhttp variables

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

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

cfldap variables

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

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

cfpop variables

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

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

cfquery and cfstoredproc variables

The `cfquery` tag returns information about the query in the variable CFQUERY.ExecutionTime.

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

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

The `cfstoredproc` tag returns the following variables:

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

cfregistry variables

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

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

cfsearch variables

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

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

Standard CGI variables

This section lists the CGI 1.1 variables that some web servers create when a CGI script is called.

The CGI variables that are available vary with the web server and configuration.

Request

```
CGI.AUTH_TYPE
CGI.CONTENT_LENGTH
CGI.CONTENT_TYPE
CGI.PATH_INFO
CGI.PATH_TRANSLATED
CGI.QUERY_STRING
CGI.REMOTE_ADDR
CGI.REMOTE_HOST
CGI.REMOTE_USER
CGI.REQUEST_METHOD
CGI.SCRIPT_NAME
```

Server

```
CGI.GATEWAY_INTERFACE
CGI.SERVER_NAME
CGI.SERVER_PORT
CGI.SERVER_PROTOCOL
CGI.SERVER_SOFTWARE
```

Client

```
CGI.CERT_ISSUER
CGI.CERT_SUBJECT
CGI.CLIENT_CERT_ENCODED
CGI.HTTP_ACCEPT
CGI.HTTP_IF_MODIFIED_SINCE
CGI.HTTP_USER_AGENT
```

The `CERT_ISSUER`, `CERT_SUBJECT, CLIENT_CERT_ENCODED` variables are available only when you use client certificates.