Developing Applications

ADOBE® COLDFUSION (2016 release)

Contents

Chapter 1: Changes in ColdFusion

Changes in ColdFusion

New in ColdFusion 11

ColdFusion 11 has gone through a lot of changes and enhancements and this section highlights those changes:

End-to-end mobile development

The ColdFusion Mobile Platform aims at providing a server and development infrastructure that facilitates rapid and robust mobile application development, debugging, packaging, and deployment. The ColdFusion 11 release introduces rapid application development through ColdFusion Builder 3. This release of ColdFusion introduces full-fledged on-device debugging to quickly debug your ColdFusion-based mobile applications on devices.

You can quickly build a mobile application by reading the information provided in the following sections:

A new lightweight edition

The **ColdFusion Express** enables you to quickly setup a development or demonstration instance of a Server without following the installation procedure. The ColdFusion Express is a new 'lighter' version of the ColdFusion Server and is ideally suited for developers to quickly setup and run a ColdFusion Server for testing and debugging purposes.

See Installing ColdFusion Express

Language enhancements

ColdFusion 11 has gone through various language enhancements that will provide a better development experience for ColdFusion developers. The core CFML language enhancements in ColdFusion 11 includes new language constructs, extended tag support, enhanced script functions, and support for new operations.

See ColdFusion Language Enhancements

WebSocket enhancements

ColdFusion 11 has introduced the proxy support for WebSocket. There is a new proxy module (that runs inside IIS and Apache Web Server) that can intercept the ColdFusion WebSocket requests and redirect the requests to the ColdFusion Server.

See WebSocket enhancements

PDF generation enhancements

To generate high quality PDFs from HTML documents, 2 new tags, **<cfhtmltopdf>** and **<cfhtmltopdfitem>** have been introduced in ColdFusion 11.

See PDF generation enhancements

Security enhancements

ColdFusion 11 has a lot of security enhancements and bug fixes. This update fixes a few security issues and has strengthened the Server to a large extent. Some notable security enhancements are described in the following document:

See Security Enhancements (ColdFusion 11)

Social enhancements

ColdFusion 11 has introduced the support for dynamically generating Like button, Tweet button, and Comment box for social media sites.

See Social Enhancements

REST enhancements

ColdFusion 11 now supports site-level REST applications and enables pluggable serializer and deserializer.

See REST Enhancements in ColdFusion 11

Charting enhancements

The server-side charting introduced in ColdFusion 10 that allowed you to create highly interactive charts has been further enhanced to produce visually more appealing charts.

See Charting enhancements

Compression enhancements

The following sections describe the enhancements made to the <cfzip> and <cfzipparam> tags:

- cfzip
- cfzipparam

New functions

The following new functions are added:

- GetSafeHTML
- isSafeHTML
- ImageGetMetadata
- GeneratePBKDFKey

Changes in functions

The following functions are enhanced:

- Canonicalize
- cflocation
- EncodeForCSS
- EncodeForHTML
- EncodeForHTMLAttribute
- EncodeForJavaScript
- EncodeForURL
- EncodeForXML

Restrictions

When you are using ColdFusion Scheduler , output can be saved only as .log or .txt files. The same restriction is applicable for validation queries on databases.

Also, for the <cfinclude> tag, this restriction is applicable. By default, you can include only CFM files. However, you can modify allowedextinclude key in neoruntime.xml file to add your own file type.

Deprecated

- For the <cfsetting> tag, URL.RequestTimeout attribute has been removed in ColdFusion 11.
- The HTMLEditFormat() function has ben deprecated.

See also

- Replacement of JRun with Tomcat
- Security enhancements
- ColdFusion WebSocket
- ColdFusion closures
- Enhanced Java integration
- ColdFusion ORM search for indexing and search
- Solr enhancements
- Scheduler enhancements
- Integration with Microsoft Exchange Server 2010
- Lazy loading across client and server in ColdFusion
- Web service enhancements
- RESTful Web Services in ColdFusion
- Media Player enhancements
- Displaying geolocation
- Client-side charting
- Caching enhancements
- Server update using ColdFusion Administrator
- Secure Profile for ColdFusion Administrator

Replacement of JRun with Tomcat

Instead of JRun, Tomcat is embedded with a stand-alone ColdFusion installation. Previous versions of the ColdFusion installer (before ColdFusion 10) offered a "multi-server" deployment option, whereas starting with ColdFusion 10, only a "server" deployment option is offered. Even so, after installing ColdFusion in server mode, users of ColdFusion Enterprise, Developer, or Trial editions can still create multiple instances and clusters with the Enterprise Manager in the ColdFusion Administrator. For details, Installing Adobe ColdFusion.

Security enhancements

Security enhancements helps to reduce security vulnerabilities, particularly the ones resulting from threats posed by XSS and CSRF attacks. The release also includes enhancements that help you manage ColdFusion sessions effectively. For details, see Security enhancements in ColdFusion 10 .

ColdFusion WebSocket

Develop realtime applications for stock, charting, online gaming, social networking, dashboard for various purposes, or monitoring using COldFusion WebSocket. ColdFusion implements WebSocket by providing a messaging layer for the WebSocket protocol, which you can easily control using CFML and JavaScript. For details, see Using ColdFusion WebSocket .

ColdFusion closures

For details, see Using closures .

Enhanced Java integration

Load Java libraries from a custom path. For details, see Enhanced Java integration in ColdFusion

ColdFusion ORM search for indexing and search

Use the indexing and search capabilities of ColdFusion ORM. When you develop an application that uses ColdFusion ORM, the search feature facilitates full text search. You can load all the persistent entities that match your search criteria based on a query text. For details, see ColdFusion ORM search

Solr enhancements

- Use Data Import Handler for database indexing
- Index and search based on dynamic custom fields
- Reload individual collections
- Add languages for search
- Secure your search system
- Autocommit indexed documents
- Boost specific fields or entire document for improved search results For details, see Solr enhancements in ColdFusion 10 .

Scheduler enhancements

Schedule your tasks in a granular, scalable, and organized way. The release supports Quartz scheduling service. For details, see Using Scheduler .

Integration with Microsoft Exchange Server 2010

Adobe ColdFusion can interact with Microsoft Exchange Server 2010 SP1. The enhancements offer support for Microsoft Exchange Web Services (EWS) which brings in efficacy with the following operations:

- Folder operations such as create, modify, or delete.
- Get rooms and roomlist in the exchange organization.
- Information on user availability, that helps effective scheduling.
- Conversation operations such as find conversation details, copy, move, and the status if the conversation is read. For details, see Connect to Microsoft Exchange Server 2010 .

Lazy loading across client and server in ColdFusion

Need-based loading of related entities for applications that use ColdFusion ORM in the back end and Flex as the front end is possible in this release. Your application can now fetch the main entity and not return the related entities. Only when the client application tries to access the related entities, they are loaded. For details, see Lazy loading across client and server .

Web service enhancements

ColdFusion has Axis 2 Web service framework integrated. This enables your web services to use WSDL 2 specifications, SOAP 1.2 protocol, and document literal wrapped style. Also enhancements from ColdFusion 10 resolve many interoperability issues that you might encounter while working with Web services in ColdFusion 9. For details, see Web service enhancements in ColdFusion .

Media Player enhancements

The enhancements in this release support

- Play back capability for HTML 5 videos
- Fallback to HTML 5 video playback if Flash player is not installed
- Browser independent video controls
- Dynamic streaming of Flash videos
- Advanced skinning for media player
- Play list for Flash videos
- Embedding subtitles in SRT format using HTML track element
- Extending media player using plug-ins built using Open Source Media Framework (OSMF), for example to:
- Play videos in the YouTube server
- Use stage video support by showing advertisements within the videos in linear and non-linear mode
- Adding title to the video

Note: The mediaplayer's behavior is subject to the video capability of your device.

Displaying geolocation

Displays user location on the map if the attribute showUser is specified in cfmap. This feature works only on HTML 5 compliant browsers.

The following sections describe the changes to the tags cfmap and cfmapitem to display the user location.

Client-side charting

ColdFusion 10 supports client-side charting. This is in addition to the existing server-side charting feature (which continues to serve the way it used to).

Client-side charting supports the following:

- **Dynamic and interactive charting:** Modify the charts, add styles, and add new series or plots.
- **Popular chart formats with appropriate fallback functionality:** Use HTML 5, Flash, SVG, or VML charts. If your browser does not support HTML 5 features relevant to charting, charts are rendered in Flash. Similarly, if Flash is not supported, charts are rendered in HTML.

- **Features identical to server-side charting:** Most of the server-side charting features are available with client-side charting.
- * Old and new charts:* In addition to the contemporary chart types, offers a new set of charts.
- **Needs minimal trips to server:** As compared to generating charts at server-level, for every user interaction.

Caching enhancements

- Application-specific caching
- Enhanced query caching using Ehcache For details, see Caching enhancements in ColdFusion 10 in Optimizing ColdFusion applications .

Server update using ColdFusion Administrator

Verify if there are any product updates using the ColdFusion Administrator (Server Update > Update). The updates can include hot fixes and security hot fixes for ColdFusion. For details, see Configuring and Administering Adobe ColdFusion.

Secure Profile for ColdFusion Administrator

ColdFusion allows you to secure ColdFusion server furthermore by enabling or disabling selected settings on the ColdFusion Administrator. When installing ColdFusion, you can enable Secure Profile by selecting the option when prompted on the Secure Profile screen. Further, you could provide a comma separate list of IP addresses that may be allowed to access the ColdFusion Administrator. For more information, see Enabling Secure Profile for ColdFusion Administrator .

Chapter 2: Introduction to application development using ColdFusion

Introduction to application development using ColdFusion

The guide is intended for web application programmers who are learning ColdFusion or want to extend their ColdFusion programming knowledge. It provides a solid grounding in the tools that ColdFusion provides to develop many different types of web applications of varying complexity.

Using the Developing ColdFusion Applications guide

The Developing ColdFusion Applications guide includes basic and advanced information on CFML. However, it is most useful if you have basic ColdFusion experience or have viewed the Getting Started experience, which is available from the Adobe ColdFusion Administrator. Use the guide in conjunction with the CFML Reference, which contains detailed information on the CFML language elements.

About Adobe ColdFusion documentation for Developers

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

Chapter 3: Introducing ColdFusion

About JEE and the ColdFusion architecture

As the Internet software market has matured, the infrastructure services required by distributed Internet applications, including ColdFusion applications, have become increasingly standardized. The most widely adopted standard today is the Java Enterprise Edition (JEE, formerly "J2EE") specification. JEE provides a common set of infrastructure services for accessing databases, protocols, and operating system functionality, across multiple operating systems.

About ColdFusion and the JEE platform

About ColdFusion

Adobe ColdFusion is a rapid scripting environment server for creating dynamic Internet Applications. ColdFusion Markup Language (CFML) is a tag-based scripting language that is easy to learn. CFML provides connectivity to enterprise data and powerful built-in search and charting capabilities. ColdFusion enables developers to easily build and deploy dynamic websites, content publishing systems, self-service applications, commerce sites, and more.

ColdFusion pages are plain text files that you use to create web applications. You can create your ColdFusion applications by writing all the code manually or by using wizards (provided with some editors) to generate the majority of the code for you.

ColdFusion Administrator

Elements of ColdFusion

Saving ColdFusion pages

Testing ColdFusion pages

About Internet applications and web application servers

With ColdFusion, you develop Internet applications that run on web application servers.

About web application servers

About web pages and Internet applications

Introducing ColdFusion

You use Adobe ColdFusion to create dynamic Internet applications.

Chapter 4: The CFML Programming Language

The CFML Programming Language

In this section, you can understand using ColdFusion variables, expressions, number signs, arrays, structures and on extending the ColdFusion pages with CFML scripting.

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.

Using ColdFusion Variables

Adobe ColdFusion variables are the most frequently used operands in ColdFusion expressions. Variable values can be set and reset, and can be passed as attributes to CFML tags. Variables can be passed as parameters to functions, and can replace most constants. To create and use ColdFusion variables, you should know the following:

- How variables can represent different types of data
- How the data types get converted
- How variables exist in different scopes
- How scopes are used

- How to use variables correctly

See also

- Creating variables
- Data types- Developing guide
- Using periods in variable references
- Data type conversion
- About scopes
- Ensuring variable existence
- Validating data
- Passing variables to custom tags and UDFs

Using Expressions and Number Signs

In CFML, you create expressions by using number signs to indicate expressions in Adobe ColdFusion tags such as cfoutput , in strings, and in expressions. You also use variables in variable names and strings to create dynamic expressions, and dynamic variables.

Expressions-Developing guide

Using number signs

Dynamic expressions and dynamic variables

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.

Extending ColdFusion Pages with CFML Scripting

Adobe ColdFusion offers a server-side scripting language, CFScript, that provides ColdFusion functionality in script syntax. This JavaScript-like language gives developers the same control flow as ColdFusion, but without tags. You can also use CFScript to write user-defined functions that you can use anywhere that a ColdFusion expression is allowed.

Using closures

ColdFusion Language Enhancements

ColdFusion has gone through various language enhancements that will provide a better development experience for ColdFusion developers. The core CFML language enhancements in ColdFusion 11 includes new language constructs, extended tag support, enhanced script functions, and support for new operations.

This document provides an overview on the language enhancements and changes made in ColdFusion 11.

Script support for tags

See Script support for tags .

Script support for tags

Using custom tags in scripts

See Using custom tags in scripts.

Query tags as ColdFusion functions

See QueryExecute.

JSON serialization

JSON serialization allows you to convert ColdFusion data into a JSON (JavaScript Object Notation) representation of the data. This feature was made available in ColdFusion 8.

See SerializeJSON.

Improved JSON Serialization

Member functions for data structure

See Using the member functions.

Support for Elvis operator (?:)

See Elvis operator.

Passing array index to callback functions in ArrayEach

Currently, ColdFusion supports passing objects in callback function. Now, from ColdFusion 11, you can pass the index of an array to the callback function. See ArrayEach.

Support for ListEach

A new function, ListEach, has been introduced. See ListEach.

Support for QueryGetRow

A new function, QueryGetRow, has been introduced. See QueryGetRow.

Preferences for built-in functions

See Built-in functions as first class citizen.

Language Enhancements in ColdFusion Splendor - Promoting built-in CF function to first class

Support for database queries

You can start using the <cfquery> tag in client-side CFML just like how you are currently using it in server-side CFML code. Note that not all of the <cfquery> features are supported in this release.

See Mobile Application Development

Added script support for cfimport based prefix custom tag. Now you can use prefix based custom tag in script block without any issue.

<cfscript> cfimport(taglib=" ../importFolder/" prefix="myTags"); myTags:customTag(); </cfscript> customTag.cfm <cfoutput > Output from custom tag </cfoutput>

Built-in functions as first class citizen

The ColdFusion built-in functions will be treated as 'first-class' functions so that any built-in function can be passed as an argument.

For instance, this is valid:

```
<cfscript>
function convertCaseForArray(Array array, function convertor)
{
for (var i=1; i <= arrayLen(array); i++){
array[i] = convertor(array[i]);
}
return array;
}
// lcase built-in function is being passed as callback.
resultantArray = convertCaseForArray(['One', 'Two','Three'], lcase);
writedump(resultantArray);
</cfscript>
```

Now, you can treat the built-in CFML functions like ucase() as objects, being able to assign them to variables, and pass them as arguments.

Chapter 5: Building Blocks of ColdFusion Applications

Building Blocks of ColdFusion Applications

To build robust ColdFusion Applications, you can create ColdFusion Elements, write and call user-defined Functions, create or use the custom CFML tags and/or build using the custom CFXAPI tags.

Creating ColdFusion Elements

You can create ColdFusion elements to organize your code. When you create any of these elements, you write your code once and use it, without copying it, in many places.

Writing and Calling User-Defined Functions

Creating custom functions for algorithms or procedures that you call frequently lets you organize and reuse the functions in your Adobe ColdFusion application pages.

Building and Using ColdFusion Components

A ColdFusion component (CFC) file contains data and functions that you define in related, multiple methods. You use CFC pages to organize related actions in one file, which provide can simplify your programming. For more information on creating applications that use CFCs, see the Adobe website: www.adobe.com.

Creating and Using Custom CFML Tags

You can extend CFML by creating and using custom CFML tags that encapsulate common code.

Refer to Pete's explanation of custom tags here

Building Custom CFXAPI Tags

Sometimes, the best approach to application development is to develop elements of your application by building executable to run with ColdFusion. Perhaps the application requirements go beyond what is currently feasible in CFML. Perhaps you can improve application performance for certain types of processing. Or, you have existing code that already solves an application problem and you want to incorporate it into your ColdFusion application. To meet these types of requirements, you can use the ColdFusion Extension Application Programming Interface (CFX API) to develop custom ColdFusion tags based on Java or C++.

Using the member functions

A lot of enhancements have been made to the core CFML syntax that will aid in developing ColdFusion-based applications elegantly. One of the primary enhancements is the introduction of member functions for data structure and data objects. You can now start coding in a true object oriented style.

For instance, consider the following headless function:

```
ArrayAppend (empArr, emp)
```

It can now be written as:

```
empArr.append(emp)
```

where 'arrayObj' is a reference to the CFArray class.

The following example depicts the new usage of member functions:

```
<cfscript> //The old way var myArray = ArrayNew(1); ArrayAppend(myArray, "objec_new"); ArraySort(myArray, "ASC"); // The new way
myArray.append("objec_new"); myArray.sort("ASC"); // The new way var myProductObject = createObject("java", "myJavaclass"); myjavaList =
myProductObject.getProductList(); myjavaList.add("newProduct"); // Java API myjavaList.append("newProduct"); // CF API
myjavaList.sort("ASC"); </cfscript>
```

Member functions for ColdFusion data types

Member function for the following data types are supported:

- Array
- String

- List
- Struct
- Date
- Spreadsheet
- XML
- Query
- Image

Supported Array member functions

The following Array member functions are supported:

ArrayAppend	someVar.append()
ArrayAvg	someVar.avg()
ArrayClear	someVar.clear()
ArrayContains	someVar.contains()
ArrayDelete	someVar.delete()
ArrayDeleteAt	someVar.deleteAt()
ArrayEach	someVar.each()
ArrayFilter	someVar.filter()
ArrayFind	someVar.find()
ArrayFindAll	someVar.findAll()
ArrayFindAllNoCase	someVar.findAllNoCase()
ArrayFindNoCase	someVar.findNoCase()
ArrayInsertAt	someVar.insertAt()
ArrayIsDefined	someVar.isDefined()
ArrayIsEmpty	someVar.isEmpty()
ArrayLen	someVar.len()
ArrayMap	someVar.map()
ArrayMax	someVar.max()
ArrayMin	someVar.min()
ArrayPrepend	someVar.prepend()
ArrayResize	someVar.resize()
ArraySet	someVar.set()
ArraySlice	someVar.slice()
ArraySort	someVar.sort()

ArraySum	someVar.sum()
ArraySwap	someVar.swap()
ArrayToList	someVar.toList()

Supported String member functions

The following String member functions are supported:

CJustify	someVar.cJustify()
Compare	someVar.compare()
CompareNoCase	someVar.compareNocase()
Find	someVar.find()
FindNoCase	someVar.findNoCase()
FindOneOf	someVar.findOneOf()
GetToken	someVar.getToken()
Insert	someVar.insert()
LCase	someVar.lCase()
LJustify	someVar.lJustify()
Left	someVar.left()
Len	someVar.len()
Mid	someVar.mid()
RJustify	someVar.rJustify()
RTrim	someVar.trim()
RemoveChars	someVar.removeChars()
RepeatString	someVar.repeatString()
Replace	someVar.replace()
ReplaceList	someVar.replaceList()
ReplaceNoCase	someVar.replaceNocase()
Reverse	someVar.reverse()
Right	someVar.right()
SpanExcluding	someVar.spanExcluding()
SpanIncluding	someVar.spanIncluding()
StripCR	someVar.stripCR()
Trim	someVar.trim()
UCase	someVar.uCase()
Wrap	someVar.wrap()

Similarly, the following String member functions are also supported:

Decrypt, Encrypt, BinaryDecode, BinaryEncode, CharsetDecode, CharsetEncode, URLDecode, URLEncodedFormat, HTMLEditFormat, HTMLCodeFormat, ParagraphFormat, JSStringFormat, XmlFormat, FormatBaseN, HTMLEditFormat, HTMLCodeFormat, ParagraphFormat, ToBinary, ToString, ToBase64, Val, GenerateSecretKey, Hash, REFind, REFindNoCase, REMatch, REMatchNoCase, REReplace, REReplaceNoCase, ReplaceList, LSParseEuroCurrency, LSParseDateTime, LSIsCurrency, LSIsDate, LSIsNumeric, LSParseCurrency, LSParseNumber, and ParseDateTime

Supported List member functions

The following List member functions are supported:

ListAppend	someVar.listAppend()
ListChangeDelims	someVar.listChangeDelims()
ListContains	someVar.listContains()
ListContainsNoCase	someVar.listContainsNoCase()
ListDeleteAt	someVar.listDeleteAt()
ListEach	someVar.listEach()
ListFind	someVar.listFind()
ListFindNoCase	someVar.listFindNoCase()
ListFirst	someVar.listFirst()
ListGetAt	someVar.listGetAt()
ListInsertAt	someVar.listInsertAt()
ListLast	someVar.listLast()
ListLen	someVar.listLen()
ListMap	someVar.listMap()
ListPrepend	someVar.listPrepend()
ListQualify	someVar.listQualify()
ListReduce	someVar.listReduce()
ListRest	someVar.listRest()
ListSetAt	someVar.listSetAt()
ListSort	someVar.listSort()
ListToArray	someVar.listToArray()
ListValueCount	someVar.listValueCount()
ListValueCountNoCase	someVar.listValueCountNoCase()

Supported Struct member functions

The following Struct member functions are supported:

StructIsEmpty	someVar.isEmpty()
StructAppend	someVar.append()
StructClear	someVar.clear()

StructCopy	someVar.copy()
StructCount	someVar.count()
StructDelete	someVar.delete()
StructFind	someVar.find()
StructFindValue	someVar.findValue()
StructUpdate	someVar.update()
StructSort	someVar.sort()
StructInsert	someVar.insert()
StructEach	someVar.each()
StructKeyArray	someVar.keyArray()
StructKeyExists	someVar.keyExists()
StructKeyList	someVar.keyList()

Supported Date member functions

The following Date member functions are supported:

CreateODBCDate	someVar.createODBCDate()
CreateODBCDateTime	someVar.createODBCDateTime()
DateDiff	someVar.diff()
CreateODBCTime	someVar.createODBCTime()
LSDateFormat	someVar.lsDateFormat()
DatePart	someVar.datepart()
DaysInYear	someVar.daysIn
Second	someVar.second()
Minute	someVar.minute()
Hour	someVar.hour()
Day	someVar.day()
Week	someVar.week()
Month	someVar.month()
Quarter	someVar.quarter()
Year	someVar.year()
DaysInMonth	someVar.daysInMonth()
DayOfWeek	someVar.dayOfweek()
DayOfYear	someVar.dayOfYear()
FirstDayOfMonth	someVar.firstDayOfMonth()
DateTimeFormat	someVar.dateTimeFormat()

TimeFormat	someVar.timeFormat()
DateFormat	someVar.dateFormat()
DateAdd	someVar.add()
DateConvert	someVar.convert()

Supported Image member functions

The following Image member functions are supported:

ImageGetWidth	someVar.getWidth()
ImageSetDrawingColor	someVar.setDrawingColor()
ImageGetBufferedImage	someVar.getBufferedImage()
ImageTranslateDrawingAxis	someVar.translateDrawingAxis()
ImageSetDrawingStroke	someVar.setDrawingStroke()
ImageNegative	someVar.negative()
ImageCopy	someVar.copy()
ImageDrawRect	someVar.drawRect()
ImageCrop	someVar.crop()
ImageGetHeight	someVar.getHeight()
ImageGetIPTCTag	someVar.getIPTCTag()
ImageDrawOval	someVar.drawOval()
ImageSharpen	someVar.sharpen()
ImageOverlay	someVar.overlay()
ImageGetEXIFTag	someVar.getEXIFTag()
ImageDrawBeveledRect	someVar.drawBeveledRect()
ImageAddBorder	someVar.addBorder()
ImageShear	someVar.shear()
ImageInfo	someVar.info()
ImagePaste	someVar.paste()
ImageDrawArc	someVar.drawArc()
ImageShearDrawingAxis	someVar.shearDrawingAxis()
ImageDrawRoundRect	someVar.drawRoundRect()
ImageGrayscale	someVar.grayscale()
ImageSetDrawingTransparency	someVar.setDrawingTransparency()
ImageScaleToFit	someVar.scaleToFit()
ImageClearRect	someVar.clearRect()
ImageTranslate	someVar.translate()

ImageFlip	someVar.flip()
ImageWriteBase64	someVar.writeBase64()
ImageSetBackgroundColor	someVar.setBackgroundColor()
ImageDrawLine	someVar.drawLine()
ImageDrawQuadraticCurve	someVar.drawQuadraticCurve()
ImageRotate	someVar.rotate()
ImageGetBlob	someVar.getBlob()
ImageWrite	someVar.write()
ImageBlur	someVar.blur()
ImageRotateDrawingAxis	someVar.rotateDrawingAxis()
ImageSetAntialiasing	someVar.setAntialiasing()
ImageDrawPoint	someVar.drawPoint()
ImageDrawCubicCurve	someVar.drawCubicCurve()
ImageXORDrawingMode	someVar.xorDrawingMode()
ImageDrawText	someVar.drawText()
ImageDrawLines	someVar.drawLines()
ImageResize	someVar.resize()

Supported Spreadsheet member functions

The following Spreadsheet member functions are supported:

SpreadsheetDeleteRow	someVar.deleteRow()
SpreadsheetFormatColumn	someVar.formatColumn()
SpreadsheetShiftRows	someVar.shiftRows()
SpreadsheetCreateSheet	someVar.createSheet()
SpreadsheetReadBinary	someVar.readBinary()
SpreadsheetWrite	someVar.write()
SpreadsheetAddRow	someVar.addRow()
SpreadsheetShiftColumns	someVar.shiftColumns()
SpreadsheetGetCellFormula	someVar.getCellFormula()
SpreadsheetDeleteColumns	someVar.deleteColumns()
SpreadsheetAddFreezePane	someVar.addFreezePane()
SpreadsheetDeleteColumn	someVar.deleteColumn()
SpreadsheetSetCellComment	someVar.setCellComment()

SpreadsheetSetActiveSheetNumber	someVar.setActiveSheetNumber()
SpreadsheetSetHeader	someVar.setHeader()
SpreadsheetAddSplitPane	someVar.addSplitPane()
SpreadsheetMergeCells	someVar.mergeCells()
SpreadsheetFormatRows	someVar.formatRows()
SpreadsheetGetCellComment	someVar.getCellComment()
SpreadsheetGetCellValue	someVar.getCellValue()
SpreadsheetAddInfo	someVar.addInfo()
SpreadsheetSetCellValue	someVar.setCellValue()
SpreadsheetSetFooter	someVar.setFooter()
SpreadsheetRemoveSheet	someVar.removeSheet()
SpreadsheetSetRowHeight	someVar.setRowHeight()
SpreadsheetSetActiveSheet	someVar.setActiveSheet()
SpreadsheetFormatCellRange	someVar.formatCellRange()
SpreadsheetFormatCell	someVar.formatCell()
SpreadsheetAddRows	someVar.addRows()
SpreadsheetFormatColumns	someVar.formatColumns()
SpreadsheetAddImage	someVar.addImage()
SpreadsheetSetCellFormula	someVar.setCellFormula()
SpreadsheetAddColumn	someVar.addColumn()
SpreadsheetDeleteRows	someVar.deleteRows()
SpreadsheetSetColumnWidth	someVar.setColumnWidth()
SpreadsheetFormatRow	someVar.formatRow()
SpreadsheetInfo	someVar.info()

Supported XML member functions

The following XML member functions are supported:

XmlTransform	someVar.transform()
XmlGetNodeType	someVar.getNodeType()
XmlChildPos	someVar.childPos()
XmlElemNew	someVar.elemNew()
XmlSearch	someVar.search()

Supported Query member functions

The following Query member functions are supported:

QueryAddColumn	someVar.addColumn()
QueryGetRow	someVar.getRow()
QueryConvertForGrid	someVar.convertForGrid()
QuerySetCell	someVar.setCell()
QueryAddRow	someVar.addRow()
	someVar.getResult()

Chapter 6: Developing CFML Applications

Developing CFML Applications

In this section, you can understand about building optimized and secured application, using persistent data and locking techniques, and using the ColdFusion threads effectively. You can additionally understand the debugging and troubleshooting techniques.

Designing and Optimizing a ColdFusion Application

Application elements and how you structure an application on your server make your Adobe ColdFusion pages an effective Internet application. You use the Application.cfc and Application.cfm files and various coding methods to optimize the efficiency of your application.

Handling Errors

Adobe ColdFusion includes many tools and techniques for responding to errors that your application encounters. These tools include error handling mechanisms and error logging tools. For information on user input validation, see Introduction to Retrieving and Formatting Data and Building Dynamic Forms with cfform Tags For information on debugging, see Debugging and Troubleshooting Applications.

Using Persistent Data and Locking

Adobe ColdFusion provides several variable scopes in which data persists past the life of a single request. These are the Client, Application, Session, and Server scopes. These scopes let you save data over time and share data between pages and even applications. Use these scopes as persistent scopes. In particular, use the Client and Session scopes to maintain information about a user across multiple requests. ColdFusion lets you lock access to sections of code to ensure that ColdFusion does not attempt to run the code, or access the data that it uses, simultaneously or in an unpredictable order. This locking feature is important for ensuring the consistency of all shared data, including data in external sources in addition to data in persistent scopes. You can use persistent scopes to develop an application and use locking to ensure data consistency.

Using ColdFusion Threads

You can use threads in Adobe ColdFusion to simultaneously run multiple streams of execution in a ColdFusion page or CFC.

Securing Applications

Resource security (Adobe ColdFusion Standard) or sandbox security (Adobe ColdFusion Enterprise) restricts access to specific resources, such as tags and files. You use the ColdFusion Administrator to configure sandbox or resource security, and structure an application to take advantage of this security.User security depends on a user identity. You can implement user security in Adobe ColdFusion applications. For detailed information on using Administrator-controlled security features, see Configuring and Administering ColdFusion.

Client-side CFML (for mobile development)

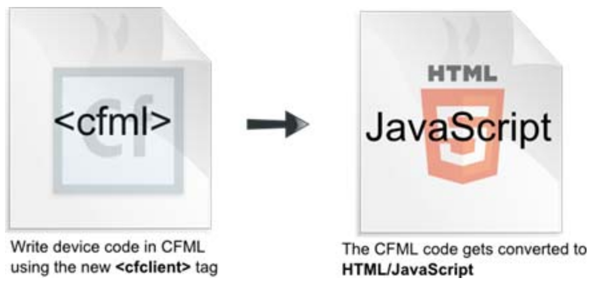
document

This document describes the client-side CFML capabilities. Client-side CFML allows the development of client-side applications using CFML. Client-side CFML can be used to develop CF-based mobile applications wherein the CFML code in the application is converted to HTML/JavaScript by the ColdFusion Server.

Note: Before you begin – To try out the examples provided in this document , you need to set up the ColdFusion mobile development environment. See Configuring the development environment .

The new <cfclient> tag

<cfclient> is a new tag introduced in ColdFusion 11 to support mobile development. This tag has been introduced to convert the CFML code that it encloses into JavaScript code. A ColdFusion developer can now develop mobile applications using CFML by leveraging the transformation functionality offered through the <cfclient> tag. So, you do not need to know JavaScript to write mobile web applications.


<cfml>
HTML
JavaScript
Write device code in CFML using the new **<cfclient>** tag
The CFML code gets converted to **HTML/JavaScript**

Even if you are an experienced JavaScript developer, <cfclient> can still be used to simplify the mobile application development as it abstracts the complexities involved in building a mobile application using JavaScript and HTML.

Note: The CFML constructs to be executed at the client-side have to be embedded within the <cfclient> tag. Not all tags, functions, and CFML functionalities are supported for conversion to HTML/JavaScript. For the complete list of CFML tags and functions that <cfclient> tag supports, see Supported CFML language constructs and Supported CFML tags.

The rationale behind choosing to support only a certain set of tags and functions is to strengthen the relevance of CFML for client-side mobile application development.

How does the transformation work

Let us see how the regular <cfoutput> tag gets rendered on a browser.

Your ColdFusion code:

```
<cfoutput>Hello World</cfoutput>
```

What the browser gets from the ColdFusion Server:

Hello World

Let us revisit the Hello World <cfclient> example mentioned in the A customary hello world example section.

Now, if your ColdFusion code is:

```
<cfclient> <cfset myvar = "Hello World"> <cfoutput>#myvar#</cfoutput> </cfclient>
```

Check the source of the web page translated by ColdFusion Server. It will be pure JavaScript wrapped in an HTML page.

As you can infer, the CFML code available in the <cfclient> block gets converted to JavaScript. Though this example is simple, the translation works the same way for complex CFML code.

Supported CFML language constructs

The following CFML language constructs are supported in client-side CFML, which includes all the logical/conditional and flow constructs:

- IF/ELSE/ELSEIF
- WHILE/DO WHILE
- CFLoop/CONTINUE/BREAK
- SWITCH/CASE/DEFAULTCASE
- TRY/CATCH/FINALLY
- FOR
- TERNARY OPERATOR
- THROW
- IMPORT
- INCLUDE
- ABORT
- EXIT
- FUNCTION/ARGUMENT/RETURN
- FUNCTION INVOCATION
- CUSTOM TAGS

Supported CFML tags

The following CFML tags are supported in client-side CFML:

- CFSET
- CFOUTPUT
- CFINCLUDE
- CFSCRIPT
- CFOBJECT
- CFINVOKE
- CFMODULE
- CFSAVECONTENT
- CFPARAM
- CFPROPERTY

- CFCOMPONENT
- CFABORT
- CFEXIT
- CFRETURN
- CFBREAK
- CFCONTINUE
- CFQUERY
- CFQUERYPARAM
- CFFLUSH

Note that member functions are also supported in client-side CFML. The <cfparam> tag does not support device APIs.

Where client-side CFML differs from the server-side CFML

Though you can have any valid CFML code in the <cfclient> code block, there are behavioral restrictions on the CFML tags and constructs. Some of the behavioral restrictions are listed here:

- The keys for implicit structure will be static. For instance, you cannot declare {"#a#":"value"}. Also, {a:a} will become {'a':a}.
- In the <cfinclude> tag, dynamic template name(##) is not supported:

```
<cfclient> <cfset x="abc.cfm"> <cfinclude template="#x#"> </cfclient>
```

  This limitation is applicable for the <cfmodule> tag too.

- Also, the <cfinclude> tag only supports files with extensions .cfm, .js, and .css.
- The Boolean behavior differs in <cfclient>. For example, In ColdFusion, 0/1, true/false, 'true'/'false', yes/no are all treated as Boolean. However, in <cfclient>, only true/false are Boolean.
- In ColdFusion, x="1" is still a number even with the quotes. However, <cfclient> treats this as a string. Ensure that you follow strict data types for the functions to avoid abnormal behavior.
- In ColdFusion, <cfset x="1a"> is a date but inside <cfclient> it is not. Note that <cfclient> follows JavaScript date format instead of ColdFusion date format.
- In ColdFusion, <cfset x = 1/2> is 0 but inside <cfclient> it is 0.5.
- ColdFusion server exceptions will not work on client side.
- The format of the Date/Time/DateTime objects created by createDate, createTime, and createDateTime respectively differs from the server side CFML behavior. For instance, the following code:

```
<cfoutput>#CreateDateTime( 1776, 7, 4, 12, 59, 0)#</ cfoutput >
```

  On server side, you will get the output: {ts '1776-07-04 12:59:00'} When you use parseDateTime on client side CFML, ensure that you pass the output obtained from the creatDateTime function as an argument to create the DateTime object.

- The following code will not work because of the strict data types:

```
<cfset date1 = #createdatetime(2001,07,04,6,30,45)#> <cfset mytimespan = #createtimespan(2, 1, 16, 30)#> <cfoutput>#date1 +
mytimespan#</cfoutput> <cfoutput>#DateFormat(date1 + mytimespan)#</cfoutput>
```

  In the above example, timespan when added to datetime, becomes a string.

- **Note:** The DateFormat function does not work with Firefox and Internet Explorer. In Chrome, the function displays output correctly. A possible workaround is to use a function like createDateTime(2003,6,11,10,50,32) and pass the date object to dateFormat function.
- Function naming convention – Functions supported by browser and PhoneGap will have server CFML syntax. For instance, FileXXX. The PhoneGap functions will follow the Object access approach. For instance, Camera.XXX.
- Scopes available on server side is not supported on client side.
- Argument Collection will not be supported for passing arguments.
- arraySort function differs in the behavior when its numeric numbers like 0002, 00001, 1.0E+5 and the sort type is text.

```
<cfscript> anumeric = arrayNew(1); anumeric[1] = 01; anumeric[2] = 001; anumeric[3] = 1; anumeric[4] = 1.001; anumeric[5] = 1.1;
anumeric[6] = 1.101; anumeric[7] = 1.109; anumeric[8] = 1.11; anumeric[9] = 2; anumeric[10] = 02; anumeric[11] = 00002; anumeric[12] =
20; anumeric[13] = 50; anumeric[14] = 1.0E+2; anumeric[15] = 100; anumeric[16] = 1000; anumeric[17] = 1.0E+5; </cfscript> <cfset
arraySort(anumeric, "text")> <cfloop array="#anumeric#" index=i> <cfoutput>#i#</cfoutput> </cfloop>
```

  Actual output for the above code: 1 1 1 1.001 1.1 1.101 1.109 1.11 100 100 1000 100000 2 2 2 20 50 Expected output for the above code: 00002 001 01 02 1 1.001 1.0E+2 1.0E+5 1.1 1.101 1.109 1.11 100 1000 2 20 50 This is because JavaScript represents 02, 002, 2 in the same way as '2' and hence differs in sort.

- Duplicate function behavior for struct differs when the struct internally has a reference to another struct more than once. On server side, changing the value in the duplicated struct's referred key will also change the values at other referred points. However, in the case of client side CFML, this does not happen.

```
<cfset str1 = {name: 'str1', value: 'org struct'}> <cfset str2 = {name: 'str2', value1: str1, value2: str1}> <cfset str2dup = duplicate(str2)>
```

  In the above example, if you change the value of str2dup.value1.value, on the server side, value of str2dup.value2.value is also changed automatically as they both refers to same structure. But on client side, this is not the behavior.

- On client side, calling a super function from an included CFM or CFC is not supported.
- Positional arguments are not supported.

When you use the <cfoutput> tag inside the <cfclient> tag, the contents of the <cfoutput> tag is not immediately processed. Hence, you may encounter certain issues while using this code:

```
<cfclient> <cfoutput> <div id="result"></div> </cfoutput> <cfset document.getElementById("result").innerHTML = "Hello"> <cfclient>
```

In this case, while the document.getElementById() statement is being invoked, cfoutput is not processed. Hence, you will not find an element with id "result", which will result in a runtime error.

In this case, you can directly write to the DOM, instead of using cfoutput:

```
<div id="myDiv"></div> <cfclient> <cfset document.getElementById("myDiv").innerHTML += "<div id=""result""></div> <cfset
document.getElementById("result")+="Hello"> </cfclient>
```

Or another workaround is to use flush explicitly after <cfoutput>:

```
<cfclient> <cfoutput> <div id="result"></div> </cfoutput> <cfflush> <cfset document.getElementById("result").innerHTML = "Hello">
<cfclient>
```

If you follow this approach, ensure that the HTML content in the cfoutput is well formed.

Client-side CFML and JavaScript

Important: The variable names and function names in CFML are case sensitive.

From **CFML**, when you are invoking **JavaScript**:

- Use the correct case for function name and variable name when referencing CFML functions and variables from CFML as client-side CFML is case sensitive.

From **JavaScript**, when you are invoking **CFML**:

- Use the correct case for function name and variable name when referencing CFML functions and variables from JavaScript as client-side CFML is case sensitive.

Loading JavaScript files

You can load content of the JavaScript files in your ColdFusion code using the loadJSFile() function as shown in the following example:

```
component client="true" { function init () { cfclient.loadJSFile("yourjsfile.js", function () { alert("Script loaded"); }); } }
```

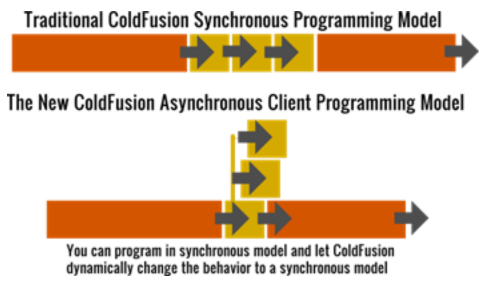
You can also use <cfinclude> to load a JavaScript file.

Synchronous and asynchronous function calls

ColdFusion automatically determines whether a function call is synchronous or asynchronous. However, if you need to invoke an asynchronous function in a synchronous mode, you can use the invokeInSyncMode function. The function call just needs to be wrapped around with the invokeInSyncMode function call. For instance, invokeInSyncMode (myAsyncFunc(arg1,arg2)). See InvokeCFClientFunction .

Asynchronous behavior

As a ColdFusion developer, you have always been using synchronous programming models. However, programming client applications using JavaScript needs to follow an asynchronous model. Fortunately, ColdFusion does most of the synchronous to asynchronous translation automatically.


Traditional ColdFusion Synchronous Programming Model
The New ColdFusion Asynchronous Client Programming Model
You can program in synchronous model and let ColdFusion dynamically change the behavior to a synchronous model

For instance, see the following script:

```
<cfscript> try { //Your code that throws an exception } catch (e) { // This statement will be reached unlike // in typical asynchronous model }
</cfscript>
```

The ability to use asynchronous functions in <cfclient> through the ‘known’ synchronous model provides a lot of convenience while building mobile applications using ColdFusion.

Since ColdFusion automatically translates synchronous code to asynchronous code, exception handling becomes easier in client code.

The behavior of certain tags has been modified to support the asynchronous behavior. In the process, functionalities of some tags may differ. For instance, <cfloop> does not support the following features when used along with <cfclient>:

- Looping over a COM collection
- Looping over a date or time range
- Looping over a directory

Support for ColdFusion Functions

You can start writing mobile applications in ColdFusion using existing data types and functions. The <cfclient> tag supports CFML data types and functions.

The following functions depict usage of data types and functions in your ColdFusion-based mobile projects.

Using CFML simple types

The following example shows the usage of simple data types:

```
<cfclient> <cfset myVar1 = 1> <cfset myVar2 = “hello”> <cfset myVar3 = true> </cfclient>
```

Using CFML structures

The following example shows the usage of simple structures:

```
<cfclient> <cfset myStruct = structNew()> <cfset myStruct.key1= “hello”> <cfif structKeyExists(myStruct, “key1”)> ... </cfif> </cfclient>
```

Using CFML arrays

The following example shows the usage of arrays:

```
<cfclient> <cfset myArray = arrayNew(1)> <cfset myArray[1] = “hello”> </cfclient>
```

Using CFML functions

The following example shows the usage of functions:

```
<cfclient> <cfif arrayLen(myArray) gt 1 > ... </cfif> <!--- using the math function---> <cfset sum = arraySum(myArray) > <!--- using the date/time
function---> <cfset currentDate = now() > <!--- using the locale function---> <cfset locale = getLocale() > </cfclient>
```

List of all supported functions

The following list shows all the **supported Array functions** in client-side CFML:

- arraySlice
- arrayAppend
- arrayIsDefined
- arrayAvg
- arrayIsEmpty
- arrayClear
- arrayLen
- arrayMax
- arrayNew

- arrayMin
- arraySort
- arrayDelete
- arrayToList
- arrayPrepend
- isArray
- arrayResize
- listToArray
- arraySet
- arrayFind
- arrayFindAll
- arraySum
- arraySwap
- arrayFindNoCase
- arrayFindAllNoCase

The following Array function is **NOT supported**:

- arrayFilter (closure function)

The following list shows all the **supported Structure functions** in client-side CFML:

- isStruct
- structDelete
- structAppend
- structInsert
- structClear
- structIsEmpty
- structCopy
- structKeyExists
- structCount
- structNew
- structFind
- structUpdate
- structFindKey
- structFindValue
- structGet
- structKeyArray
- structKeyList
- structSort

The following list shows all the **supported List functions** in client-side CFML:

- find
- findNoCase
- findOneOf
- formatBaseN
- lCase
- left
- len
- listAppend
- listChangeDelims
- listContains
- listContainsNoCase
- listDeleteAt
- listFind
- listFindNoCase
- listFirst
- listGetAt
- listInsertAt
- listLast
- listLen
- listPrepend
- listQualify
- listRest
- listSetAt
- listSort
- listToArray
- listValueCount
- listValueCountNoCase
- replaceList
- valueList

The following List function **is NOT supported**:

- getClientVariablesList

The following String functions **are NOT supported**:

- binaryEncode
- binaryDecode
- charsetEncode

- charsetDecode
- toBase64
- toBinary
- toString
- uRLDecode
- uRLEncodeFormat
- lSParseNumber
- lSParseCurrency
- lSParseEuroCurrency
- lSIsDate
- lSIsNumeric
- lSIsCurrency

The following **Regex functions are supported**:

- rEFind
- rEMatch
- rEFindNoCase
- rEMatchNoCase
- rEReplace
- rEReplaceNoCase

The following Math function **is NOT supported**:

- precisionEvaluate

The following Date functions **are NOT supported**:

- createODBCDate
- createODBCTime
- createODBCDateTime
- lSDateFormat
- lSIsDate
- lSParseDateTime
- lSTimeFormat

The following **utility functions are supported**:

- isBoolean
- isDefined
- decimalFormat
- isNumeric
- dollarFormat
- isNull

- htmlCodeFormat
- isSimpleValue
- htmlEditFormat
- isValid
- numberFormat
- createUUID

Support for custom tags

You have been using custom tags in ColdFusion for the past few releases of ColdFusion. Custom tags allowed you to extend CFML by adding your own tags to the ones shipped with ColdFusion. Custom tags can now be created in <cfclient> too. The following sections provide an overview of the supported features and restrictions while using custom tags for building mobile applications.

Note: Application and Server mappings are also supported in custom tags.

Paths for custom tags

Custom tags are detected when they are made available in the following locations:

- The custom tag available in the same directory as the calling page
- The custom tag available in the cfusion/CustomTags directory
- The custom tag available in sub-directories under the cfusion/CustomTags directory
- The custom tag available in server/application mapped folders

Invoking custom tags

The custom tags can be invoked in the following ways:

- Using the cf_<tagname>. For instance, by calling <cf_mytag>
- Using the <cfmodule> tag. For instance, <cfmodule template="../cf_mytag.cfm">
  - Also, <cfmodule name=”tags.mytag”>
  - For the <cfimport> tag, we use the taglib to import the custom tags:
    - For instance, <cfimport prefix = “myTags” taglib = “/custom”>
    - The DOT(.) notation can be used to access custom tags available inside sub directories. For instance, use <cfmodule name = “tags.mytag”>

<cfimport> supports only path to custom tags and hence you cannot have JSP tag libraries.

Passing values

You can pass values to a custom tag using a name-value pair:

```
<cf_mytag myname=#myvalue#>
```

Also, multiple name-value pairs can be passed to a custom tag:

```
<cf_mytag myname1=#myvalue1# myname2=#myvalue2#>
```

To access the arguments passed to the custom tag, the custom tag CFM file can use the attributes scope as follows:

```
#attributes.myname1# and #attributes.myname2#
```

To send the data back to the calling page, the custom tag CFM file can use the Caller scope as follows:

```
<cfset caller.myname=#attributes.myname1# & " " & #attributes.myname2#>
```

You can also pass a struct to the custom tag:

```
<cfset args=structNew()> <cfset args.x = "-X-"> <cfset args.y = "-Y-"> <cf_mytag arg1="value1" attributeCollection=#args# anotherarg="16">
```

Tag instance data

When a custom tag page executes, ColdFusion keeps data related to the tag instance in the **thisTag** structure. You can access the thisTag structure from within your custom tag to control processing of the tag.

To determine if an end tag is specified, use the hasEndTag as follows:

```
<cfif thisTag.hasEndTag is 'false'> <!--- Abort the tag---> <cfabort /> </cfif>
```

To determine the tag execution mode, use the executionMode attribute. Three modes are supported:

- Start mode – For processing the start tag
- End mode – For processing the end tag
- Inactive mode – For processing custom tags using nested tags

```
<cfif thisTag.executionMode is 'start'> <!--- Process start tag ---> <cfelseif thisTag.executionMode is 'end'> <!--- Process end tag ---> </cfif>
```

You can access the body text within the custom tag using the thisTag.generatedContent variable. You can modify this text during processing of the tag. The contents of the thisTag.generatedContent variables are returned to the browser as part of the tag's output. The content includes all text and HTML code in the body, the results of evaluating ColdFusion variables, expressions, and functions, and the results generated by descendant tags.

See the following example:

```
<cfif thisTag.executionMode is 'end'> <cfset thisTag.generatedContent ='<!--#thisTag.generatedContent#-->'> </cfif>
```

The nested sub tag can pass its attributes to the parent tag. A sub tag can use cfassociate to communicate its attributes to the base/ancestor tag.

```
<cfassociate baseTag="tagName" dataCollection="collectionName">
```

The following code shows how you can access the subtag attributes in the base tag:

```
<cfparam Name='thisTag.assocAttribs' default=#arrayNew(1)#>
```

You can also access the ancestral data in the sub tag using the getBaseTagList() helper method as follows:

```
<cfset ancestorlist = getBaseTagList()>
```

The getBaseTagList() method returns a comma-delimited list of uppercase ancestor tag names, as a string. You can also use the getBaseTagData() method to return an object that contains all the variables of the nth ancestor.

Aborting custom tag processing

The <cfexit>/<cfabort> tag exits the page execution.

Deviation list for custom tags

The following list contains some known issues and deviations in behavior of the custom tags:

- In <cfclient>, variables scope is supported. But you have to explicitly scope it.

Server-side CFML	**Client-side CFML**
caller.cfm **customtag.cfm**	**caller.cfm** **customtag.cfm**

- If you use the "#attributes.attributename#" syntax in the custom tag after an asynchronous call, you will see an unexpected behavior.

Server-side CFML	**Client-side CFML**
	The above code will not work. Use:

- Numeric values passed to the attributes in caller are passed as a string to the custom tags:

```
<cf_custom attr1="1">
```

  In the above example, attr1 is converted to a number, if you are accessing the attribute in a numeric operation. However, it does not work in this manner for client-side custom tags. You need to use:

```
<cf_custom attr1=1>
```

  Or:

```
<cfset x = 1> <cf_custom attr1=#x#>
```

  Type conversion is not handled in client custom tags.

- Function declared in the caller CFM is accessible in the custom tag (CFM) using the caller.functionname() on the server-side. However, this is not the behavior on the client side.

Server-side CFML	Client-side CFML
	The functions defined in the caller CFM are not available in the custom tags.

- Using variables to pass the path of the included file does not work inside <cfclient>.

Server-side CFML	Client-side CFML
	This is not supported.

- Passing the template/name (with <cfmodule>) attribute as a variable does not work with <cfclient>.

Server-side CFML	Client-side CFML
	This is not supported as we need to do the translation during the compile time itself.

- Exception thrown in the custom tag template will not be handled by the exception handler defined in the caller CFM. If the custom tag name is wrong or if the included CFM name is wrong, in client-side CFML, you will get an exception during the compilation time itself.

Note: Based on the location of the JavaScript file (specified in the <cfinclude> tag or using the <script> tag), the order of execution of statements differ.

Non-<cfclient> custom tags cannot be called from caller CFMs of <cfclient>. Also, a client-side custom tag cannot have server-side ColdFusion tags outside the <cfclient> tag. This is true for client-side included-CFMs too. For better debugging, do not add script blocks/link tags/style tags in the client-side custom tags. Always create a separate JavaScript file or a CSS file and add them using the <cfinclude> tag.

```
<cfinclude template="utils.js"> or <cfinclude template="new.css">
```

This is applicable for client-side included CFMs too.

Support for CFC (Client-side and Server-side)

A client-side CFC can be written using the client=true attribute for a cfclient component. For instance, a client-side CFC can identify itself by having client=true along with other component attributes. See the following example:

```
component client=true { public function foo() { //some code here } }
```

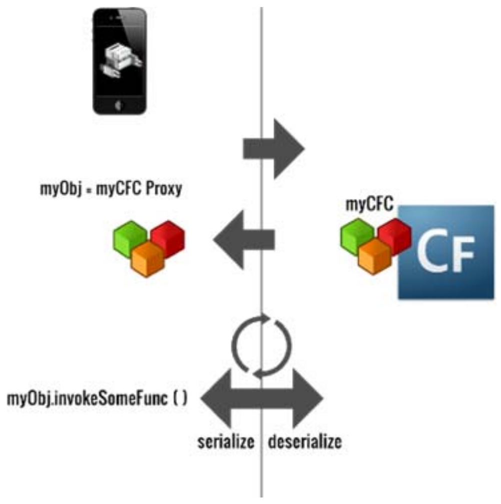
<cfclient> communicates with the ColdFusion Server quite seamlessly so much so that you can create objects of a server component and call functions on that object just like how you do ColdFusion Server programming. The ColdFusion Server facilitates all the various interactions required between a mobile client and the server without any restrictions on the language.

See the following example:

```
<cfclient> <!-- Create a JS proxy of server side component myCFC.cfc---> <cfset proxy = new app1.controls.myCFC(id)> <!-- Update some data
---> <cfset proxy.setVar("myVar1")> <cfset proxy.setProperties(someStructVar)> </cfclient>
```

In the above example, you are calling a function on a remote CFC within the <cfclient> tag. The new operator creates a proxy to the server CFC. You can use the instantiated CFC object to invoke remote functions on that CFC.

Note that a server CFC is similar to any other CFC without the client attribute being set to true.



The <cfclient> tag allows the usage of CFCs just like any other CFML constructs. There are multiple ways of using CFCs in the <cfclient> block.

The following example shows a very simple usage:

```
<cfclient> <cfset obj = new mycfc()> <cfset obj1 = createObject("component","mycfc")> </cfclient>
```

In the above example, mycfc.cfc can be a client-side CFC or a server-side CFC. As you can infer, CFCs can be created using createObject, new keyword, and cfinvoke techniques.

You can also use a CFC that extends functionalities from another CFC:

```
<cfclient> <cfset obj = new mycfc()> <!--- mycfc extends mycfc1.cfc present in the same directory ---> <cfset result = obj.getResult() > <!---
getResult() function present in mycfc1.cfc and can be accessed from the extending classes ---> </cfclient>
```

Note: Ensure that if mycfc.cfc is a client-side CFC, then mycfc1.CFC should also be a client-side CFC. This is applicable even for the server-side CFC if mycfc.cfc is a server-side CFC.

You can also use <cfimport> for importing mapped paths:

```
<cfimport path="com.*" /> <cfset obj = new com.mycfc() /> <!--- mycfc present in directory mapped to com ---> You can also use functions
within a CFC: <cfclient> <cfset obj = new mycfc() > <cfset obj.foo() > <!--- invoke function on cfc instance ---> </cfclient>
```

Note: Ensure that the function foo() is a remote function, if mycfc.cfc is a server-side CFC.

Support for database queries

You can start using the <cfquery> tag in client-side CFML just like how you are currently using it in server-side CFML code. Note that not all of the <cfquery> features are supported in this release. The support for database queries in client-side CFML is based on Web Database (Web SQL). So, this feature may not work on certain browsers. To check if your browser supports Web SQL, see this web page.

What is supported

The following list shows the extent of <cfquery> support available in client-side CFML:

- The <cfquery> tag supports ONLY the following attributes:
  - name = "query name"
  - dataSource = "data source name"
  - result = "resultVar"

The result variable will contain sql, recordCount, columnList, and sqlparameters.

- You can use the <cfloop> tag to iterate over the query. The <cfloop> tag will support query, startrow, and endrow attributes.
- You can use the <cfqueryparams> tag for parameterized query.
- In the <cfqueryparam> tag, only the value attribute is supported.
- The queryparam attribute value can be provided through position parameters in an array using the queryExecute function:

queryexecute("sql", queryparams, queryoptionsmap)

- Note that serialization and deserialization of queries from client to server and server to client will be seamless.
- You can use the isQuery function in client-side CFML to check if a variable is of query type.

What is NOT supported

The following features are not supported:

- Performing query of queries.
- In-memory query creation functions like QueryNew and QueryAddRow.

Usage example

The following example shows the basic usage of the <cfquery> tag in client-side CFML:

```
<div id="actual_1" class="async_actual"> <cfclient> <cfquery datasource="cfds" >drop table if exists birthDates</cfquery> <cfquery
datasource="cfds" > CREATE TABLE if not exists birthDates( serialNo INTEGER PRIMARY KEY AUTOINCREMENT, firstname VARCHAR(20),
lastname VARCHAR(20), dob TEXT) </cfquery> <!---Insert string. ---> <cfquery datasource="cfds" name="q1"> INSERT INTO
birthDates(firstName, lastname,dob) VALUES('Jon', 'Doe', 'Mark') </cfquery> <cfset d1=createDate(1975, 12, 25)> <cfset
a=dateFormat(d1,"yyyy-mm-dd")> <cfquery datasource="cfds" name="q2"> INSERT INTO birthDates(firstName, lastname, dob) VALUES('Jon',
'Doe', '#a#') </cfquery> <cfset d2=createDate(1980, 01, 01)> <cfset b=dateFormat(d2, "yyyy-mm-dd")> <cfquery datasource="cfds"
name="q2"> INSERT INTO birthDates(firstName, lastname, dob) VALUES('Jon', 'Doe','#b#') </cfquery> <cfset d3=createDate(1985, 12, 27)>
<cfset c=dateFormat(d3, "yyyy-mm-dd")> <cfquery datasource="cfds" name="q3"> INSERT INTO birthDates(firstName, lastname, dob)
VALUES('Jon', 'Doe','#c#') </cfquery> <cfset startRow="2"> <cfset endRow="4"> <cfquery datasource="cfds" name="q4" result="test"> SELECT
* FROM birthDates where serialNo between <cfqueryparam value="#startRow#"> and <cfqueryparam value="#endRow#"> </cfquery> <cfset
write_to_div("actual_1", test.sql & "<br>" & test.recordCount & "<br>" & test.columnList)> <cfloop query="q4"> <cfset write_to_div("actual_1",
firstname & " " & lastname & ":" & dob &"<br>")> </cfloop> </cfclient> <script type="text/javascript"> function
write_to_div(div_id,data_to_write) { document.getElementById(div_id).innerHTML+=data_to_write; } </script> </div>
```

See the server-side <cfquery> support .

Using the ColdFusion Debugger

Adobe ColdFusion provides debugging information for individual pages. However, for complex development tasks, you require a robust and interactive debugger. ColdFusion provides a line debugger that you can use when developing ColdFusion applications in Eclipse or Adobe Flash Builder. You can set breakpoints, step over, into, or out of code, and inspect variables. You can also view ColdFusion log files.

Debugging and Troubleshooting Applications

Adobe ColdFusion provides detailed debugging information to help you resolve problems with your application. You configure ColdFusion to provide debugging information, and use the cftrace and cftimer tags to provide detailed information on code execution. You can also use tools for validating your code before you run it and troubleshoot particular problems.

Note: Adobe Dreamweaver provides integrated tools for displaying and using ColdFusion debugging output. For information on using these tools, see the Dreamweaver online Help.

Developing Globalized Applications

Resource security (Adobe ColdFusion Standard) or sandbox security (Adobe ColdFusion Enterprise) restricts access to specific resources, such as tags and files. You use the ColdFusion Administrator to configure sandbox or resource security, and structure an application to take advantage of this security.User security depends on a user identity. You can implement user security in Adobe ColdFusion applications. For detailed information on using Administrator-controlled security features, see Configuring and Administering ColdFusion.

REST Enhancements in ColdFusion 11

ColdFusion 11 now supports site-level REST applications and enables pluggable serializer and deserializer.

Site-level REST application support

See Site-level REST application support

Support for pluggable serializer and deserializer

See Support for pluggable serializer and deserializer

Authentication through OAuth

The <cfoauth> tag allows you to easily integrate third-party OAuth 2 authentication providers in your application. This tag currently supports Facebook and Google authentication. Also, this tag supports OAuth providers that support the OAuth 2 protocols. For instance, Microsoft and Github.

See cfoauth

Social Enhancements

ColdFusion has introduced the support for dynamically generating Like button, Tweet button, and Comment box for social media sites. The supported widgets are:

- Like button
- Tweet button
- Facebook comment box
- Google Plus button
- Facebook subscribe button
- Like box
- Activity feed
- Follow

Examples

1. Syntax for Facebook Like button:

```
<cf_socialplugin type = "like" url = "" layout = "standard|box_count|button_count" showfaces = "true|false" verb = "like|recommend"
colorscheme = "light|dark" style = "" width = "" extraoptions = "" >
```

2. Syntax for Facebook Likebox button:

```
<cf_socialplugin type = "likebox" url = "This refers to a Facebook Page." showfaces = "true|false" showstream = "true|false" showheader =
"true|false" colorscheme = "light|dark" style = "" height = "" width = "" extraoptions = "" >
```

3. Syntax for Activity feed for a Facebook application:

```
<cf_socialplugin type = "activityfeed" appid= "facebook_app_id" width = "" height = "" colorscheme = "light|dark" showheader = "true|false"
action = "" linktarget = "_blank|_top|_parent" recommendations = "true|false" style = "" extraoptions = "" >
```

4. Syntax for a Twitter Follow button:

```
<cf_socialplugin type = "follow" showcount= "true|false" buttonsize= "medium|large" language = "en|fr|.........." showusername = "true|false"
username = "" style = "" extraoptions = "" >
```

5. Syntax for a Google Plus button:

```
<cf_socialplugin type = "plusone" url = "url to plus one" buttonsize= "small|medium|large|tall" language = "en|fr|.........." width = "" annotation
= "none|inline|bubble" style = "" extraoptions = "" >
```

6. Syntax for a Facebook Comment box:

```
<cf_socialplugin type = "commentbox" url = "url" width= "" colorscheme = "dark|light" numberofposts = "" style = "" extraoptions = "" >
```

7. Syntax for a Facebook Subscribe button:

```
<cf_socialplugin type = "subscribe" url = "profile to subscribe" width = "" colorscheme = "dark|light" showfaces = "true|false" layout =
"standard|button_count|box_count" style = "" extraoptions = "" >
```

8. Syntax for Tweet button:

```
<cf_socialplugin type = "tweet" url = "url to share" tweettext = "default tweet text" language = "en|fr|.........." count = "none|horizontal" hashtag
= "Comma separated hash tags appended to tweet text. Do not include the #. It is preprended to each list item automatically." buttonsize =
"small|large" via = "" recommend = "" style = "" extraoptions = "" >
```

Chapter 7: Accessing and Using Data

Accessing and Using Data

Effective Database management and using the querying facility to search for data is explained in this section.

Introduction to Databases and SQL

Adobe ColdFusion lets you create dynamic applications to access and modify data stored in a database. You do not require a thorough knowledge of databases to develop ColdFusion applications, but you need to know some basic database and SQL concepts and techniques. Each database server (such as SQL Server, Oracle, or DB2) has unique capabilities and properties. For more information, see the documentation that ships with your database server.

Accessing and Retrieving Data

Several ColdFusion tags provide a way to retrieve data from a database and work with query data. Use the cfquery tag to query a data source, the cfoutput tag to output the query results to a web page, and the cfqueryparam tag to help reduce security risks in your applications.

Updating Your Database

Adobe ColdFusion lets you insert, update, and delete information in a database.

Using Query of Queries

A query that retrieves data from a recordset is called a Query of Queries. After you generate a recordset, you can interact with its results as if they are database tables by using Query of Queries.

Managing LDAP Directories

CFML applications use the cfldap tag to access and manage LDAP (Lightweight Directory Access Protocol) directories. The following information teaches you to query and update an LDAP database. It is not assumed that you are familiar with LDAP, and hence an introduction to LDAP directories and the LDAP protocol is provided. However, it is assumed that you have information on the structure and attributes of your LDAP database. Hence, procedures to create an LDAP directory or manage a directory server are not provided. To learn more about LDAP and LDAP servers, see your LDAP server documentation and published books on LDAP. The examples here use the Airius sample LDAP database that is supplied with the Netscape and iPlanet Directory Servers.

Solr search support

The Solr search service is an open source enterprise search server based on the Lucene Java search library.

Configuring Solr memory

Solr enhancements in ColdFusion 9.0.1

Solr enhancements in ColdFusion 10

Chapter 8: ColdFusion ORM

ColdFusion ORM

Relational databases are the core of most enterprise applications. However, when you map a relational database to objects, it becomes a challenge. Object relational mapping (ORM) is a programming framework that allows you to define a mapping between an application object model and the relational database. In an object model, the application objects are not aware of the database structure. Objects have properties and references to other objects. Databases consist of tables with columns that may be related to other tables. ORM provides a bridge between the relational database and the object model. By using ORM, you can access and update data entirely using the object model of an application. ORM provides features such as:

- Database vendor independence
- Caching
- Concurrency
- Performance optimization

Introducing ColdFusion ORM

In previous ColdFusion releases, database access was achieved by:

- Managing relational data using tags such as cfquery, cfinsert, and cfupdate, which handle SQL statements, or via stored procedures.
- Managing objects using ColdFusion components (CFCs), and object lifecycle using the application itself

- Writing SQL queries for each CFC, even for basic CRUD (Create, Retrieve, Update, and Delete) operations. The complexity of managing these tasks increases as your application grows. ColdFusion ORM automates most of these tasks, which:
- Makes application code cleaner and more manageable
- Enhances your productivity and lets you develop database applications faster
- Creates applications that can run faster because of built-in ORM optimizations
- Minimizes the amount of code you write

Apart from providing a framework for mapping the object model with the relational database, ColdFusion ORM provides data query and retrieval facilities. For more information, see www.hibernate.org.

Architecture

In ColdFusion ORM, you need to define an object mapping to create persistent objects. The object mapping includes details such as:

- The table name for the object's class
- The column name that corresponds to each field in the object
- The join conditions for related objects

ColdFusion allows you to specify the mapping in CFCs. Such CFCs are called as persistent CFCs. Each persistent CFC usually maps to a table in the database. Each property in the CFC usually maps to a column in the table. Additional properties may be used to define relationships and other mapping details. When ColdFusion creates the Hibernate configuration for the application, these persistent CFCs are used to automatically generate Hibernate mapping files, which have the extension ".hbmxml". For example, if ARTISTS.cfc is a persistent CFC, ColdFusion would automatically generate Artists.hbmxml. Hibernate mapping files contain the mapping information in XML format that Hibernate defines, to work with ColdFusion ORM. These Hibernate mapping files can be created manually. For more information about creating Hibernate mappings manually, see Advanced mapping. To use ColdFusion ORM, ColdFusion application must have ormenabled set to true in the THIS scope of Application.cfc. To define a persistent CFC, set persistent="true" in cfcomponent tag. An array of attributes are available in cfcomponent and cfproperty to specify mapping information. For details, see Define ORM mapping. When the application starts, ColdFusion first loads the Hibernate configuration file if it is specified in the application. The Hibernate configuration file contains various configuration parameters like including, dialect, cache settings, and mapping files that are required for the application. If a configuration file is not specified, ColdFusion ORM creates the Hibernate configuration using the default settings. After loading the Hibernate configuration, all the mapping files (*.hbmxml) in the application folder and its mapped folders are loaded and added to the configuration. ColdFusion then searches for persistent CFCs in the application folder and its mapped folders. If the hibernate mapping file is not present for any persistent CFC, ColdFusion generates it. If mapping information, such as primary key, foreign key, and column data type is missing in the persistent CFCs, ColdFusion automatically inspects the database and identifies the mapping. ColdFusion then checks if DDL needs to be generated. This can be configured using the dbcreate option in the ORM settings. Based on the configuration option specified in dbcreate, tables are created or updated.The Hibernate SessionFactory is then built and made available to the application as long as the application is running. The SessionFactory is used to create Hibernate sessions that manage the persistent object lifecycle. In ColdFusion, a Hibernate session starts when the first CRUD method is called and ends when the request ends or when the ORMCloseSession() method is called. To improve performance, Hibernate batches all the Create/Update/Delete operations in the session and runs them when the session is flushed or only when necessary. Session Flush happens when the request ends or when the ORMFlush() method is called. For transactions, a new session is always created at the start of a transaction and ends at the end of a transaction. Any previous open sessions are flushed and closed at the start of the transaction. The Hibernate configuration is created and loaded only when the application starts. Therefore, any modifications to the mapping in the persistent CFCs or in the Hibernate mapping files are not loaded automatically. To load these modifications, you can either restart the application or call ORMReload(). To restart the application, you can stop the application using ApplicationStop() and the next request to any page in this application automatically starts it.

Configure ORM

The configuration for ORM is done in Application.cfc which makes this configuration application specific. For a ColdFusion application to use ORM, the following are the mandatory settings that need to be configured:

1 Enable ORM for the application. To do this, set the ormenabled property to true in the THIS scope of application.cfc

2 Provide the data source name by either setting data source property to true in the THIS scope of application or by defining it in ORM configuration for the application. Note that the data source should be configured on the server. The ORM configuration is specified using a struct called ormsettings, which is defined in the THIS scope of Application.cfc. The following table describes the settings for ORM that can be defined in Application.cfc.

Property Name	Description
ormenabled	Specifies whether ORM should be used for the ColdFusion application.Set the value to true to use ORM. The default is false.
datasource	Defines the data source that should be used by ORM.
ormsettings	The struct that defines all the ORM settings. For details, see ORM settings

Define ORM mapping

The ORM mapping can be defined either in the CFC or in a separate Hibernate mapping file (.hbmxml). See Advanced mapping for details on Hibernate mapping file. The ORM mapping is defined in the CFC using ORM-specific attributes on cfcomponent and cfproperty tag. Following example shows a CFC (ARTIST.cfc) with mapping information:

```
<cfcomponent persistent="true" entityname="Artist" table="Artists"> <cfproperty name="id" column="ARTISTID" generator="increment">
<cfproperty name="firstname"> <cfproperty name="lastname"> <cfproperty name="address"> <cfproperty name="city"> <cfproperty
name="state"> <cfproperty name="postalcode"> <cfproperty name="email"> <cfproperty name="phone"> <cfproperty name="fax">
<cfproperty name="thepassword"> </cfcomponent>
```

Working with objects

Operations can be performed on an Entity object, and the auto-generated methods in the entity can be called.

ORM session management

Hibernate Session is a thread-safe and short-lived object that represents a conversation between the application and the persistence layer. This is required to perform CRUD operations (Create, Delete, Update, Retrieval) on a persistent entity in Hibernate. For ColdFusion applications that use ORM, this session is automatically managed by ColdFusion. Hibernate sessions also act as the first level of cache, which ensures that only one copy of an object exists in the session. When CRUD operations are performed in the session, data of the entity is not synchronized with the database immediately. That is, the SQL statements for the operations are not issued immediately and they are queued. The data is synchronized with the database when the session is flushed. When the session is flushed, the SQL operations are performed in the following order:

1. all entity insertions, in the same order the corresponding objects were saved using EntitySave()
2. all entity updates
3. all collection deletions
4. all collection element deletions, updates, and insertions
5. all collection insertions
6. all entity deletions, in the same order the corresponding objects were deleted using EntityDelete() The only exception to this is that objects with nativeId generation are inserted immediately when the object is saved.

Note: ColdFusion creates and manages Hibernate sessions only if ormenabled is set to true in application scope.

When the ColdFusion application starts, it builds the Hibernate session factory that is available for the application life time. This factory is used to create Hibernate sessions that manage the persistent object lifecycle. When the first CRUD method is called, a Hibernate session gets created and is closed when the request ends or when the ormclosesessionmethod is called. For details on how ColdFusion works with Hibernate, see the Architecture. In multiple data source scenarios supported in ColdFusion 9 Update 1, there are multiple sessions (one for each data source) in the same request. For all entity functions, the appropriate sessions are used transparently. ColdFusion exposes a few methods to let CFML developers work with the Hibernate sessions directly. ORM session-related functions also take optional data source argument. If you do not specify a data source, the default data source specified for ORM is used. The methods are as follows:

Transaction and concurrency

When ORM methods are invoked without any transaction, all the data is committed to the database when the ORM session is flushed. ORM session is flushed whenORMFlush()is called or if autoflush is enabled when the request ends.

This works fine when there is not much concurrency, however in most practical scenarios you would need to use transaction in your application so that the data in your database is always in a consistent state.

With ColdFusion ORM, you can manage transactions in the following two ways:

- **Using Hibernate transaction:** User has full control and ColdFusion does not intervene. The application has to flush/close the session and commit/rollback the transaction.

  For more information on transactions, go to the following URL:

  http://community.jboss.org/wiki/sessionsandtransactions

- **Using CFTransaction:** ColdFusion manages the transaction. Since a transaction cannot be distributed (across different data sources), application must ensure that the changes made in the transaction affect only one Hibernate session. That is, only one data source.

  ColdFusion allows reading of data from other sessions (data source) in a transaction but changes must be made in only one session. Multiple dirty sessions at any time in the transaction can result in exceptions and the transaction is rolled back. Before transaction begins, all existing sessions in the request are flushed. The previous session (if any) is reused.

  When the transaction is committed, the dirty session is automatically flushed (before committing the transaction). When the transaction is rolled back, the changed session cannot be used any longer because it can cause rolled back data to get committed later. Therefore, the session participating in the transaction is cleared when transaction is rolled back.

A description of transaction is beyond the scope of this document. For more information on transactions, see the hibernate documentation.

To run the ORM methods inside a transaction, they must be inside<cftransaction>. A simple example snippet of using ORM with<cftransaction>is as follows:

```
<cftransaction> <cfset acct1 = EntityLoad("Account", "101")> <cfset acct2 = EntityLoad("Account", "102")> <cfset acct1.debit(1000)> <cfset
acct2.credit(1000)> <cfset EntitySave(acct1)> <cfset EntitySave(acct2)> </cftransaction>
```

Because we have not called commit on the<cftransaction>specifically, it is automatically committed when the<cftransaction>ends.

All<cftransaction>semantics including savepoint, multiple rollbacks, multiple commits, and nested transactions work with ORM. You can also have both queries and ORM in the same<cftransaction>.

When<cftransaction>begins, any existing ORM session is flushed and closed, and a new ORM session is created. The<cftransaction>can be committed or rolled back using appropriate ColdFusion tags in<cftransaction>. When the transaction ends and has not been committed or rolled back explicitly, it is automatically committed and the ORM session is closed. If there is any error inside the transaction, without any exception handling, the transaction is rolled back.

For more details on<cftransaction>, see the CFML Reference Guide.

Note: Even if ORMFlush() is called explicitly inside a <cftransaction> tag, the SQL runs but the data is committed only when the transaction commits.

Performance optimization

Using queries

ColdFusion lets you use HQL (Hibernate Query Language) to run queries directly on the database. If you are familiar with HQL, you can use it for running complex queries. In general, use HQL in the following scenarios:

- The query is not specific to a particular object but only to some fields in the object.
- To retrieve some fields of the object without loading the object.
- When you use table joins.
- When you use aggregate functions like min, max, avg, and count.
- To retrieve entities by specifying a filter that needs to use operators other than AND. The HQL methods return a single or multi-dimensional array of values or entities, based on what the HQL query returns. If you are sure that only one record exists that matches this filter criteria, specify unique=trueso that a single entity is returned instead of an array. You can use unique=true to suppress the duplicate records from the query result.

Note: entityname and properties used in HQL are case sensitive.

The following HQL methods are available: ORMExecuteQuery(hql, [params] [,unique]) ORMExecuteQuery(hql, [,unique] [, queryoptions]) ORMExecuteQuery(hql, params [,unique] [,queryOptions]) ORMExecuteQuery (hql, params, boolean unique, Map queryOptions)

Autogenerating database schema

ColdFusion automatically creates tables when ORM is initialized for the application. For auto-generating tables, do the following: In the THIS scope of Application.cfc, in ormsettings struct, set the dbCreate property to one of the following values:

- update: Creates the table (if it does not exist) or updates the table (if it exists).
- dropcreate: Drops the table if it exists and then creates it. For example,

```
<cfset this.ormsettings.dbCreate="update">
```

Certain specific attributes (DDL-only attributes) defined for the tags cfcomponent and cfproperty can be use to define various attributes for the auto-generated tables and columns. DDL-only attributes are used only for DDL generation. For details of these attributes, see the table in the section DDL-only attributes in Column.

Support for multiple data sources for ORM

Note: This feature applies only if you have installed ColdFusion 9 Update 1.

ColdFusion ORM search

Enhancements in ColdFusion 10 provide indexing and search capabilities for ColdFusion ORM. When you develop an application that uses ColdFusion ORM, the search feature facilitates full text search. You can load all the persistent entities that match your search criteria based on a query text. Full text search is a two-step process that involves indexing the persistent entity and search based on the indexed information.

Chapter 9: ColdFusion and HTML5

ColdFusion and HTML 5

ColdFusion provides a powerful set of HTML 5 features. Since being the future standard of interaction and communication on the Web, ColdFusion's HTML 5 features make your web experience richer and easier than before.

Using ColdFusion WebSocket

Media Player enhancements-Developing guide

The enhancements in this release support:

- Play back capability for HTML 5 videos
- Fallback to HTML 5 video playback if Flash player is not installed
- Browser independent video controls

- Dynamic streaming of Flash videos
- Advanced skinning for media player
- Play list for Flash videos
- Extending media player using plug-ins built using Open Source Media Framework (OSMF), for example to:
- Play videos in the YouTube server
- Use stage video support by showing advertisements within the videos in linear and non-linear mode
- Adding title to the video For details, see Media Player enhancements.

Client-side charting-Developing guide

ColdFusion 10 supports client-side charting. This is in addition to the existing server-side charting feature (which continues to serve the way it used to). Client-side charting supports the following:

- Dynamic and interactive charting: Modify the charts, add styles, and add new series or plots
- Popular chart formats with appropriate fallback functionality: Use HTML 5, Flash, SVG, or VML charts.
- If your browser does not support HTML 5 features relevant to charting, charts are rendered in Flash. Similarly, if
- Flash is not supported, charts are rendered in HTML.
- Features identical to server-side charting: Most of the server-side charting features are available with client-side charting.
- Old and new charts: In addition to the contemporary chart types, offers a new set of charts.
- Needs minimal trips to server: As compared to generating charts at server-level, for every user interaction. For details, see Client-side charting.

Displaying geolocation - Developing guide

Displays user location on the map if the attribute showUser is specified in cfmap. This feature works only on HTML 5 compliant browsers. For details, see Displaying geolocation.

Chapter 10: Flex and AIR Integration in ColdFusion

Flex and AIR Integration in ColdFusion

Using the Flash Remoting service of Adobe ColdFusion, ColdFusion developers can work together with Flash designers to build dynamic Flash user interfaces for ColdFusion applications. For a complete description of Flash Remoting capabilities, including how ColdFusion interacts with Flash Remoting, see Using Flash Remoting MX 2004 and Flash Remoting ActionScript Dictionary in Flash Help. You can also access the Flash Remoting documentation on the Flash Remoting Developer Center at www.adobe.com/go/learn_cfu_flashremoting_en.

Using the Flash Remoting Service

Using the Flash Remoting service of Adobe ColdFusion, ColdFusion developers can work together with Flash designers to build dynamic Flash user interfaces for ColdFusion applications. For a complete description of Flash Remoting capabilities, including how ColdFusion interacts with Flash Remoting, see Using Flash Remoting MX 2004 and Flash Remoting ActionScript Dictionary in Flash Help. You can also access the Flash Remoting documentation on the Flash Remoting Developer Center at www.adobe.com/go/learn_cfu_flashremoting_en.

Using Flash Remoting Update

You can use Flash Remoting Update to create Rich Internet Applications by using Adobe ColdFusion with Adobe Flash Builder or earlier versions of Flex Builder, with the advanced data retrieval features of ColdFusion, such as the cfpop, cfldap, and cfquery tags. In addition, you can use Flash Remoting Update to create Flash Forms and SWF files that contain features, such as server call backs and customized user interface.

Offline AIR Application Support

ColdFusion provides offline AIR application support, which includes data persistence and synchronization. These features let an AIR application use a local SQLite database that represents data on the ColdFusion server. You cannot use these features in applications built with Flash, which run in a browser or Flash Player. These features only support AIR applications with intermittent connectivity to the ColdFusion data provider. They enable users to run the AIR application offline and then synchronize data with the ColdFusion application the next time the application runs online. To support offline AIR data access, you code ActionScript elements on the client side and CFML on the server side.

Note: Some of the code in the following discussion uses an AIR application that displays and updated an Employee database that ColdFusion manages for its sample code. However, the snippets below are not all from this example, and do not make up a complete or consistent application.

Proxy ActionScript Classes for ColdFusion Services

Flex-based applications in AIR and Flash can access several ColdFusion services by using ColdFusion proxy ActionScript classes. This feature is available in all Flex-based applications that run on Flash and AIR. ColdFusion provides services corresponding to the following tags and their child subtags: cfchart, cfdocument, cfimage, cfmail, cfpdf, cfpop. Using ColdFusion you can also upload files from the application to the server. ColdFusion provides the following Flex proxy classes and related support classes:

- Config (configures the application for using ColdFusion services)
- Util (includes file upload support)
- Chart (cfchart)
- Document (cfdocument)
- Image (cfimage)
- Mail (cfmail)
- PDF (cfpdf)
- Pop (cfpop) These classes are part of coldfusion.service.mxml package, distributed in the cfservices.swc file. You normally use these classes in MXML tag format, using the cf namespace identifier, as in the following line:

```
{{<cf:Image id="image" action="AddBorder" source="Uploaded Image server URL" thickness="5" color="Blue"/>}}
```

To use a ColdFusion service in an application built with Flex, you use the Config class to establish the connection, and then use the other classes to access the ColdFusion services. Since ColdFusion 9, you can also specify the remoting destination in the Config class as well as all the proxy tags.

Note: To use ColdFusion services from Flex and AIR, you must enable access to the services as described in "Enable ColdFusion Services" in the ColdFusion Web Services section.

Using the LiveCycle Data Services ES Assembler

To use Adobe ColdFusion as the back-end data manager for an Adobe Flex application, you use the Adobe LiveCycle Data Services ES assembler. You configure the LiveCycle Data Services ES assembler and write an application that uses the assembler. To use LiveCycle Data Services ES with ColdFusion, you have to be familiar with ColdFusion components; accessing and using data in ColdFusion applications; and using LiveCycle Data Services ES.

Using Server-Side ActionScript

Adobe ColdFusion server configuration includes the Flash Remoting service, a module that lets Adobe Flash developers create server-side ActionScript. These ActionScript files can directly access ColdFusion query and HTTP features through two new ActionScript functions: CF.query and CF.http.

Chapter 11: Requesting and Presenting Information

Requesting and Presenting Information

Adobe ColdFusion lets you request and present information through multiple formats.

Introduction to Retrieving and Formatting Data

Adobe ColdFusion lets you retrieve and format data. You can use forms to get user data and control the data that a dynamic web page displays. You can also populate a table with query results and use ColdFusion functions to format and manipulate data. To use these features, you should be familiar with HTML forms.

Building Dynamic Forms with cfform Tags

You can use the cfform tag to create rich, dynamic forms with sophisticated graphical controls, including several Java applet or Flash controls. You can use these controls without writing a line of Java or Flash code.

Validating Data-Developing guide

You can validate data in Adobe ColdFusion, including form data, variable data and function parameters.

Creating Forms in Flash

You can create effective forms in Adobe Flash format, in which Adobe ColdFusion displays forms using Flash, not HTML.

Creating Skinnable XML Forms

You can create XML skinnable forms, which are forms that generate XForms-compliant XML and are normally formatted using an XSLT (extensible stylesheet language transformations) skin. You can use XML skinnable forms with the skins that Adobe ColdFusion provides without having any knowledge of either XML or XSLT. For information on using XML with ColdFusion, see Using XML and WDDX.

Using Ajax Data and Development Features

Adobe ColdFusion supports Ajax features to use data dynamically in web pages. For information on ColdFusion Ajax user interface capabilities, see Using Ajax User Interface Components and Features.

Using Ajax User Interface Components and Features

Use Adobe ColdFusion Ajax-based layout and form controls and other Ajax-based user interface capabilities to create a dynamic application. For information about how ColdFusion uses the Ajax framework in general, or how to use ColdFusion Ajax data and programming capabilities, including binding to form data and managing JavaScript resources, see Using Ajax Data and Development Features.

Chapter 12: Office file interoperability

Office file interoperability

Adobe ColdFusion provides interfaces to work with PDF, Adobe Flash, and Adobe Connect. ColdFusion 9 extended the integration support to OpenOffice and Microsoft Office application formats such as Excel, PowerPoint, and SharePoint.

Office interoperability supports both OpenOffice and Apache POI libraries (see http://poi.apache.org/ for information on Apache POI). OpenOffice libraries support conversion of all Office file formats, including Word documents to PDF. When you use the cfdocument, cfpresentation, or cfspreadsheet tags to convert Office files, the tags first search for an OpenOffice installation.

If an OpenOffice installation is not found, POI libraries are used. POI libraries support conversion of all office files except Word documents.

See Supported Office conversion formats for the complete list of supported Office conversion formats.

Using cfdocument

In addition to the existing functionality, the {{cfdocument}}tag lets you convert Word documents and PowerPoint presentations to PDF. All versions of Microsoft Word and Microsoft PowerPoint from 97 to 2003 are supported.

Using cfpresentation

The cfpresentation tag is the parent tag for one or more cfpresentationslide tags, where you define the content for the presentation, and the cfpresenter tags, which provide information about the people presenting the slides.

You use the cfpresentation tag to convert a PowerPoint presentation to an Adobe Connect presentation or HTML. Browsers like Internet Explorer, Mozilla Firefox, and Safari are all compatible with the conversion from PPT to a Connect presentation or HTML.

For complete information about cfpresentation, and cfpresentationslide, see CFML Reference.

Using cfspreadsheet

The cfspreadsheet tag lets you manage Excel spreadsheets. The tag lets you do the following:

- Read a spreadsheet file (XLS file) and store it in a ColdFusion spreadsheet object, query, CSV string, or HTML string.
- Write a single sheet to a new XLS from a query, ColdFusion spreadsheet object, or CSV string variable.
- Add a sheet to an existing XLS file. Use the spreadsheet functions to manipulate rows and columns in a spreadsheet and the data in the rows and columns. You can also specify and get comments, values, and formulas for a cell in the spreadsheet. Microsoft Office Excel 2007 is supported by cfspreadsheet and all the spreadsheet functions except the following:
- SpreadSheetAddSplitPane
- SpreadSheetAddFreezePane For detailed information about cfspreadsheet and all the spreadsheet functions, see CFML Reference.

Supported Office conversion formats

The following table lists the conversion formats supported by Office applications, and the CFML tags that support the conversion. It also shows whether OpenOffice installation is required for the conversion.

All versions of Microsoft Word and Microsoft PowerPoint from 97 to 2003 are supported. Also, all versions of Microsoft Excel from Versions 97 to 2007 are supported.

Format		CFML Tag	OpenOffice installation
From	To		
PPT	Connect Presentation	cfpresentation	optional
PPT	HTML	cfpresentation	optional
PPT	PDF	cfdocument	optional
HTML	PPT	cfpresentation	not required
Excel	HTML	cfspreadsheet	not required
Excel	Query	cfspreadsheet	not required
Excel	In-memory Variable	cfspreadsheet	not required
Query	Excel	cfspreadsheet	not required
In-memory variable	Excel	cfspreadsheet	not required
Word	PDF	cfdocument	required

SharePoint integration

You can use ColdFusion with Microsoft Windows SharePoint Services 2.0 or 3.0, and Microsoft Office SharePoint Portal Server 2003 or Microsoft Office SharePoint Server 2007. You can integrate ColdFusion applications with SharePoint features that are exposed as web service actions.

Chapter 13: ColdFusion Portlets

ColdFusion Portlets

You can now build your own portlets by leveraging Adobe ColdFusion components (CFCs). You can create your own portlet using ColdFusion and run it on:

- JBoss portal server
- WebSphere portal server 6.1

Run a ColdFusion portlet on JBoss Portal Server

You can run and access ColdFusion portlets on a JBoss portal server, which can be either local or remote.

- A local host: A portal can access portlets on the same computer where JBoss Portal server exists.
- A remote host: A portal can access portlets deployed on a remote ColdFusion server instance.

Run a ColdFusion portlet on WebSphere Portal Server

To access and run ColdFusion portlets on WebSphere Portal Server 6.1:

1. Create cfusion.war file.
2. Extract it using jar -xvf cfusion.war
3. Create a /portlets directory under cfusion.war.
4. Add portlets to the /portlets directory. Add portlet entries to portlet.xml, present at: {{cfusion.war/WEB-INF/portlet.xm}}l
5. Go to repackage the WAR file using jar cvf cfusion.war.

6 Deploy it through WebSphere Portal Server administrator console. The portlets would be visible present in portlet.xml

7 Create a portal page and add the portlets.

Common methods used in portlet.cfc

Some common methods that are used frequently while creating the ColdFusion component portlet, such as HelloPortlet.cfc, are:

Method	Description	Syntax
doView()	This method renders the portlet content. It is called by the portlet container to allow the portlet to generate the content of the response based on its current state.	{{<cffunction name="doView" returntype="void" output="true"> <cfargument name="renderRequest" type="any" required="true" hint="A javax.portlet.RenderRequest java object"> <cfargument name="renderResponse" type="any" required="true" hint="A javax.portlet.RenderResponse java object"> <!--- User code goes here --> </cffunction>}}
doHelp()	Helper method to serve up the HELP mode.	{{<cffunction name="doHelp" returntype="void" output="true"> <cfargument name="renderRequest" type="any" required="true" hint="A javax.portlet.RenderRequest java object"> <cfargument name="renderResponse" type="any" required="true" hint="A javax.portlet.RenderResponse java object"> <!--- User code goes here --> </cffunction>}}
doEdit()	Helper method to serve up the EDIT mode.	{{<cffunction name="doEdit" returntype="void" output="true"> <cfargument name="renderRequest" type="any" required="true" hint="A javax.portlet.RenderRequest java object"> <cfargument name="renderResponse" type="any" required="true" hint="A javax.portlet.RenderResponse java object"> <!--- User code goes here --> </cffunction>}}

ProcessAction()	Called by the portlet container to allow the portlet to process an action request	{{<cffunction name="processAction" returntype="void" access="public" output="false" hint="Called by the portlet container to allow the portlet to process an action request."> <cfargument name="actionRequest" type="any" required="true" hint="A javax.portlet.ActionRequest java object"> <cfargument name="actionResponse" type="any" required="true" hint="A javax.portlet.ActionResponse java object"> <!--- User code goes here --> </cffunction>}}
Init()	Called by the portlet container to indicate to a portlet that the portlet is being placed into service	{{<cffunction name="init" returntype="void" access="public" output="false" hint="Called by the portlet container to indicate to a portlet that the portlet is being placed into service."> <cfargument name="portletConfig" type="any" required="true" hint="A javax.portlet.PortletConfig java object"> <!--- User code goes here --> </cffunction>}}
processEvent	This is used to consume the event once it is published.	

ColdFusion portlet components

You can configure your ColdFusion portlet components to define its modes, window states, title, scope, and parameters.

For references of ColdFusion Portlet API, see the JSR-168 specification for all{{ javax.portlet.*}} classes.

Currently, WSRP 1.0 is the supported standard for portlets.

JSR-286 Support

ColdFusion 9 also supports JSR-286 specifications. In portlets there are three types of requests: action, event, and render. A portlet first handles an action request, and then an event request, and only after that, it would render any request. Some of the capabilities of JSR-286 include the following:

Chapter 14: Working with Documents, Charts, and Reports

Working with Documents, Charts, and Reports

To create powerful documents, charts, and reports, Adobe ColdFusion provides interfaces to work with PDF, Adobe Flash, Adobe Connect and extends the integration support to OpenOffice and Microsoft Office application formats such as Excel, PowerPoint, and SharePoint.

Manipulating PDF Forms in ColdFusion

You can use Adobe ColdFusion to manipulate PDF forms created in Adobe® Acrobat® Professional and Adobe® LiveCycle™ Designer.

Assembling PDF Documents

You can use Adobe ColdFusion to assemble PDF documents. You create a unified document from multiple source files or pages from multiple files by using the cfpdf and cfpdfparam tags.

Creating and Manipulating ColdFusion Images

You can use Adobe ColdFusion to create and manipulate images, retrieve and store images in a database, retrieve image information for indexes and searches, convert images from one format to another, and write images to the hard drive.

Creating Charts and Graphs

You can use the cfchart tag to display charts and graphs.

Creating Reports and Documents for Printing

You can use Adobe ColdFusion tags, functions, and tools to create pages and reports that are suitable for printing.

Creating Reports with Report Builder

Improve your access to important business data by creating integrated business reports with Adobe ColdFusion Report Builder and CFML.

Creating Slide Presentations

You can use Adobe ColdFusion to create slide presentations.

PDF Generation in ColdFusion 11

See PDF Generation in ColdFusion.

Chapter 15: Using Web Elements and External Objects

Using Web Elements and External Objects

This section helps you understand about using ColdFusion Web Services as well as integrating with JEE and Java elements; using Microsoft .Net assemblies; and integrating with COM and CORBA objects.

Using XML and WDDX

You can use Adobe ColdFusion to create, use, and manipulate XML documents. You can also use Web Distributed Data Exchange (WDDX), an XML dialect, for transmitting structured data, including transferring data between applications and between CFML and JavaScript.

Using Web Services

Web services let you publish and consume remote application functionality over the Internet. When you consume web services, you access remote functionality to perform an application task. When you publish a web service, you let remote users access your application functionality to build it into their own applications.

Using ColdFusion Web Services

ColdFusion can now expose many of its features as document or literal style SOAP web services. You can leverage ColdFusion functionality using web services from other languages like PHP, .NET, or Ruby.You can access the features of the following tags (and their child tags) as SOAP services:

- cfchart
- cfdocument
- cfimage
- cfmail
- cfpop
- cfpdf

Integrating JEE and Java Elements in CFML Applications

You can integrate JEE elements, including JSP pages and servlets; JSP tags; and Java objects, including Enterprise JavaBeans (EJBs); into your Adobe ColdFusion application.

Using Microsoft .NET Assemblies

You can use Adobe ColdFusion to call local or remote Microsoft .NET assembly class methods and access assembly fields. This documentation describes how to configure and run the ColdFusion .NET extension software and how to access and use .NET classes in your ColdFusion code. For information about .NET technology or how to develop .NET applications, see Microsoft .NET documentation.

Integrating COM and CORBA Objects in CFML Applications

You can invoke COM (Component Object Model) or DCOM (Distributed Component Object Model) and CORBA (Common Object Request Broker) objects.

Chapter 16: Using External Resources

Using External Resources

Sending and Receiving E-Mail

You can add interactive e-mail features to your Adobe ColdFusion applications by using the cfmail and cfpop tags. This complete two-way interface to mail servers makes the ColdFusion e-mail capability a vital link to your users.

Interacting with Microsoft Exchange Servers

You can use Adobe ColdFusion to interact with Microsoft Exchange servers to send, get, and manage mail; and to create, get, and manage calendar events, connections, and tasks.

Interacting with Remote Servers

Adobe ColdFusion wraps the complexity of Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP) communications in a simplified tag syntax that lets you extend your site offerings across the web.

Managing Files on the Server

The cffile, cfdirectory, and cfcontent tags handle browser and server file management tasks, such as uploading files from a client to the web server, viewing directory information, and changing the content type that is sent to the web browser. To perform server-to-server operations, use the cfftp tag, described in Performing file operations with cfftp.

Using Event Gateways

Adobe ColdFusion provides event gateways, which you can use when writing applications. You configure an event gateway for an application and deploy the application. To use event gateways, you should have a thorough knowledge of ColdFusion development concepts and practices, including ColdFusion components (CFCs). To write applications for custom gateways that are not provided in ColdFusion, you must also know the details of the event gateway you are using, including its requirements.

Creating Custom Event Gateways

Adobe ColdFusion lets you create event gateways. Building a gateway requires a knowledge of Java programming, including Java event-handling and thread-handling concepts, and of the technology to which you are providing the gateway, including the types of messages that you handle. This documentation also assumes that you have a thorough knowledge of ColdFusion development concepts and practices, including ColdFusion components (CFCs).

Using the ColdFusion Extensions for Eclipse

The Adobe ColdFusion Extensions for Eclipse include wizards that help generate code for common tasks and an extension that lets you connect to remote servers from Adobe Flash Builder and Eclipse. To use the ColdFusion Extensions for Eclipse, you should be familiar with ColdFusion components, as well as accessing and using data in ColdFusion applications. You should also be familiar with Eclipse or Adobe Flash Builder.

Using the Data Services Messaging Event Gateway

Using the Data Services Messaging gateway type provided with Adobe ColdFusion, you can create applications that send messages to and receive messages from LiveCycle Data Services ES. You configure the Data Services Messaging gateway and write and test an application that uses the event gateway. Before you use the Data Services Messaging gateway, become familiar with ColdFusion event gateway principles and programming techniques (see Using Event Gateways). Also be familiar with Adobe LiveCycle Data Services ES.

Using the Data Management Event Gateway

Using the Data Management event gateway type provided with Adobe ColdFusion, you can have ColdFusion applications notify Adobe Flex applications when data managed by a destination has changed. You configure the Data Management event gateway and write an application that uses the event gateway. Before using the Data Management event gateway, become familiar with ColdFusion event gateway principles and programming techniques (see Using Event Gateways). Also, be familiar with LiveCycle Data Services ES.

Using the FMS event gateway

The FMS event gateway provides interfaces between the Flash Media Server 2 and the Adobe ColdFusion server. As a result, ColdFusion applications and Adobe Flash clients can share data. Before you use the gateway, become familiar with ColdFusion event gateway principles and programming techniques (see Using Event Gateways). A basic knowledge of Flash Media Server is also helpful.

Using the Instant Messaging Event Gateways

You can develop an application that uses either of two instant message (IM) event gateway types provided with Adobe ColdFusion: an IBM Lotus Sametime gateway, and an Extensible Messaging and Presence Protocol (XMPP) gateway. Before you use the IM event gateways, become familiar with ColdFusion event gateway principles and programming techniques (see Using Event Gateways).

Using the SMS Event Gateway

You can develop an application that uses the short message service (SMS) event gateway type provided with Adobe ColdFusion. ColdFusion provides tools for developing SMS applications. Before you use the SMS event gateway, become familiar with ColdFusion event gateway principals and programming techniques (see Using Event Gateways). Although not required, a basic knowledge of SMS is helpful.