Torry Harris

Distributed e-enterprise **Computing**

Migrating ColdFusion to WebSphere

Authors: Sunil Prasad

Intended Audience: This paper is primarily intended for managers and architects considering options for migrating applications from ColdFusion platform to WebSphere..

Further Inquiries and Feedback: info@thbs.com

Notices:

1 Introduction

1.1 Scope of Document

This document gives a detailed study of the issues involved in migrating ColdFusion applications to J2EE architecture. It includes the detailed Tag , Function and operator mappings into the J2EE world.

2 Technology Overview

ColdFusion Application is a page based web application, developed using propritary markup language called the **CFML - ColdFusion Markup Language**. ColdFusion Application is a collection of ColdFusionApplicationPage.

2.1 ColdFusion Application Pages

Application pages are the functional parts of a ColdFusion application, including the user interface pages and forms that handle data input and format data output. They can contain ColdFusion tags (CFML), HTML tags, CFScript, JavaScript, and anything else you can normally embed in an ordinary HTML page. The default file extension used for ColdFusion application pages is .CFM.

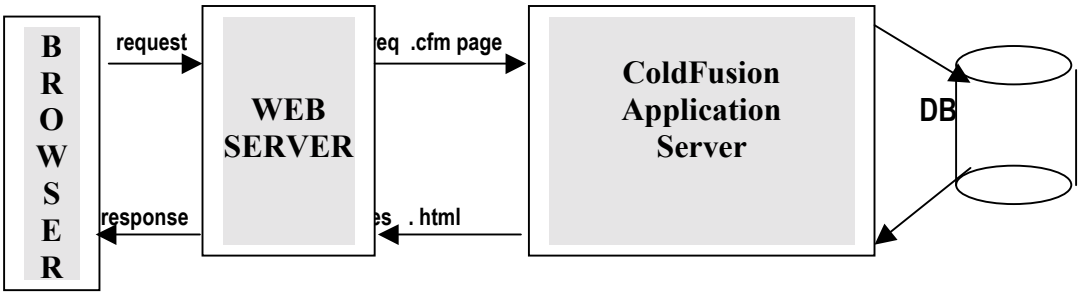
CFML is a tag based language like HTML and XML. It is used to create server side scripts for accessing database, manipulating the data and dynamically generating the web page. CFML is the ColdFusion Application Development Platform containing nearly 105 CFML Tags and 235 CFML Functions.

2.2 ColdFusion Server

ColdFusion Server is an extension to the WebServer that processes ColdFusion Application Page from the WebServer.

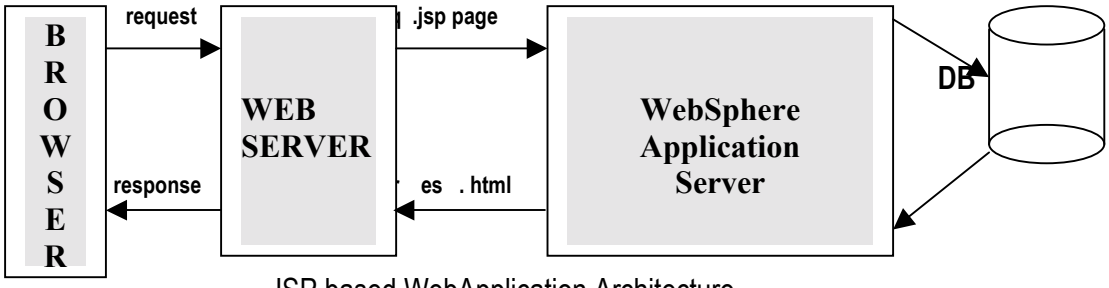
When the Web Browser requests a page that contains CFML tags, the Web server passes the file with the ColdFusion file extension to ColdFusion Server. ColdFusion Server scans the page and processes all CFML tags. ColdFusion Server then returns only HTML and other client-side technologies to the Web server. The Web server passes the page back to the Web Browser.

2.3 Architecture Overview



2.3.1.1.1 ColdFusion WebApplication Architecture

The above mentioned is the simple ColdFusion application architecture. When the client requests a ColdFusion Application page, the webserver passes the .cfm files to the ColdFusion Application Server. ColdFusion application pages are processed on the server at runtime. ColdFusion Server then returns only HTML and other client-side technologies to the Web server. The Web Server passes the HTML page back to the browser client.



JSP based WebApplication Architecture

When the client from the browser requests a JSP Page, the WebServer forwards the request to the WebSphere Application Server. When JSP pages are requested for the first time, they are converted to Servlets and then compiled to a .class file. Compiled Servlet is then loaded to handle the client request. After processing it creates a dynamic HTML page as response and sends it to the client through the WebServer.

3 Migration to WebSphere

Migration requires the conversion of **.cfm files** to **.jsp files**. It requires parsing the ColdFusion Application Page (.cfm file) and replacing ColdFusion tags with equivalent JSP tags and replacing ColdFusion functions with Java API function equivalents or functions within the THBS CF2WAS toolkit.

The sections below provide a high level view of the complexity involved in the Tag and Function mapping. All the Functions/Tags are categorized according to the complexity in the migration.

- **Low** – Equivalent exists or can be achieved easily.
- **Medium/Moderate** – Equivalent functionality can be achieved , but not straight forward.
- **High** – These are areas in which equivalent functionality does not exist in J2EE and Involve complex coding efforts
- **Critical** – This involves serious complexities

3.1 CFML Tags

A detailed study of all the CFML tags are done. Tags are grouped into different categories. For each CFML tag mapping to JSP tag if exists, else possible solution to create the JSP tag equivalent, any issues and the level of complexity are documented.

3.1.1 Complexity Analysis

Table below gives the complexity analysis for mapping CFML tags to J2EE .

Tag	Critical	High	Medium	Low
Form	0	6	6	0
Extensibility	4	3	0	2
Internet Protocol	0	5	2	0
Web Application	3	2	0	0
Database Manipulation	0	7	1	0
Data Output	0	3	2	0
File Management	0	1	1	0
Flow Control	1	1	10	2
Variable Manipulation	0	3	1	0
Others	0	2	1	2
Total	8	33	24	6

Table 4.1

Appendix A lists all the tag mappings.

3.2 CFML Functions

Function mapping involves the mapping of CFML functions to J2EE equivalents. Appendix B gives a detailed description of the CFML functions and J2EE mapping.

3.2.1 Complexity Analysis

Table below gives the complexity analysis for mapping CFML functions to J2EE .

Function	Critical	High	Medium	Low
Array	0	0	11	8
Structure	0	0	10	3
Date & Time	0	7	17	8
System	1	3	3	3
Other	0	3	8	5
Dynamic Evaluation	0	2	0	2
Display & Formatting	0	4	2	9
International	0	2	1	13
List	0	6	3	12
Authentication	0	0	1	3
Decision	0	4	1	12
String	0	6	13	25
Mathematical	0	0	7	27
Query	0	5	0	2
Total	1	42	77	132

Table 5.1

Appendix B lists all the function mappings

4 Appendix A - Tag Mapping

4.1.1 Conversion of CF Tags – Risks Involved

There are some issues involved in creating CFML tags equivalent JSP tags library. There are some CFML tags for which equivalent JSP tag may not be possible.

- CFREGISTRY tag gives you programmatic access to the Windows Registry. The CFREGISTRY tag reads, writes, and deletes keys and values in the system registry. CFREGISTRY is supported on all platforms, including Solaris and HP-UX.

- CFREPORT runs a predefined Crystal Reports report.

- CFSCRIPT tag encloses a code segment containing CFScript ColdFusion server-side scripting language. CFScript is similar to Java Script and VBScript. CFScript uses ColdFusion functions, expressions, and operators. Read and write ColdFusion variables inside of CFScript. One use of CFSCRIPT is to wrap a series of assignment functions that would otherwise require CFSET statements.

- CFWDDX tag serializes and de-serializes CFML data structures to the XML-based WDDX format (Web Distributed Data eXchange). It is used to generate JavaScript statements instantiating JavaScript objects equivalent to the contents of a WDDX packet or some CFML data structures. WDDX is used for representing objects in a language independent manner.

- CFSCHEDULE provides a programmatic interface to the ColdFusion scheduling engine. You can run a specified page at scheduled intervals with the option to write out static HTML pages. This allows you to offer users access to pages that publish data, such as reports, without forcing users to wait while a database transaction is performed in order to populate the data on the page.

- CFCOLLECTION tag allows you to create and administer Verity collections.

- CFINDEX tag used to populate collections with indexed data.

- CFSEARCH tag used to execute searches against data indexed in Verity collections.

  Note: Verity Collection – Indexing and searching

  Free Text searching is a very powerful programming tool that lets you search thousands of files or database records for any text any where within them. ColdFusion implements text searching looping with Verity using the <CFSEARCH> and <CFINDEX> tags. The free text indexing and searching functionality in ColdFusion is based on Verity, Inc.'s SEARCH'97 product.

ColdFusion allows you to index and search collections populated with data from:
· ASCII text files.
· Binary Office documents
· ColdFusion queries resulting from data returned by a <CFQUERY> operation.

- CFASSOCIATE tag allows sub-tag data to be saved with the base tag. This applies to custom CFML tags only.

- CFCACHE allows to speed up pages considerably in cases where the dynamic content doesn't need to be retrieved each time a user accesses the page. To accomplish this, it creates temporary files that contain the static HTML returned from a particular run of the ColdFusion page. You can use CFCACHE for simple URLs and URLs that contain URL parameters.

4.2 Tag Mappings

4.2.1 Category: Database Manipulation Tags

Total CFML Tags = 8
Equivalent Available = Nil
Complexity(Low/Medium/High/Critical) = 0L / 1M / 7H / 0C

CFML Tags	JSP Tags	Complexity
Tag: CFINSERT Description: CFINSERT used to insert new records in data sources. <CFINSERT DATASOURCE="ds_name" DBTYPE="type" DBSERVER="dbms" DBNAME="database name" TABLENAME="tbl_name" TABLEOWNER="owner" TABLEQUALIFIER="tbl_qualifier" USERNAME="username" PASSWORD="password" PROVIDER="COMProvider" PROVIDERDSN="datasource" FORMFIELDS="formfield1, formfield2, ...">	Exist (Yes/No/Partial): No JSP tag to be created to insert records using JDBC API. java.sql.Statement java.sql.PreparedStatement	H
Tag: CFUPDATE Description: The CFUPDATE tag updates existing records in data sources <CFUPDATE DATASOURCE="ds_name" DBTYPE="type" DBSERVER="dbms" DBNAME="database name" TABLENAME="table_name" TABLEOWNER="name"	Exist (Yes/No/Partial): No JSP tag to be created to update records using JDBC API. java.sql.Statement java.sql.PreparedStatement	H

TABLEQUALIFIER="qualifier" USERNAME="username" PASSWORD="password" PROVIDER="COMProvider" PROVIDERDSN="datasource" FORMFIELDS="field_names">		
Tag: CFQUERY CFQUERY passes SQL statements for any purpose to your data source. Not limited to queries. <CFQUERY NAME="query_name" DATASOURCE="ds_name" DBTYPE="type" DBSERVER="dbms" DBNAME="database name" USERNAME="username" PASSWORD="password" MAXROWS="number" BLOCKFACTOR="blocksize" TIMEOUT="milliseconds" CACHEDAFTER="date" CACHEDWITHIN="timespan" PROVIDER="COMProvider" PROVIDERDSN="datasource" DEBUG="Yes/No"> SQL statements </CFQUERY>	Exist (Yes/No/Partial): No JSP tag to be created to execute any SQL using JDBC API. java.sql.Statement java.sql.PreparedStatement	H
Tag: CFSTOREDPROC Description: The CFSTOREDPROC tag is the main tag used for executing stored procedures via an ODBC or native connection to a server database. It specifies database connection information and identifies the stored procedure. <CFSTOREDPROC PROCEDURE="procedure name" DATASOURCE="ds_name" USERNAME="username" PASSWORD="password" DBSERVER="dbms" DBNAME="database name" BLOCKFACTOR="blocksize" PROVIDER="COMProvider" PROVIDERDSN="datasource" DEBUG="Yes/No" RETURNCODE="Yes/No">	Exist (Yes/No/Partial): No JSP tag to be created to execute Stored Procedures using JDBC API java.sql.CallableStatement	H
Tag: CFPROCPARAM Description: The CFPROCPARAM tag is nested within a CFSTOREDPROC tag. You use it to specify parameter information, including type, name, value, and length.	Exist (Yes/No/Partial): No JSP tag to be created to specify information and parameters to	H

<CFPROCPARAM TYPE="IN/OUT/INOUT" VARIABLE="variable name" DBVARNAME="DB variable name" VALUE="parameter value" CFSQLTYPE="parameter datatype" MAXLENGTH="length" SCALE="decimal places" NULL="yes/no">	Stored Procedures using JDBC API java.sql.CallableStatement	
Tag: CFQUERYPARAM Description: CFQUERYPARAM checks the data type of a query parameter. The CFQUERYPARAM tag is nested within a CFQUERY tag. More specifically, it is embedded within the query SQL statement. If you specify its optional parameters, CFQUERYPARAM also performs data validation. SELECT STATEMENT WHERE column_name= <CFQUERYPARAM VALUE="parameter value" CFSQLType="parameter type" MAXLENGTH="maximum parameter length" SCALE="number of decimal places" DBNAME="database name" NULL="Yes/No" > AND/OR ...additional criteria of the WHERE clause...	Exist (Yes/No/Partial): No JSP tag to be created to check query parameter data type and data validation.	H
Tag: CFTRANSACTION Description: Use CFTRANSACTION to group multiple queries into a single unit. CFTRANSACTION also provides commit and rollback processing. <CFTRANSACTION ACTION="BEGIN" or "COMMIT" or "ROLLBACK" ISOLATION="Read_Uncommitted" or "Read_Committed" or "Repeatable_Read" > </CFTRANSACTION>	Exist (Yes/No/Partial): No JSP tag to be created to manage transaction using JTA API. javax.transaction. UserTransaction	M
Tag: CFPROCRESULT Description: The CFPROCRESULT tag is nested within a CFSTOREDPROC tag. This tag's NAME parameter specifies a result set name that other ColdFusion tags, such as CFOUTPUT and CFTABLE, use to access the result set. It also allows you to optionally identify which of the stored procedure's result sets to return. <CFPROCRESULT NAME="query_name" RESULTSET="1-n" MAXROWS="maxrows">	Exist (Yes/No/Partial): No JSP tag to be created to process multiple result set using JDBC API	H

4.2.2 Category: Variable Manipulation Tags

Total CFML Tags = 4
Equivalent Available = Nil
Complexity(Low/Medium/High/Critical) = 0L / 1M / 3H / 0C

CFML Tags	JSP Tags	Complexity
Tag: CFCOOKIE Description: Defines cookie variables, including expiration and security options. <CFCOOKIE NAME="cookie_name" VALUE="text" EXPIRES="period" SECURE="Yes/No" PATH="urls" DOMAIN=".domain">	Exist (Yes/No/Partial): No JSP tag to be created using Servlet API javax.servlet.http.Cookie	M
Tag: CFPARAM Description: CFPARAM is used to test for a parameter's existence, and optionally test its data type, and provide a default value if one is not assigned. .	Exist (Yes/No/Partial): No JSP tag to be created using Servlet API javax.servlet.http. HttpServletRequest	H
Tag: CFSCHEDULE Description: CFSCHEDULE provides a programmatic interface to the ColdFusion scheduling engine. You can run a specified page at scheduled intervals with the option to write out static HTML pages. This allows you to offer users access to pages that publish data, such as reports, without forcing users to wait while a database transaction is performed in order to populate the data on the page.	Exist (Yes/No/Partial): No JSP tag to be created.	H
Tag: CFSET Description: Use the CFSET tag to define a ColdFusion variable. If the variable already exists, CFSET resets it to the specified value.	Exist (Yes/No/Partial): No JSP tag to be created	H

4.2.3 Category: Other Tags

Total CFML Tags = 5
Equivalent Available = 2
Complexity(Low/Medium/High/Critical) = 2L / 1M / 2H / C

CFML Tags	JSP Tags	Complexity
Tag: CFHTMLHEAD Description: CFHTMLHEAD writes the text specified in the TEXT attribute to the <HEAD> section of a generated HTML page. CFHTMLHEAD can be useful for embedding JavaScript code, or placing other HTML tags such as META, LINK, TITLE, or BASE in an HTML page header.	Exist (Yes/No/Partial): No JSP tag to be created using HTML	M
Tag: CFINCLUDE Description: CFINCLUDE lets you embed references to ColdFusion pages in your CFML.	Exist (Yes/No/Partial): Yes <jsp:include> <%@include%>	
Tag: CFSILENT Description: CFSILENT suppresses all output that is produced by the CFML within the tag's scope.	Exist (Yes/No/Partial): Yes <%-- comment --%>	
Tag: CFSETTING Description: CFSETTING is used to control various aspects of page processing, such as controlling the output of HTML code in your pages. One benefit of this option is managing whitespace that can occur in output pages that are served by ColdFusion.	Exist (Yes/No/Partial): No To be checked may not be really required in JSP.	H
Tag: CFCACHE Description: CFCACHE allows you to speed up pages considerably in cases where the dynamic content doesn't need to be retrieved each time a user accesses the page. To accomplish this, it creates temporary files that contain the static HTML returned from a particular run of the ColdFusion page.	Exist (Yes/No/Partial): No To be checked may not be really required.	H

4.2.4 Category: Internet ProtocolTags

Total CFML Tags = 7
Equivalent Available = Nil
Complexity(Low/Medium/High/Critical) = 0L / 2M / 5H / 0C

CFML Tags	JSP Equivalent	Complexity
Tag: CFFTP Description: CFFTP allows users to implement File Transfer Protocol operations	Exist (Yes/No/Partial): No JSP tag has to be created using java API's java.io.* And Servlet API's	H
Tag: CFHTTP Description: The CFHTTP tag allows you to execute POST and GET operations on files. Using CFHTTP, you can execute standard GET operations as well as create a query object from a text file. POST operations allow you to upload MIME file types to a server, or post cookie, formfield, URL, file, or CGI variables directly to a specified server.	Exist (Yes/No/Partial): No JSP tag has to be created using java API's java.net.* java.io.*	H
Tag: CFHTTPPARAM Description: Required for CFHTTP POST operations, CFHTTPPARAM is used to specify the parameters necessary to build a CFHTTP POST.	Exist (Yes/No/Partial): No JSP tag has to be created using java API's java.net.* java.io.*	M
Tag: CFLDAP Description: CFLDAP provides an interface to LDAP (Lightweight Directory Access Protocol) directory servers	Exist (Yes/No/Partial): JSP tag has to be created using java API's java.net.* java.io.*	H
Tag: CFMAIL Description: CFMAIL allows you to send email messages via an SMTP server.	Exist (Yes/No/Partial): No JSP tag has to be created using java API's java.net.* java.io.*	H
Tag: CFMAILPARAM Description: CFMAILPARAM can either attach a file or add a header to a message. If you use CFMAILPARAM, it is nested within a CFMAIL tag	Exist (Yes/No/Partial): No JSP tag has to be created using java API's java.net.* java.io.*	M
Tag: CFPOP Description: CFPOP retrieves and deletes email Messages from a POP mail server	Exist (Yes/No/Partial): JSP tag has to be created using java mail API	H

Issues:
CFFTP,CFHTTP,CFMAIL can be created with the J2EE compliance.
For accomplishing those protocols there are some readymade code implementation.
We can make use of those implementation.

4.2.5 Category: Forms Tags

Total CFML Tags = 12
Equivalent Available = 0
Complexity(Low/Medium/High/Critical) = 0L / 6M / 6H / 0C

CFML Tags	HTML Equivalent	Complexity
Tag: CFAPPLET Description: CFAPPLET allows you to reference custom Java applets that have been previously registered using the ColdFusion Administrator.	Exist (Yes/No/Partial): No HTML Tag: <APPLET> </APPLET> Proposed Solution in case of Non or Partial Existence:	M
Tag: CFFORM Description: CFFORM allows you to build a form with CFML custom control tags that provide much greater functionality than standard HTML form input elements.	Exist (Yes/No/Partial): No HTML Tag: <FORM></FORM> Proposed Solution in case of Non or Partial Existence:	M
Tag: CFGRID Description: CFGRID allows you to place a grid control in a ColdFusion form. A grid control is a table of data divided into rows and columns. CFGRID column data is specified with individual CFGRIDCOLUMN tags.	Exist (Yes/No/Partial): No HTML Tag: Proposed Solution in case of Non or Partial Existence:	M
Tag: CFGRIDCOLUMN Description: you use CFGRIDCOLUMN to specify individual column data in a CFGRID control. Font and alignment attributes used in CFGRIDCOLUMN override any global font or alignment settings defined in CFGRID.	Exist (Yes/No/Partial): No Proposed Solution in case of Non or Partial Existence:	H
Tag: CFGRIDROW Description: CFGRIDROW allows you to define a CFGRID that does not use a QUERY as source for row data. If a QUERY attribute is specified in CFGRID, the CFGRIDROW tags are ignored.	Exist (Yes/No/Partial): No Proposed Solution in case of Non or Partial Existence:	H
Tag: CFGRIDUPDATE	Exist (Yes/No/Partial): No	

Description: CFGRIDUPDATE allows you to perform updates to data sources directly from edited grid data. CFGRIDUPDATE provides a direct interface with your data source.	Proposed Solution in case of Non or Partial Existence:	H
Tag: CFINPUT Description: CFINPUT is used inside CFFORM to place radio buttons, checkboxes, or text boxes. Provides input validation for the specified control type.	Exist (Yes/No/Partial): No HTML Tag: <INPUT> Proposed Solution in case of Non or Partial Existence:	M
Tag: CFSELECT Description: CFSELECT allows you to construct a drop-down list box form control. You can populate the drop-down list box from a query, or using the OPTION tag. Use OPTION elements to populate lists. Syntax for the OPTION tag is the same as for its HTML counterpart.	Exist (Yes/No/Partial): No HTML Tag: <SELECT> </SELECT> Proposed Solution in case of Non or Partial Existence:	M
Tag: CFSLIDER Description: CFSLIDER allows you to place a slider control in a ColdFusion form. A slider control is like a sliding volume control. The slider groove is the area over which the slider moves.	Exist (Yes/No/Partial): No Proposed Solution in case of Non or Partial Existence:	H
Tag: CFTEXTINPUT Description: The CFTEXTINPUT form custom control allows you to place a single-line text entry box	Exist (Yes/No/Partial): Partial HTML Tag: <INPUT> Proposed Solution in case of Non or Partial Existence:	M
Tag: CFTREEITEM Description: Use CFTREEITEM to populate a tree control created with CFTREE with individual elements	Exist (Yes/No/Partial): No Proposed Solution in case of Non or Partial Existence:	H
Tag: CFTREE Description: The CFTREE form custom control allows you to place a tree control	Exist (Yes/No/Partial): No Proposed Solution in case of Non or Partial Existence:	H

4.2.6 Category: Flow Control Tags

Total CFML Tags = 14
Equivalent Available = 1
Complexity(Low/Medium/High/Critical) = 2L / 10M / 1H / 1C

CFML Tags	JSP Tags	Complexity
Tag: CFLOOP Index Loops An index loop repeats for a number of times determined by a range of numeric values. Index loops are commonly known as FOR loops, as in "loop FOR this range of values. " <CFLOOP INDEX="parameter_name" FROM="beginning_value" TO="ending_value" STEP="increment"> ... HTML or CFML code to execute ... </CFLOOP>	Exist (Yes/No/Partial): No JSP tag to be created for wrapping Java Control statement // for loop	M
Conditional Loops A conditional loop also known as while loop iterates over a set of instructions while a given condition is TRUE. <CFLOOP CONDITION="expression"> ..... </CFLOOP>	Exist (Yes/No/Partial): No JSP tag to be created for wrapping Java Control statement // while loop	M
Looping over a Query A loop over a query repeats for every record in the query record set. The CFLOOP results are just like a CFOUTPUT. During each iteration of the loop, the columns of the current row will be available for output. CFLOOP allows you to loop over tags that can not be used inside CFOUTPUT. <CFLOOP QUERY="query_name" STARTROW="row_num" ENDROW="row_num"> </CFLOOP>	Exist (Yes/No/Partial): No JSP tag to be created using Java Control statement and JDBC API. java.sql.ResultSet	M
Looping over a List Looping over a list offers the option of walking through elements contained within a variable or value returned from an expression. In a list loop, the INDEX attribute specifies the name of a variable to receive the next element of the list, and the LIST attribute holds a list or a variable containing a list. <CFLOOP INDEX="index_name"	Exist (Yes/No/Partial): No JSP tag to be created using Java API java.util.StringTokenizer	M

LIST="list_items" DELIMITERS="item_delimiter"> </CFLOOP> Looping over a COM Collection or Structure The CFLOOP COLLECTION attribute allows you to loop over a structure or a COM/DCOM collection object: A COM/DCOM collection object is a set of similar items referenced as a group rather than individually. For example, the group of open documents in an application is a type of collection. A structure can contain either a related set of items or be used as an associative array. Looping is particularly useful when using a structure as an associative array.	Exist (Yes/No/Partial): No JSP tag to be created.	Critical
Tag: CFABORT Description: The CFABORT tag stops processing of a page at the tag location. ColdFusion simply returns everything that was processed before the CFABORT tag. CFABORT is often used with conditional logic to stop processing a page because of a particular condition.	Exist (Yes/No/Partial): No JSP tag to be created	L
Tag: CFBREAK Description: Used to break out of a CFLOOP.	Exist (Yes/No/Partial): No JSP tag to be created for wrapping Java Control statement // break	M
Tag: CFEXECUTE Description: Enables ColdFusion developers to execute any process on the server machine <CFEXECUTE NAME=" ApplicationName " ARGUMENTS="CommandLine Arguments" OUTPUTFILE="Output file name" TIMEOUT="Timeout interval in seconds">	Exist (Yes/No/Partial): No JSP tag to be created.	H

Tag: CFIF/CFELSEIF/CFELSE
 	
Description: Used with CFELSE and CFELSEIF, CFIF lets you create simple and compound conditional statements in CFML. The value in the CFIF tag can be any expression.
 	
<CFIF expression>
HTML and CFML tags
<CFELSEIF expression>
HTML and CFML tags
<CFELSE>
HTML and CFML tags
</CFIF>
	Exist (Yes/No/Partial): No
	
	JSP tag to be created for wrapping Java Control statement
	//if
	//else if
	//else
		M

Tag: CFLOCATION		

Description: CFLOCATION opens a specified ColdFusion page or HTML file. For example, you might use CFLOCATION to specify a standard message or response that you use in several different ColdFusion applications. Use the ADDTOKEN attribute to verify client requests.	Exist (Yes/No/Partial): Yes <jsp: forward>	L

Tag: CFSWITCH/CFCASE/CFDEFAULTCASE
 	
Description: Used with CFCASE and CFDEFAULTCASE, the CFSWITCH tag evaluates a passed expression and passes control to the CFCASE tag that matches the expression result. You can optionally code a CFDEFAULTCASE tag, which receives control if there is no matching CFCASE tag value.
 	
<CFSWITCH EXPRESSION="expression">
<CFCASE VALUE="value1">
.....HTML and CFML tags
</CFCASE>
<CFCASE VALUE=”value2”>
.....HTML and CFML tags
</CFCASE>
<CFDEFAULTCASE>
.....HTML and CFML tags
</CFDEFAULTCASE>
</CFSWITCH>
	Exist (Yes/No/Partial): No
	
	JSP tag to be created for wrapping Java Control statement
	//switch
	//case
	//default
		M

Tag: CFTHROW Description: The CFTHROW tag raises a developer-specified exception that can be caught with CFCATCH tag	Exist (Yes/No/Partial): No JSP tag to be created for wrapping Java Control statement //throw	M
Tag: CFRETHROW Description: Rethrows the currently active exception. <CFRETHROW> preserves the exception's CFCATCH.TYPE and CFCATCH.TAGCONTEXT information.	Exist (Yes/No/Partial): No JSP tag to be created for wrapping Java Control statement //throws	M

Tag: CFTRY/CFCATCH
 	
Description: Used with one or more CFCATCH tags, the CFTRY tag allows developers to catch and process exceptions in ColdFusion pages.
	Exist (Yes/No/Partial): No
	JSP tag to be created for wrapping Java Control statement
	//try
	//catch
	//finally
		M

4.2.7 Category: File Management Tags

Total CFML Tags = 2
Equivalent Available = Nil
Complexity(Low/Medium/High/Critical) = 0L / 1M / 1H / 0C

CFML Tags	JSP Tags	Complexity
Tag: CFDIRECTORY Description: CFDIRECTORY tag is used to handle all interactions with directories. <CFDIRECTORY ACTION="directory action" DIRECTORY="directory name" NAME="query name" FILTER="list filter" MODE="permission" SORT="sort specification" NEWDIRECTORY="new directory name">	Exist (Yes/No/Partial): No JSP tag to be created using Java IO API java.io.File class java.io.FileNameFilter	H
Tag: CFFILE Description: CFFILE tag is used to handle all interactions with files. .	Exist (Yes/No/Partial): No JSP tag to be created using Java IO API java.io.File class java.io.InputStream javax.servlet.ServletRequest CFFILE tag has 8 diferent action attribute and each has 5 corresponding attributes.	Critical

4.2.8 Category: ExtensibilityTags

Total CFML Tags = 9
Equivalent Available = 2
Complexity(Low/Medium/High/Critical) = 2L / 0M / 3H / 4C

CFML Tags	JSP Equivalent	Complexity
Tag: CFCOLLECTION Description: The CFCOLLECTION tag allows you to create and administer Verity collections	Exist (Yes/No/Partial): No JSP tag has to be created using java API's Java.lang.System Java.io.*	Critical
Tag: CFEXECUTE Description: Enables ColdFusion developers to execute any process on the server machine.	Exist (Yes/No/Partial): No JSP tag has to be created using java API's Java.lang.System	H
Tag: CFOBJECT Description: The CFOBJECT tag allows you to call methods in COM, CORBA, and JAVA objects	Exist (Yes/No/Partial): No Proposed Solution in case of Non or Partial Existence:	Critical

Tag: CFREPORT Description: CFREPORT runs a predefined Crystal Reports.	Exist (Yes/No/Partial): No JSP tag has to be created using java API's java.util.*.	Critical
Tag: CFSEARCH Description: Use the CFSEARCH tag to execute searches against data indexed in Verity collections.	Exist (Yes/No/Partial): No JSP tag has to be created using java API's We can accomplish the functionality with the optimal search algorithm using java.util.Hashtable.	H
Tag: CFSERVLET Description: Executes a Java servlet on a JRun engine.	Exist (Yes/No/Partial): Yes JSP Tag:<SERVLET> Proposed Solution in case of Non or Partial Existence:	L
Tag: CFSERVLETPARAM Description: The CFSERVLETPARAM is a child of CFSERVLET. It is used to pass data to the servlet.	Exist (Yes/No/Partial): Yes JSP Tag: <PARAM> Proposed Solution in case of Non or Partial Existence:	L
Tag: CFINDEX Description: Use the CFINDEX tag to populate collections with indexed data	Exist (Yes/No/Partial): No JSP tag has to be created using Java API's Java.util.Hashtable.	H
Tag: CFWDDX Description: The CFWDDX tag serializes and de-serializes CFML data structures to the XML-based WDDX format	Exist (Yes/No/Partial): No JSP tag has to be created using Java API's It has to implement the interface Serializable.	Critical

Issues
CFOBJECT, CFREPORT creating the equivalent J2EE compliance tag will be difficult.
In CFOBJECT they are instantiating the COM , DCOM objects.
In CFREPORT they are generating crystal report which is specific to Microsoft.

4.2.9 Category: Data Output Tags

Total CFML Tags = 5
Equivalent Available = Nil
Complexity(Low/Medium/High/Critical) = 0L / 2M / 3H / 0C

CFML Tags	JSP Tags	Complexity
Tag: CFCOL Description: Defines table column header, width, alignment, and text. Only used inside a CFTABLE. <CFCOL HEADER="text" WIDTH="number"	Exist (Yes/No/Partial): No JSP tag to be created wrapping HTML table header.	M

ALIGN="position" TEXT="text">		
Tag: CFTABLE Description: Builds a table in your ColdFusion page. Use the CFCOL tag to define column and row characteristics for a table. CFTABLE renders data either as preformatted text, or, with the HTMLTABLE attribute, as an HTML table. Use CFTABLE to create tables if you don't want to write your own HTML TABLE tag code, or if your data can be well presented as preformatted text. <CFTABLE QUERY="query_name" MAXROWS="maxrows_table" COLSPACING="number_of_spaces" HEADERLINES="number_of_lines" HTMLTABLE BORDER COLHEADERS STARTROW="row_number"> </CFTABLE>	Exist (Yes/No/Partial): No JSP tag to be created wrapping HTML table to create table and populating the table with database resultset.	H
Tag: CFCONTENT Description: Defines the MIME type returned by the current page. Optionally, allows you to specify the name of a file to be returned with the page. . <CFCONTENT TYPE="file_type" DELETEFILE="Yes/No" FILE="filename" RESET="Yes/No">	Exist (Yes/No/Partial): No JSP tag to be created using Servlet API javax.servlet.http.HttpServletResponse	H
Tag: CFOUTPUT Description: Displays the results of a database query or other operation. <CFOUTPUT QUERY="query_name" GROUP="query_column" GROUPCASESENSITIVE="yes/no" STARTROW="start_row" MAXROWS="max_rows_output"> </CFOUTPUT>	Exist (Yes/No/Partial): No JSP tag to be created using JDBC API	H
Tag: CFHEADER Description: CFHEADER generates custom HTTP response headers to return to the client. <CFHEADER NAME="header_name" VALUE="header_value">	Exist (Yes/No/Partial): No JSP tag to be created using Servlet API javax.servlet.http.HttpServletResponse	M

4.2.10 Category: Web Application Frame WorkTags

Total CFML Tags = 5
Equivalent Available = Nil
Complexity(Low/Medium/High/Critical) = 0L / 0M / 2H / 3C

CFML Tags	JSP Equivalent	Complexity
Tag: CFAPPLICATION Description: Defines scoping for a ColdFusion application, enables or disables storing client variables, and specifies a client variable storage mechanism. By default, client variables are disabled. Also, used to enable session variables and to set timeouts for both session and application variables. Session and application variables are stored in memory.	Exist (Yes/No/Partial): No Sugession: JSP tag has to be created using java API's java.io.* And Servlet API's	Critical
Tag: CFASSOCIATE Description: The CFASSOCIATE tag allows sub-tag data to be saved with the base tag. This applies to custom tags only.	Exist (Yes/No/Partial): No	Critical
Tag: CFAUTHENTICATE Description: The CFAUTHENTICATE tag authenticates a user, setting a security context for the application. See the descriptions of the functions IsAuthenticated and AuthenticatedContext.	Exist (Yes/No/Partial): No	Critical
Tag: CFERROR Description: Provides the ability to display customized HTML pages when errors occur. This allows you to maintain a consistent look and feel within your application even when errors occur.	Exist (Yes/No/Partial): No	H
Tag: CFLOCK Description: The CFLOCK tag provides two types of locks to ensure the integrity of shared data: Exclusive lock Read-only lock	Exist (Yes/No/Partial): No	H

Issues:
The above mentioned tags doesn't have the direct mapping to the JSP tags.Those tags seems to be mission critical tags.We need to work around a lot say

- Understanding the funtionality of the tags in depth.
- Knowing the functionality by running the example application.
- Has to get hands on knowledge by writing a similar application.

5 Appendix B - Function mapping

5.1 Conversion of CF Functions – Risks Involved

- GetMetricData(Monitor_name) On Windows NT, GetMetricData returns all the internal data that is otherwise displayed in the Windows NT PerfMonitor. On UNIX, GetMetricData returns all of the internal data found by using CFStat. For it to work on NT you need to have turned on the PerfMonitor feature from the ColdFusion Administrator. The name of the performance monitor. On Windows NT, the performance monitor is PerfMonitor. On UNIX, it is CFStat.

  On Windows NT, the function returns a ColdFusion structure with the following data fields: InstanceName, PageHits, ReqQueued, DBHits, ReqRunning, ReqTimedOut, BytesIn, BytesOut, AvgQueueTime, AvgReqTime, AvgDBTime, CachePops .

  As of now we have not found equivalent functionality in java API .

5.2 Function Mappings

5.2.1 Category: Array Functions

Total CFML Functions = 19
Available Java Functions = 13 (Including Partial matches)
Complexity Level (Low/Medium/High) – 8L / 11M / 0H / 0C

CFML Functions	JAVA API's	Complexity
Name: ArrayAppend Description: Appends an array index to the end of the specified array. Returns a Boolean TRUE on successful completion	Exist (Yes/No/Partial): Yes Class: java.util.ArrayList Method: add(object) Proposed Solution in case of Non or Partial Existence:	L
Name:ArrayMax Description: Returns the largest numeric value in the specified array	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Have a new method which finds the maximum element in the ArrayList	L
Name:ArraySum Description: Returns the sum of values in the specified array	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Have a new method which adds the numerical values of all the elements in the arraylist	M

Name:ArrayAvg Description: Returns the average of the values in the specified array.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Have a new method which makes use of the method which is implemented for the ArraySum. Divide the resulting value by the number of elements present	L
Name:ArrayMin Description: Returns the smallest numeric value in the specified array.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Have a new method which finds the minimum element in the ArrayList	L
Name:ArraySwap Description: Swaps array values for the specified array at the specified positions. ArraySwap can be used with greater efficiency than multiple CFSETs. Returns a Boolean TRUE on successful completion.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Have a new method which swaps the objects at the specified loactions of the ArrayList	M
Name: ArrayClear Description: Deletes all data in the specified array. Returns a Boolean TRUE on successful completion.	Exist (Yes/No/Partial): Partial Class: java.util.ArrayList Method: clear() Proposed Solution in case of Non or Partial Existence: The clear method deletes the date in the array, but it returns void instead of a boolean value. A wrapper method can be used which returns true all the time.	M
Name:ArrayNew Description: Creates an array of between 1 and 3 dimensions. Array elements are indexed with square brackets: [ ]. Note that ColdFusion arrays expand dynamically as data is added.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Constructor of ArrayList constructs a single dimensional array. Separate implementation is needed for 2 & 3 dimensional arrays.	M
Name: ArrayToList Description: Converts the specified one dimensional array to a list, delimited with the character you specify. Syntax : ArrayToList(array [, delimiter]) *array* – Name of the array containing elements you want to use to build a list *delimiter* - Specify the character(s) you want to use to delimit elements in the list. Default is comma (,)	Exist (Yes/No/Partial): Partial Class: java.util.Arrays Method: Arrays.asList() Proposed Solution in case of Non or Partial Existence: The above method takes a one dimensional array and returns an java.util.List object which is internally represented as [elem1,elem2,elem3...].It is not possible to change the delimiter	L
Name:ArrayDeleteAt	Exist (Yes/No/Partial): Partial	M

Description: Deletes data from the specified array at the specified index position. Note that when an array index is deleted, index positions in the array are recalculated. For example, in an array containing the months of the year, deleting index position [5] removes the entry for May. If you then want to delete the entry for November, you delete index position [10], not [11], since the index positions were recalculated after index position [5] was removed. Returns a Boolean TRUE on successful completion.	Class: java.util.ArrayList Method: Object remove(int index) Proposed Solution in case of Non or Partial Existence: ArrayDeleteAt returns a boolean whereas remove() returns the object which was removed. And remove () throws an exception to indicate if something went wrong. An wrapper function can be used to return a boolean.	
Name:ArrayPrepend Description: Adds an array element to the beginning of the specified array. Returns a Boolean TRUE on successful completion	Exist (Yes/No/Partial): Partial Class: java.util.ArrayList Method: void add(int index, Object element) Proposed Solution in case of Non or Partial Existence: The index parameter should always be set to 0 in case of add() method.	M
Name:IsArray Description: Returns TRUE if value is an array.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: The "Class" class in java has a similar method isArray() which checks if the current class is an array or not. But cheking for the exact dimensions of the array is not there.	M
Name:ArrayInsertAt Description: Inserts data in the specified array at the specified index position. All array elements with indexes greater than the new position are shifted right by one. The length of the array increases by one index. Returns a Boolean TRUE on successful completion.	Exist (Yes/No/Partial): Partial Class: java.util.ArrayList Method: void add(int index, Object element) Proposed Solution in case of Non or Partial Existence: ArrayInsertAt returns a boolean whereas add() returns void. Add() Throws a exception if index is invalid. An wrapper function can be used to have the same method signature	M
Name:ArrayResize Description: Resets an array to a specified minimum number of elements. ArrayResize can provide some performance gains if used to size an array to its expected maximum. Use ArrayResize immediately after creating an array with ArrayNew for arrays greater than 500 elements. Returns a Boolean TRUE on successful completion.	Exist (Yes/No/Partial): Yes Class: java.util.ArrayList Method: void ensureCapacity(int minCapacity) Proposed Solution in case of Non or Partial Existence:	M
Name: ListToArray Description: Converts the specified list into an array.	Exist (Yes/No/Partial): Yes Class: java.util.List Method: toArray() Proposed Solution in case of Non or Partial Existence:	L

Syntax : ListToArray(list [, delimiter]) *list* - Any list *delimiters* - Set of delimiters used in list		
Name:ArrayIsEmpty Description: Determines whether the specified array is empty of data. Returns a Boolean TRUE if specified array is empty, FALSE if not empty.	Exist (Yes/No/Partial): Yes Class: java.util.ArrayList Method: boolean isEmpty() Proposed Solution in case of Non or Partial Existence:	L
Name:ArraySet Description: In a one-dimensional array, sets the elements in a specified range to the specified value. Useful in initializing an array after a call to ArrayNew. Returns a Boolean TRUE on successful completion.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Have a new method which initializes the elements in the given range to the given value	M
Name:ArrayLen Description: Returns the length of the specified array	Exist (Yes/No/Partial): Yes Class: java.util.ArrayList Method: size(object) Proposed Solution in case of Non or Partial Existence:	L
Name:ArraySort Description: Returns the specified array with elements numerically or alphanumerically sorted	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Get the array of objects from the ArrayList , using the static sort method of the Array Class, the array can be sorted.	M

Note :
The Cold Fusion Arrays have the functionality which is almost similar to the functionality provided by the ArrayList class in Java. Most of the functions could be mapped either directly or with some modifications to the ArrayList functions in case of single dimensional array. Support for multi dimensional array is not clear.

5.2.2 Category: Structure Functions

Total CFML Functions = 13
Available Java Functions = 11(including partial)
Complexity Level (Low/Medium/High) – 3L / 10M / 0H / 0C

CFML Functions	JAVA API's	Complexity
Name: IsStruct(variable) Description: Returns true if the variable is a structure.	Exist (Yes/No/Partial):Yes Class: java.lang.class Method:public boolean isInnstance(Object obj) Description :This method returns true if the object passed is non-null can be cast to the reference type represented by this Class object . It returns false otherwise.	

	Proposed Solution in case of Non or Partial Existence:	
Name: StructIsEmpty(structure) Description: Indicates whether the specified structure contains data. Returns TRUE if *structure* is empty and FALSE if it contains data.	Exist (Yes/No/Partial): Partial Class: java.util.Collection Method:public boolean isEmpty() Description : true if this collection has no elements; false otherwise Proposed Solution in case of Non or Partial Existence:The isEmpty () methods in various java classes can be modified to map the same functionality.	M
Name: StructClear(structure) Description: Removes all data from the specified structure. Always returns Yes	Exist (Yes/No/Partial): Partial Class: java.lang.class Method: public Field[] getFields() Description: Returns an array containing Field objects reflecting all the accessible public fields of the class or interface represented by this Class object. Class:java.util.jar.Attribute Method: public void clear() Description: Removes all attributes from this Map. Proposed Solution in case of Non or Partial Existence:For partial functionality many java class methods in Java2 API can be referenced.eg.java.util.List,java.util.Vector etc.We can get all the fields of the object and then set them to null.thereby clearing the data in the object.	M
Name: StructKeyArray(structure) Description: Returns an array of the keys in the specified ColdFusion structure.	Exist (Yes/No/Partial): Partial Class java.util.jar.Attribute Method: public set keyset() Description: Returns a Set view of the attribute names (keys) contained in this Map. Proposed Solution in case of Non or Partial Existence: For partial functionality many java class methods in Java2 API can be referenced.eg.java.util.List,java.util.Vector etc.to achieve full functionality some wrapping is needed.	M
Name: StructCopy(structure) Description: Returns a new structure with all the keys and values of the specified structure.	Exist (Yes/No/Partial): Partial Class: java.util.ArrayList Method: public Object clone() Description: Returns a clone of the ArrayList Class:java.util.Collections. Method: public static void copy(List dest,List src) Description: Copies all of the elements from one	M

	list into another. After the operation, the index of each copied element in the destination list will be identical to its index in the source list. Proposed Solution in case of Non or Partial Existence:almost similar functionalities are available in Java2 API.Action as earlier functions should be taken.	
Name: StructKeyExists(structure,key) Description: Returns TRUE if the specified key is in the specified structure and FALSE if it is not.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence:Java2 API's is* method list can be enhanced and new method can be created on the same lines (eg. Checking a key of the objects in a vector class.)	M
Name: StructCount(structure) Description: Returns the number of keys in the specified structure.	Exist (Yes/No/Partial): Partial Interface: java.text.attributedcharacterIterator Method: public Set getAllAttributeKeys() Description: Returns the keys of all attributes defined on the iterator's text range. The set is empty if no attributes are defined. We need to count the number of items in the set.Hence an extension of the functionality is needed.	M
Name: StructKeyList(*structure*, [*delimiter]*) Description: Returns the list of keys that are in the specified ColdFusion structure	Exist (Yes/No/Partial): Yes Class: java.util.Hashtable Method: public Set keySet() Description: Returns a Set view of the keys contained in this Hashtable Proposed Solution in case of Non or Partial Existence:We do not have a delimiter in the hashtable methods.we have to modify to a certain extent to achieve the goal.	L
Name: StructDelete(*structure*, *key* [, *indicatenotexisting*]) Description: Removes the specified item from the specified structure.	Exist (Yes/No/Partial): Yes Class: java.util.Hashtable Method: public Object remove(Object key) Description: Removes the key (and its corresponding value) from this hashtable. This method does nothing if the key is not in the hashtable. Proposed Solution in case of Non or Partial Existence:	L
Name: StructNew() Description: Returns a new structure.	Exist (Yes/No/Partial): Yes Class: java.Lang.Class Method: public Object newInstance() Description: Creates a new instance of the class represented by this Class object. The class is instantiatied as if by a new expression with an empty argument list. The class is initialized if it has not already been initialized. In java programming Language any Instace of a	M

	user defined class can be obtained as follows: A b=new A();(where"A" is the class ,"b" the instance) Proposed Solution in case of Non or Partial Existence:	
Name: StructFind(*structure*, *key*) Description: Returns the value associated with the specified key in the specified structure	Exist (Yes/No/Partial): Partial Class: java.util.Hashtable Method: public boolean contains(Object value) Description: Tests if some key maps into the specified value in this hashtable. Proposed Solution in case of Non or Partial Existence:	M
Name: StructUpdate(*structure*, *key*, *value*) Description: Updates the specified key with the specified value. Returns Yes if the function is successful and throws an exception if an error occurs.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence:The class org.omg.CORBA.StructMember serves partially as its constructor method public StructMember(String __name,TypeCode __type, IDLType __type_def)Constructs a StructMember object. A method which can take all three arguments and modify the member can show the way.	M
Name: StructInsert(*structure*, *key*, *value [, allowoverwrite]*) Description: Inserts the specified key-value pair into the specified structure. Returns Yes if the insert was successful and No if an error occurs.	Exist (Yes/No/Partial): Partial Class: java.util.Vector Method: public void insertElementAt(Object obj, int index) Description: Inserts the specified object as a component in this vector at the specified index Proposed Solution in case of Non or Partial Existence:Java2 API 's insert* methods can be wrapped to serve desired fnctionality.	M

ISUUES/NOTES:

As structure in cold Fusion itself is a key-value pair entity it has all its function typically pertaining to that.In java we can have somehing of that sort edit into the existinfg API.We can have a java class with a key-value pair implementation(collection ,hashtable,vector .java.sql.struct classes can be looked into) and then inherit and extend its functionality to work with.

5.2.3 Category: String Functions

Total CFML Functions = 44
Available Java Functions = 20(including partial)
Complexity Level (Low/Medium/High) – 25L / 13M / 6H / 0C

CFML Functions	JAVA API's	Complexity
Name: Asc(string) Description: Returns the ASCII value (character code) of the first character of a string. Returns 0 if string is empty.	Exist :No Class: java.lang.String Method: Proposed Solution : A method has to be developed to do the necessary conversion with necessary logic. A name value pair for the conversion can be created.	L
Name: Ljustify(string,length) Description: Returns left-justified *string* of the specified field length.	Exist : No Class: java.lang.String Method: Proposed Solution : String Ljustify(String, length) The return string should have length equal to the "length" argument. The string value passed as argument should start at index 0.	L
Name: Replace(string, substring1, substring2 [,scope]) Description: Returns *string* with occurrences of *substring1* being replaced with *substring2* in the specified scope.	Exist : No Class: java.lang.String Method: substring, compareTo Proposed Solution : There should be two overloaded methods for the scope variable. The logic for search and replace to be developed.	L
Name: Chr(number) Description: Returns a character of a given ASCII value (character code).	Exist :No Class: java.lang.String Method: Proposed Solution : A method has to be developed to do the necessary conversion . A name value pair for the conversion can be created.	L
Name: ListValueCount(*list, value* [, *delimiters*]) Description: Returns the number of instances of a specified value in a list. The underlying search that finds the instances is case-sensitive.	Exist : No Class: Method: Proposed Solution : The method developed should read from list (form), form a query and get result from database.	H
Name: ReplaceList(*string*, *list1*, *list2*) Description: Returns *string* with all occurrences of the elements from the specified comma-delimited list being replaced with their corresponding	Exist :No Class: java.lang.String Method: substring, compareTo Proposed Solution : ReplaceList(*string*, *list1*, *list2*)	M

elements from another comma-delimited list. The search is case-sensitive.	This method should replace strings. The logic has to be developed so that it takes in "list1" and "list2"(which are comma separated strings) and does the replacement.	
Name: Cjustify(*string*, *length*) Description: Centers a string in the specified field length.	Exist : No Class: java.lang.String Method: Proposed Solution : String Cjustify(String, length) The return string should have length equal to the "length" argument. The string value passed as argument should start at "center" of the newly formed string.	L
Name: ListValueCountNoCase(*list, value* [, *delimiters*]) Description: Returns the number of instances of a specified value in a list. The underlying search that finds the instances is not case-sensitive.	Exist : No Class: Method: Proposed Solution : The method developed should read from list (form), form a query and get result from database. Case should be ignored.	H
Name: ReplaceNoCase(string, substring1, substring2 [,scope]) Description: Returns *string* with occurrences of *substring1* being replaced regardless of case matching with *substring2* in the specified scope.	Exist : No Class: java.lang.String Method: compareToIgnoreCase Proposed Solution : There should be two overloaded methods for the scope variable. The logic for replacing should be developed. String.compareToIgnoreCase() method should be used while comparing.	M
Name: Compare(*string1*, *string2*) Description: Performs a case-sensitive comparison of two strings. Returns a negative number if *string1* is less than *string2*; 0 if *string1* is equal to *string2*; or a positive number if *string1* is greater than *string2*.	Exist : Yes Class: java.lang.String Method: compareTo(String) Proposed Solution in case of Non or Partial Existence:	L
Name: LSParseCurrency(*string*) Description: Converts a locale-specific currency string to a number. Attempts conversion through each of the three default currency formats (none, local, international). Returns the number matching the value of *string*.	Exist :No Class: java.lang.String Method: substring, Proposed Solution : The method should have a logic to parse the String and produce a number.	M
Name: REReplace(*string*, *reg_expression*, *substring* [, *scope*]) Description: Returns *string* with a regular expression being replaced with *substring* in the specified scope. This is a case-sensitive search.	Exist : No Class: java.lang.String Method: substring, compareTo Proposed Solution: This function takes arguments which searches for multiple strings and replaces with another string. This will require parsing the argument and then replacing . The implementation for	M

	"Replace" function can be called.	
Name: CompareNoCase(*string1*, *string2*) Description: Performs a case-insensitive comparison of two strings. Returns a negative number if *string1* is less than *string2*; 0 if *string1* is equal to *string2*; or a positive number if *string1* is greater than *string2*.	Exist :Yes Class: java.lang.String Method: compareToIgnoreCase	L
Name: LSParseDateTime(*date-time-string*) Description: A locale-specific version of the ParseDateTime function, except that there is no option for POP date/time object parsing. Returns a date/time object.	Exist :No Class: Method: Proposed Solution: Here a logic for conversion has to be developed.	M
Name: REReplaceNoCase(*string*, *reg_expression*, *substring* [, *scope*]) Description: Returns *string* with a regular expression being replaced with *substring* in the specified scope. The search is case-insensitive.	Exist : No Class: java.lang.String Method: compareToIgnoreCase Proposed Solution: This function takes arguments which searches for multiple strings and replaces with another string. This will require parsing the argument and then replacing . The implementation for "ReplaceNoCase" function can be called.	M
Name: DayOfWeekAsString(*day_of_week*) Description: Returns the day of the week corresponding to *day_of_week*, an integer ranging from 1 (Sunday) to 7 (Saturday).	Exist :Yes Class: java.util.Calendar Method: get	L
Name: LSParseEuroCurrency(*currency-string*) Description: Converts a locale-specific currency string that contains the Euro symbol (□) or sign (EUR) to a number. Attempts conversion through each of the three default currency formats (none, local, international). Returns the number matching the value of *string*.	Exist :No. Class: Method: Proposed Solution: Logic should be developed for parsing and to return the number.	M

Name: Reverse(*string*) Description: Returns *string* with reversed order of characters.	Exist :Yes Class: java.lang.StringBuffer Method: StringBuffer.reverse() Proposed Solution :	L
Name: FormatBaseN(*number*, *radix*) Description: Converts a *number* to a string in the base specified by *radix*.	Exist : Partial Class: java.lang.Integer Method: toHexString, toBinaryString, toOctalString Proposed Solution : The method should use the above mentioned methods for the conversion.	L
Name : LSParseNumber(*string*) Description: LSParseNumber converts a locale-specific string to a number. Returns the number matching the value of string.	Exist : No Class: Method: Proposed Solution : Logic should be developed for parsing and to return the number.	H
Name: Right(*string*, *count*) Description: Returns the rightmost *count* characters of a string.	Exist :Yes Class: java.lang.String Method: String.substring(beginIndex) Proposed Solution : Calculate the beginIndex as String.length() - count	L
Name: Find(*substring*, *string* [, *start*]) Description: Returns the first index of an occurrence of a *substring* in a *string* from a specified starting position. Returns 0 if *substring* is not in *string*. The search is case-sensitive.	Exist :Partial Class: java.lang.String Method: String.indexOf(String, fromIndex) Proposed Solution: This method should be overloaded (Third Parameter). This is similar to java.lang.String.indexOf() method . It should return “0” if substring is not a string.	L
Name: Ltrim(String) Description: Returns *string* with leading spaces removed.	Exist : No Class: java.lang.String, java.lang.Character Method: isSpaceChar, substring Proposed Solution: This can be done by comparing the characters with “space” (Claracter.isSpaceChar()) and getting the index of the first non-space character. Then String.substring() method can be used.	L
Name: RJustify(*string*, *length*) Description: Returns right-justified *string* in the specified field length.	Exist :No Class: java.lang.String Method: Proposed Solution: This can be done by creating a new string with blank spaces in the beginning. The number of blank spaces should be length – string.length()	L
Name: FindNoCase(*substring*, *string* [, *start*])	Exist : Partial Class: java.lang.String Method: equalsIgnoreCase, substring Proposed Solution :	M

Description: Returns the first index of an occurrence of a *substring* in a *string* from a specified starting position. Returns 0 if *substring* is not in *string*. The search is case-insensitive.	This method should be overloaded (Third Parameter). Here the comparison should be done by ignoring the case.	
Name: Mid(*string*, *start*, *count*) Description: Returns *count* characters from *string* beginning at *start* position.	Exist :No Class: Method: Proposed Solution :Here a algorithm has to be developed to parse the string and return the necessary result string.	M
Name: RTrim(*string*) Description: Returns *string* with removed trailing spaces.	Exist :No Class: java.lang.String, java.lang.Character Method: isSpaceChar, substring Proposed Solution: This can be done by comparing the characters with "space character" (Claracter.isSpaceChar()) starting from the last character to the non-space character and making a result string with the trailing spaces trimmed.	L
Name: FindOneOf(*set*, *string* [, *start*]) Description: Return the first index of the occurrence of any character from *set* in *string*. Returns 0 if no characters are found. The search is case-sensitive.	Exist : Partial Class: java.lang.String Method: String.indexOf() Proposed Solution in case of Non or Partial Existence: This requires parsing the argument "set" and finding the index for each in the "set".	H
Name: MonthAsString(*month_number*) Description: Returns the name of the month corresponding to *month_number*.	Exist :Yes Class: java.util.Calendar Method: get Proposed Solution in case of Non or Partial Existence:	L
Name: SpanExcluding(*string*, *set*) Description: Returns all characters from *string* from its beginning until it reaches a character from the *set* of characters. The search is case-sensitive.	Exist :Partial Class: java.lang.String Method:String.indexOf(), String.substring() Proposed Solution: Index of the first occurrence of set is determined. Using this index, the substring can be made.	L
Name: GetToken(*string*, *index* [, *delimiters*]) Description: Returns the specified token in a string. Default delimiters	Exist : Yes Class: java.util.StringTokenizer Method: nextToken() Proposed Solution in case of Non or Partial Existence:	M

are spaces, tabs, and newline characters. If *index* is greater than the number of tokens in *string*, GetToken returns an empty string.		
Name: ParseDateTime(*date-time-string* [, *pop-conversion*]) Description: Returns a date/time object from a string.	Exist : No Class: Method: Proposed Solution: Logic has to be developed for the parsing and conversion. This method should be overloaded.	M
Name: SpanIncluding(*string*, *set*) Description: Returns all characters from *string* from its beginning until it reaches a character that is not included in the specified *set* of characters. The search is case-sensitive.	Exist : Partial Class: java.lang.String Method: String.indexOf, String.subString Proposed Solution in case of Non or Partial Existence: Index of the first occurrence of set is determined. Using this index, the substring can be made.	L
Name: Insert(*substring*, *string*, *position*) Description: Inserts a *substring* in a *string* after a specified character *position*. Prepends the *substring* if *position* is equal to 0.	Exist : Partial Class: java.lang.String Method: substring Proposed Solution : This method should form a new string by appending the substrings from the original string and inserting the new string.	L
Name: REFind(*reg_expression*, *string* [, *start*] [, *returnsubexpressions*]) Description: Returns the position of the first occurrence of a regular expression in a string starting from the specified position. Returns 0 if no occurrences are found. This search is case sensitive.	Exist : No Class: java.lang.String Method: indexOf Proposed Solution : Logic have to be developed in the method which parses the argument, and searches for the occurrence of the expression. Java.lang.String.indexOf() method has to be used.	M
Name: ToBase64(*string or binary_object*) Description: Returns the Base 64 representation of the *string or binary object. Base64 is a format that uses printable characters, allowing binary data to be sent in forms and Email, and stored in a database or file.*	Exist :No Class: Method: Proposed Solution : The logic for this conversion has to be developed.	H
Name: JSStringFormat(*string*) Description: Returns a string that is safe to use with JavaScript.	Exist : No Class: Method: Proposed Solution : Logic has to be developed to remove characters from string so that it can be used with Java	H

	Script.	
Name: REFindNoCase(*reg_expression*, *string* [, *start*] [, *returnsubexpressions*]) Description: Returns the position of the first occurrence of a regular expression in a string starting from the specified position if the *returnsubexpressions* parameter is not set to True. Returns 0 if no occurrences are found.The search is case-insensitive.	Exist : No Class: java.lang.String Method: indexOf Proposed Solution : Logic have to be developed in the method which parses the argument, and searches for the occurrence of the expression. Java.lang.String.indexOf() method has to be used.	M
Name: Ucase(String) Description: Returns *string* converted to uppercase.	Exist :Yes Class: java.lang.String Method: toUpperCase Proposed Solution :	L
Name: Left(*string*, *count*) Description: Returns the count of characters from the beginning of a string argument.	Exist :Yes Class: java.lang.String Method: String.substring(beginIndex, endIndex) Proposed Solution : Here , the endIndex is the “count” value.	L
Name: RemoveChars(*string*, *start*, *count*) Description: Returns *string* with *count* characters removed from the specified starting position. Return 0 if no characters are found.	Exist : Partial Class: java.lang.String Method: subString(beginIndex, endIndex) Proposed Solution : The method should get the substring from the original string and form a new string.	L
Name: Val(*string*) Description: Returns a number that the beginning of a string can be converted to. Returns 0 if conversion is not possible.	Exist : Partial Class: java.lang.String, java.lang.Character, java.lang.Integer Method: charAt , substring, isDigit Proposed Solution in case of Non or Partial Existence: This method will check each character till it finds a non-numeric value. Get the index. Get the substring. And convert to and integer using the constructor of java.lang.Integer.	L
Name: Len(*string or Binary object*) Description: Returns the length of a string or a binary object.	Exist : Yes Class: java.lang.String Method: length() Proposed Solution : Has to be checked with binary object.	L

Name: RepeatString(*string*, *count*) Description: Returns a string created from *string* being repeated a specified number of times.	Exist : Partial Class: java.lang.String Method: Proposed Solution : The result string can be created in a loop.	L

Notes:

a. There are methods which can be downloaded from the net.
ToBase64(String). We have to make sure about the functionality through testing.

b. Some methods may not be required after the migration.
JSStringFormat(String). This method returns a string which is safe to be used with Java Script.

c. There can be a package say util
There should be methods for

i. Separating the "list arguments" commonly used with the CFML functions.

5.2.4 Category: Query Functions

Total CFML Functions = 7
Available Java Functions = 3
Complexity Level (Low/Medium/High) – 2L / 0M / 5H / 0C

CFML Functions	JAVA API's	Complexity
Name: IsQuery Description: Returns TRUE if *value* is a query.	Exist (Yes/No/Partial): Yes Class: java.sql.Statement Method: execute(String) throws SQLException Proposed Solution in case of Non or Partial Existence:	L
Name: QuerySetCell Description: Sets the cell in a specified column to a specified value. If no row number is specified, the cell on the last row will be set. Returns TRUE.	Exist (Yes/No/Partial): Yes Class: java.sql.ResultSet Method: updateObject(String/int , Object), update<type>(String/int,<type>) Proposed Solution in case of Non or Partial Existence:	L
Name: QueryAddColumn Description: Adds a new column to a specified query and populates the column's rows with the contents of a one-dimensional array. Returns the query object with the additional column. Padding is added, if necessary, on the query columns to ensure that all columns have the same number of rows.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: If the query's data is for a temporaray purpose, which is not going to be stored in database, any of the collection classes can be used to store tabular form of data with rows and columns.	H
Name: QuotedValueList	Exist (Yes/No/Partial): No Class:	H

Description: Returns a comma-separated list of the values of each record returned from a previously executed query. Each value in the list is enclosed in single quotes.	Method: Proposed Solution in case of Non or Partial Existence: Alternate Method has to be defined using ResultSet class extracting the values of all fields of a row added to a String looping through the entire set of records.	
Name: QueryAddRow Description: Adds a specified number of empty rows to the specified query. Returns the total number of rows in the query that you are adding rows to.	Exist (Yes/No/Partial): Partial Class: java.sql.resultSet Method: insertRow() Proposed Solution in case of Non or Partial Existence: Description : Inserts the contents of the insert row into the result set and the database. Must be on the insert row when this method is called.	H
Name: ValueList Description: Returns a comma-separated list of the values of each record returned from a previously executed query.	Exist(Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Alternate Method has to be defined using ResultSet class extracting the values of all fields of a row added to a String looping through the entire set of records.	H
Name: QueryNew Description: Returns an empty query with a set of columns or an empty query with no columns. See Usage for more information.	Exist(Yes/No/Partial):No Class: Method: Proposed Solution in case of Non or Partial Existence: If the query's data is for a temporaray purpose, which is not going to be stored in database, any of the collection classes can be used to store tabular form of data.	H

Issues/Concerns : Cfml query tags like QueryNew,QueryAddColumn doesnot have any type of java contemporary methods.These tags help in generating table structures used for storing temporary database whereas java doesnot support any such class .Collections can be used to some extent to obtain the functionality of the above mentioned tags.

5.2.5 Category: Other Functions

Total CFML Functions = 16
Available Java Functions = 12(including partial matches)
Complexity Level (Low/Medium/High) – 5L / 8M / 3H / 0C

CFML Functions	JAVA API's	Complexity level
Name: CreateObject Description: Allows you tocreate COM, CORBA, and JAVA objects.	Exist (Yes/No/Partial): Yes Class: org.omg.CORBA.ORB Method: public abstract Any create_any() Description: Creates an IDL Any object initialized	L

	to contain a Typecode object whose kind field Creating any java object in java platform can be done by using any class' constructorAS the cfml function just needs the type and class of the object created.For eg. We can create an object as following(there can be any java object inplace .Creates a new java object that is a copy of this Graphics object. Proposed Solution in case of Non or Partial Existence:	
Name: GetClientVariablesList Description: Returns a comma-delimited list of non-readonly client variables available to a template. This list contains the custom client variables about a particular client. However the standard The standard system-provided client variables (CFID, CFToken, URLToken, HitCount, TimeCreated, and LastVisit) are not returned in the list.	Exist (Yes/No/Partial): Partial For Client state management(which matches the CFML API's standard system-provided variables)we can use the the following- Class: javax.servlet.http.HttpSession Method: public java.lang.String getId() Description: Returns a string containing the unique identifier assigned to this session. The identifier is assigned by the servlet engine and is implementation dependent. Class: javax.servlet.http.HttpServletRequest Method: public Cookie[] getCookies() Description:Returns an array of all the Cookies included with this request, or null if the request has no cookies Proposed Solution in case of Non or Partial Existence:We need to write a new Function(functions/classes)in order to satisfy the objective.Ideally we save the custom client variables in a database or a persistent bean.but I feel we need to think in a different line here.	M
Name: CreateUUID Description: Returns a Universally Unique Identifier (UUID)	Exist (Yes/No/Partial): Partial Class: java.rmi.server.UID. Method: public UID() Description: Creates a pure identifier that is unique with respect to the host on which it is generated Class: javax.servlet.http.HttpSession Method: public java.lang.String getId() Description: Returns a string containing the unique identifier assigned to this session. The identifier is assigned by the servlet container and is implementation dependent. Proposed Solution in case of Non or Partial Existence:No actual match found.We can have unique ids to identify clients in a client-server app and can have some other unique(typecode object)representation to access in a distributed environment.	L
Name: GetTickCount	Exist (Yes/No/Partial):Partial	M

Description: Returns a millisecond clock counter that can be used for timing sections of CFML code or any other aspects of page processing.	Class: java.lang.System
	Method: public static long currentTimeMillis()
	Description: Returns the current time in milliseconds. , between the current time and midnight, January 1, 1970 UTC.
	Consder a Jsp Page:
	<html>
	<body >
	<%
	System s;
	Time t1,t2,t3;
	t1=s.currentTimeMills();
	←------ jsp page functionality-------→
	t2=s.currentTimeMills();
	t3=can get the value. Of(t2-t1) and get the processing time.
	Proposed Solution in case of Non or Partial Existence:	

Name: Decrypt Description: Decrypts an encrypted string.	Exist (Yes/No/Partial): Yes Class: javax.crypto.Cipher Description: This class provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework. Cipher in=Cipher.getInstance(cipheralgo,"SUN"); In.init(Cipher.ENCRYPT_MODE,key); CipherInputStream cIn=new CipherInputStream(bIn,in); Other ref. Class: javax.crypto.CipherInputStream Proposed Solution in case of Non or Partial Existence:	L
Name: PreserveSingleQuotes Description: Prevents ColdFusion from automatically "escaping" single quotes contained in *variable*.	Exist (Yes/No/Partial): Partial Class: java.lang.String Method:No Existing single method Proposed Solution in case of Non or Partial Existence: Strings are constant; their values cannot be changed after they are created char data[] = {'a', 'b', 'c'}; String str = new String(data); We can check the string objects by Public char charAt(int index) eg. String s; if((s.charAt(0))==""" && if(s.endsWith())==""") then accept as satisfied condition; or the other methods like int compareTo(String anotherString) boolean equals(Object anObject) boolean startsWith(String prefix) etc. can be used.	M
Name: DeleteClientVariable Description: Deletes the client variable specified by *name*. Returns a Boolean TRUE when variable is successfully deleted, even if variable	Exist (Yes/No/Partial): Partial As discussed here we are writing the solutions for the standard system variables found. Class: javax.servlet.http.Cookie Method: public void setMaxAge(int expiry)	M

did not previously exist. To test for the existence of a variable, use IsDefined.	Class:javax.servlet.http.HttpSession. Method:Void RemoveAttribute(java.lang.String name) Description: Removes the object bound with the specified name from this session. Proposed Solution in case of Non or Partial Existence:The client information comes through cookie.we can delete the cookie by setting the maxage to zero. Eg. Cookie c; c.setMaxAge(0); res.addCookie(c);(res=HttpResponseObject)	
Name: QuotedValueList Description: Returns a comma-separated list of the values of each record returned from a previously executed query.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence:At present the JDBC2.0 API does not support such functionality.For enhancement Recommended Java.sql.Resultset, the getXXX methods and interface ResultsetMetaData helps in querying and getting the result.StringBuffer class's append method can be used after getting the resultset object. Resultset rs=stmt.executeQuery(sql); While(rs.next) StringBuffer s=rs.getArray(string col).append(“”).toString: The approach like above can be made.	H
Name: Encrypt Description: Encrypts a String.	Exist (Yes/No/Partial): Partial We can actually achieve encryptuion using Java Cryptographic architecture(JCA).This is done by JCE 1.2.1(java API for cryptography support) Encrypt: Cipher out=Cipher.getInstance(cipheralgo,”SUN”); Out.init(Cipher.ENCRYPT_MODE,key); CipherOutputStream cIn=new CipherOutputStream(bout,out); Other ref. Class: javax.crypto.CipherOutputStream Proposed Solution in case of Non or Partial Existence:	L
Name: StripCR Description: Returns *string* with all carriage return characters removed.	Exist (Yes/No/Partial): No Class: java.lang.String Method: public String replace(char oldChar,char newChar), Public String trim(). Description:First method Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar. The second one Removes white space from both ends	M

	of this string. Proposed Solution in case of Non or Partial Existence: String s1,s2,s3; S1="string with carriage return chars"; S2=s1.replace(char carriagereturn,' '); S3=s2.trim(); System'out.println(s3);	
Name: GetBaseTagData Description: Returns an object that contains data (variables, scopes, etc.) from a specified ancestor tag.	Exist (Yes/No/Partial): Partial Class: javax.servlet.jsp.tagext.Tag javax.servlet.jsp.tagext.TagData javax.servlet.jsp.tagext.TagInfo Method: public Tag getParent() Description: the parent extension tag instance Method: public TagData getTagData() Description: return the immmutable TagData for this tag Proposed Solution in case of Non or Partial Existence:The approach should be on the lines of working with the classes in conjunction.	H
Name: URLEncodedFormat Description: Returns a URL-encoded *string*. Spaces are replaced with + and all non-alphanumeric characters with equivalent hexadecimal escape sequences.	Exist (Yes/No/Partial): Yes Class: java.net.URLEncoder Method: public static String encode(String s) Description: Translates a string into x-www-form-urlencoded format. Proposed Solution in case of Non or Partial Existence:	L
Name: GetBaseTagList Description: Returns a comma-delimited list of uppercase ancestor tag names. The first element of the list is the parent tag. If you call this function for a top-level tag, it returns an empty string.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence:Almost Same as GetBaseTagData. With a difference that here we need an array of Parent tags from getTagData() meyhods.	H
Name: ValueList Description: Returns a comma-separated list of the values of each record returned from a previously executed query	Exist (Yes/No/Partial):Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Almost Same as QuotedValueList And here again we are to operate upon JDBC2.0 APIs.	M
Name: GetBaseTemplatePath Description: Returns the fully specified path of the base template	Exist (Yes/No/Partial): No Class: java.io.File Method: public File getParentFile() public String getParent() public String getAbsolutePath() Description: Returns the pathname string of this abstract pathname's parent, or null if this pathname does not	M

	name a parent directory. Returns the absolute pathname string of this file. Proposed Solution in case of Non or Partial Existence:	
Name: WriteOutput Description: Appends text to the page output stream. Although you can call this function anywhere within a page, it is most useful inside a CFSCRIPT block.	Exist (Yes/No/Partial): Yes Class: javax.servlet.http.HttpServletResponse Method: public java.io.PrintWriter getWriter() or public ServletOutputStream getOutputStream() Description: First one returns Returns a PrintWriter object that can send character text to the client Second one returns Returns a ServletOutputStream suitable for writing binary data in the response Proposed Solution in case of Non or Partial Existence: HttpservletResponse res; PrintWriter out=res.getWriter(); Enhancement on these lines.	M

Issues/Notes:
The GetClientVariables() method returns the custom client variables,and not the system supplied variables and in a j2ee environment we save custom client data(suppose favourite colour of page)in a component(bean) and retrieve from there.So we need to develop methods to access this data and same applies for the DeleteClientVariables() method.
The GetBaseTemplate* functions' mapping can be done in java's Get* methods,for which we need to write fresh code and aciheve the required functionality.
We have Resultset Object in JDBC API.For the DataBase access related methods like QuotedValueList,ValueList we need to extend functionality of our Jdbc resultset methods.

5.2.6 Category: Mathematical Functions

Total CFML Functions = 34
Available Java Functions = 28
Complexity Level (Low/Medium/High) – 27L / 7M / 0H / 0C

CFML Functions	JAVA API's	Complexity
Name: ACos Description: Returns the arccosine of a number in radians. The arccosine is the angle whose cosine is *number*	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: acos(double) Proposed Solution in case of Non or Partial Existence:	L
Name: Ceiling Description: Returns the closest integer greater than	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: ceil(double)	L

a given number.	Proposed Solution in case of Non or Partial Existence:	
Name: Min Description: Returns the minimum, or smaller, value of two numbers.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: min(long , long) Proposed Solution in case of Non or Partial Existence:	L
Name: ASin Description: Returns the arcsine of a number in radians. The arcsine is the angle whose sine is *number*.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: asin(double) Proposed Solution in case of Non or Partial Existence:	L
Name: Cos Description: Returns the cosine of a given angle in radians.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: cos(double) Proposed Solution in case of Non or Partial Existence:	L
Name: Pi Description: Returns the number 3.14159265358979, the mathematical constant (read as Pi), accurate to 15 digits.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: Proposed Solution in case of Non or Partial Existence: Observation: Java has a contemporary variable equivalent to Pi method: public final static double : PI	L
Name: Atn Description: Returns the arctangent of a number. The arctangent is the angle whose tangent is *number*.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: atan(double) Proposed Solution in case of Non or Partial Existence:	L
Name: DecrementValue Description: Returns integer part of *number* decremented by one.	Exist (Yes/No/Partial): No Class: java.lang.Integer Method: parseInt() Operator: Proposed Solution in case of Non or Partial Existence:An alternative method has to be defined decrementing the integer value by 1 and truncating the decimal portion.	M
Name: Rand Description: Returns a random decimal number in the range 0 to 1.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: random() Proposed Solution in case of Non or Partial Existence:	L
Name: BitAnd Description: Returns the bitwise AND of two long integers.	Exist (Yes/No/Partial): Yes Class: java.lang.BigInteger Method: and(BigInteger) Proposed Solution in case of Non or Partial Existence:	L

Name: Exp Description: Returns e raised to the power of *number*. The constant e equals 2.71828182845904, the base of the natural logarithm.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: exp(double) Proposed Solution in case of Non or Partial Existence:	L
Name: Randomize Description: Seeds the random number generator in ColdFusion with the integer part of a *number*. By seeding the random number generator with a variable value, you help to ensure that the Rand function generates highly random numbers.This method has to be called before invoking Rand method.	Exist (Yes/No/Partial): Yes Class: java.util.Random Method: setSeed(long) Proposed Solution in case of Non or Partial Existence:	L
Name: BitMaskClear Description: Returns *number* bitwise cleared with *length* bits beginning from *start*.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: An alternate method has to be defined using the Integer,StringBuffer and String classes to obtain the functionality of the tag.	M
Name: Fix Description: Returns the closest integer less than *number* if *number* is greater than or equal to 0. Returns the closest integer greater than *number* if *number* is less than 0.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: An alternate method has to be defined using the ceil and floor methods.	M
Name: RandRange Description: Returns a random integer between two specified numbers. Note: Requests for random integers greater than 100,000,000 will result in non-random behavior.	Exist (Yes/No/Partial): Yes Class: java.util.Random Method: nextInt(int) Proposed Solution in case of Non or Partial Existence:	L
Name: BitMaskRead Description: Returns the integer created from *length* bits of *number* beginning from *start*.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: An alternate method has to be defined using the Integer,StringBuffer ,BitSet and String classes to obtain the functionality of the tag.	M
Name: IncrementValue Description: Returns integer part of *number* incremented by one.	Exist (Yes/No/Partial): No Class: java.lang.Integer Method: parseInt() Proposed Solution in case of Non or Partial Existence:An alternative method has to be defined incrementing the integer value by 1 and further truncating the decimal portion.	M

Name: Round Description: Rounds a number to the closest integer .	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: round(long/double) Proposed Solution in case of Non or Partial Existence:	L
Name: BitMaskSet Description: Returns *number* bitwise masked with *length* bits of *mask* beginning from *start*.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: An alternate method has to be defined using the Integer,StringBuffer and String classes to obtain the functionality of the tag.	M
Name: InputBaseN Description: Returns the number obtained by converting *string* using the base specified by *radix*, an integer ranging from 2 to 36.	Exist (Yes/No/Partial): Yes Class: java.lang.Integer Method: valueOf(String,int) Proposed Solution in case of Non or Partial Existence:	L
Name: Sgn Description: Determines the sign of a number. Returns 1 if *number* is positive; 0 if *number* is 0; and -1 if *number* is negative.	Exist (Yes/No/Partial): Yes Class: java.lang.BigInteger/BigDecimal Method: signum() Proposed Solution in case of Non or Partial Existence:	L
Name: BitNot Description: Returns the bitwise NOT of a long integer.	Exist (Yes/No/Partial): Yes Class: java.lang.BigInteger Method: not() Proposed Solution in case of Non or Partial Existence:	L
Name: Int Description: Returns the closest integer smaller than a number.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: floor(long/float) Proposed Solution in case of Non or Partial Existence:	L
Name: Sin Description: Returns the sine of the given angle.	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: sin(double) Proposed Solution in case of Non or Partial Existence:	L
Name: BitOr Description: Returns the bitwise OR of two long integers.	Exist (Yes/No/Partial): Yes Class: java.lang.BigInteger Method: or(BigInteger) Proposed Solution in case of Non or Partial Existence:	L
Name: Log Description: Returns the natural logarithm of a number. Natural logarithms are based on the constant e (2.71828182845904).	Exist (Yes/No/Partial): Yes Class: java.lang.Math Method: log(double) Proposed Solution in case of Non or Partial Existence:	L
Name: Sqr	Exist (Yes/No/Partial): Yes	L

Description: Returns a positive square root.	Class: java.lang.Math Method: sqrt(double) Proposed Solution in case of Non or Partial Existence:	
Name: BitSHLN Description: Returns *number* bitwise shifted without rotation to the left by *count* bits.	Exist (Yes/No/Partial): Yes Class: java.lang.BigInteger Method: shiftLeft(int) Proposed Solution in case of Non or Partial Existence:	L
Name: Log10 Description: Returns the logarithm of *number* to base 10.	Exist (Yes/No/Partial): Partial Class: java.lang.Math Method: log(double) (natural log) Proposed Solution in case of Non or Partial Existence:Conversion factor can be used to obtain the log base 10 value from the natural log value.	M
Name: Tan Description: Returns the tangent of a given angle.	Exist (Yes/No/Partial):Yes Class: java.lang.Math Method: tan(double) Proposed Solution in case of Non or Partial Existence:	L
Name: BitSHRN Description: Returns *number* bitwise shifted without rotation to the right by *count* bits.	Exist(Yes/No/Partial):Yes Class: java.lang.BigInteger Method: shiftRight(int) Proposed Solution in case of Non or Partial Existence:	L
Name: Abs Description: Returns the absolute value of a number. The absolute value of a number is the number without its sign.	Exist(Yes/No/Partial): Yes Class: java.lang.Math Method: abs(long/float/double/int) Proposed Solution in case of Non or Partial Existence:	L
Name: BitXor Description: Returns bitwise XOR of two long integers.	Exist(Yes/No/Partial): Yes Class: java.lang.Math Method: xor(BigInteger) Proposed Solution in case of Non or Partial Existence:	L
Name: Max Description: Returns the maximum, or higher, value of two numbers.	Exist(Yes/No/Partial): Yes Class: java.lang.Math Method: max(long,long) Proposed Solution in case of Non or Partial Existence:	L

Issues/Concerns : Given any of the Cfml Mathematical Tags , defining a corresponding Java Method , if it doesnot exist in the API , requires minimum efforts to accomplish the same functionality using the simple classes like String , StringBuffer , Integer ,Float ,Number etc.,

5.2.7 Category: List Functions

Total CFML Functions = 21
Available Java Functions = 16
Complexity level(Low\Medium\High)- 12L / 3M / 6H

CFML Functions	JAVA API's	Complexity Level
Name: ArrayToList Description: Converts the specified one dimensional array to a list, delimited with the character you specify. Syntax : ArrayToList(array [, delimiter]) *array* - Name of the array containing elements you want to use to build a list *delimiter* - Specify the character(s) you want to use to delimit elements in the list. Default is comma (,)	Exist (Yes/No/Partial): Partial Class: java.util.Arrays Method: Arrays.asList() Proposed Solution in case of Non or Partial Existence: The above method takes a one dimensional array and returns an java.util.List object which is internally represented as [elem1,elem2,elem3...].It is not possible to change the delimiter (it is internally represented by ',').	L
Name: ListLast Description: Returns the last element of the list. Syntax : ListLast(list [, delimiters]) *list* - List whose last element is being retrieved. *delimiters* - Set of delimiters used in list	Exist (Yes/No/Partial): Yes Class: java.util.List Method: get() Returns the element at the specified position in this list.Passing the index of the element will return the last object of the List. Proposed Solution in case of Non or Partial Existence:	L
Name: ListAppend Description:Returns *list* with *value* appended behind its last element. Syntax : ListAppend(list, value [, delimiters]) *list* - Any list *delimiters* - Set of delimiters used in list value – Number or list being added.	Exist (Yes/No/Partial): Yes Class: java.util.List Method: add() or addAll() Appends the specified element to the end of this list. Proposed Solution in case of Non or Partial Existence:	L
Name: ListLen Description: Returns the number of elements in the list Syntax : ListLen(list [, delimiters]) *list* - Any list *delimiters* - Set of delimiters used in list	Exist (Yes/No/Partial): Yes Class: java.util.List Method: size() Proposed Solution in case of Non or Partial Existence:	L
Name: ListChangeDelims Description:Returns *list* with all delimiter characters changed to *new_delimiter* string. Syntax:ListChangeDelims(list, new_delimiter [, delimiters])	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: The delimiters are internal representation they cannot be altered. In the presenr java API we do not have support for that. A proposed solution is to wrap each of the List item with the new_delimiter value and add them back to List replaceing the previous value.	H

Name: ListPrepend Description: Returns *list* with *value* inserted at the first position, shifting all other elements one to the right. Syntax : ListAppend(list, value [, delimiters]) *list* - Any list *delimiters* - Set of delimiters used in list value – Number or list being added.	Exist (Yes/No/Partial): Yes Class: java.util.List Method: add() or addAll() To this method pass the index as '0' to add as first element of the List. Proposed Solution in case of Non or Partial Existence:	L
Name: ListContains Description:Returns the index of the first item that contains the specified substring. The search is case-sensitive. If the substring is not found in any of the list items, it returns zero (0) Syntax : ListContains(list, substring [, delimiters])	Exist (Yes/No/Partial): Yes Class: java.util.List Method: indexOf() Returns the index in this list of the first occurrence of the specified element, or -1 if this list does not contain this element. Proposed Solution in case of Non or Partial Existence:	L
Name: ListQualify Description:Returns a list with a qualifying character around each item in the list, such as double or single quotes. Syntax : ListQualify(list, qualifier [, delimiters] [, elements]) *list* - Any list of items or a variable that names a list *qualifier* - The character that is to be placed at the beginning and end of each item in the list *delimiters* - Set of delimiters used in *list* *elements* - Either the keyword "ALL" or "CHAR." If you specify "ALL," the function qualifies all items in the list. If you specify "CHAR," the function qualifiers only items comprised of alphabetic characters; it does not qualify numeric items	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: A qualifying character around each item in the list, such as double or single quotes is internal representation and cannot be altered.Current Java API does'nt provide any mechanism to alter the delemiters in a list.A possible solution is to wrap each of the List item with single (') or double quote (") and add them back to List replaceing the previous value.	H
Name: ListContainsNoCase Description: Returns the index of the first element of a list that contains the specified substring within elements. The search is case-insensitive. If no element is found, returns 0.	Exist (Yes/No/Partial): Partial Class: java.util.List Method: indexOf() (Here the search is case – sensitive.) Proposed Solution in case of Non or Partial Existence:	M
Name: ListRest Description: Returns *list* without its first element. Returns an empty list (empty string) if *list* has only one element. Syntax : ListRest(list [, delimiters]) *list* - Any list *delimiters* - Set of delimiters used in list	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: It is possible to pass the desired List as the argument to the method and use the remove(0) on this List and return the resultant List.	H
Name: ListDeleteAt	Exist (Yes/No/Partial): Yes	L

Description: Returns *list* with element deleted at the specified position. Syntax :ListDeleteAt(list, position [, delimiters]) *list* - Any list *delimiters* - Set of delimiters used in list *position* - Positive integer indicating the position of the element being deleted. The starting position in a list is denoted by the number 1, not 0	Class: java.util.List Method: remove() Removes the element at the specified position in this list.First element is denoted by 0 (zero). Proposed Solution in case of Non or Partial Existence:	
Name: ListSetAt Description: Returns *list* with *value* assigned to its element at specified position. Syntax : ListSetAt(list, position, value [, delimiters]) *list* - Any list *delimiters* - Set of delimiters used in list value – Any value. *position* - Any position. The first position in a list is denoted by the number 1, not 0.	Exist (Yes/No/Partial): Yes Class: java.util.List Method: add() Inserts the specified element at the specified position in this list. Proposed Solution in case of Non or Partial Existence:	L
Name: ListFind Description: Returns the index of the first occurrence of a value within a list. Returns 0 if no value is found. The search is case-sensitive. Syntax : ListFind(list, value [, delimiters]) *list* – Any list. v*alue* - Number or string that is to be found in the items of the list. *delimiters* - Set of delimiters used in list	Exist (Yes/No/Partial): Partial Class: java.util.List Method: indexOf() (Here the method accepts only objects.) Proposed Solution in case of Non or Partial Existence:	M
Name: ListSort Description: Sorts and delimits the items in a list according to the specified sort type and sort order. Syntax : ListSort(list, sort_type [, sort_order] [, delimiter]) *list* – Any list. *sort_type* – Numeric (sorts numbers), Text (sorts text alphabetically),Textnocase (sorts text alphabetically. The case is ignored) *sort_order* – Asc (Ascending,Default), Desc (Descending) *delimiter* - Set of delimiters used in list	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: A partial alternative, is to convert the List in to an Array (using List.toArray()) and then passing the resultant array to java.util.Arays.sort() for sorting and then converting the resultant array abck to List using java.util.Arays.asList() method.The sorting is only ascending. Current Java API does'nt provide any mechanism to alter the delemiters in a list.	H
Name: ListFindNoCase Description: Returns the index of the first occurrence of a value within a list. Returns 0 if no value was found. The search is case-insensitive. Syntax : ListFindNoCase(list, value [,	Exist (Yes/No/Partial): Partial Class: java.util.List Method: indexOf(). This search is not case-insensitive. Proposed Solution in case of Non or Partial Existence:	M

delimiters]) *list* – Any list. v*alue* - Number or string that is to be found in the items of the list. *delimiters* - Set of delimiters used in list		
Name: ListToArray Description: Converts the specified list into an array. Syntax : ListToArray(list [, delimiter]) *list* - Any list *delimiters* - Set of delimiters used in list	Exist (Yes/No/Partial): Yes Class: java.util.List Method: toArray() Proposed Solution in case of Non or Partial Existence:	L
Name: ListFirst Description: Returns the first element of the list. Syntax : ListFirst(list [, delimiters]) *list* - Any list *delimiters* - Set of delimiters used in list	Exist (Yes/No/Partial): Yes Class: java.util.List Method: get() To this method pass 0(zero) as the argument which will return the first element of the List. Proposed Solution in case of Non or Partial Existence:	L
Name: ListValueCount Description: Returns the number of instances of a specified value in a list. The underlying search that finds the instances is case-sensitive. Syntax : ListValueCount(list, value [, delimiters])	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: It possible to write a method which compares each list element with the specified element.And have a counter varaible which is incrementd by 1 if a match occurs.	H
Name: ListGetAt Description: Returns the element at a given position. Syntax : ListGetAt(list, position [,delimiters]) *list* – Any List *position* - Positive integer indicating the position of the element being retrieved.	Exist (Yes/No/Partial): Yes Class: java.util.List Method: get() Returns the element at the specified position in this list Proposed Solution in case of Non or Partial Existence:	L
Name: ListValueCountNoCase Description: Returns the number of instances of a specified value in a list. The underlying search that finds the instances is not case-sensitive.	Exist (Yes/No/Partial): No Class: Method: Inserts the specified element at the specified position in this list. Proposed Solution in case of Non or Partial Existence: It possible to write a method which compares each list element with the specified element.And have a counter varaible which is incrementd by 1 if a match occurs.But the comparison is case-sensitive.	H
Name: ListInsertAt	Exist (Yes/No/Partial): Yes Class: java.util.List	L

Description: Returns *list* with *value* inserted at the specified position. Syntax : ListSetAt(list, position, value [, delimiters]) *list* - Any list *delimiters* - Set of delimiters used in list value – Any value. *position* - Any position. The first position in a list is denoted by the number 1, not 0.	Method: add() Proposed Solution in case of Non or Partial Existence:	

Issues :
All the functions mentioned above have an optional parameter *delimiters* which is used to specify delimiter other than ','(by default).This is not possible in Java as it is JVM dependent.
ListQualify-This function retwurns a list with a qualifying character around each item in the list, such as double or single quotes.It is not possible to alter the way the value of a list item is represented.
ListContainsNoCase & ListFindNoCase – These functions returns the index of the first occurrence of a value within a list.The search is *case-insensitive*.The corresponding matches in Java API are java.util.List.indexOf(),but the search here is case-sensitive.
ListValueCount & ListValueCountNoCase -- Returns the number of instances of a specified value in a list. The underlying search that finds the instances is not case- sensitive.No match is found which does *case-insensitive* search.Incase the List elements are of *String* type then it is possible to write a custom function that does a *case-insensitive* comparison while doing the search using the java.lang.String.equalsIgnoreCase().
ListSort - Sorts and delimits the items in a list according to the specified sort type and sort order.The sort types can be - Numeric (sorts numbers), Text (sorts text alphabetically),Textnocase (sorts text alphabetically. The case is ignored).

5.2.8 Category: International Functions

Total CFML Functions = 16
Available Java Functions = 14(Including Partial)
Complexity Level(Low\Medium\High)- 13L / 1M / 2H / 0C

CFML Functions	JAVA API's	Complexity
Name: DateConvert Description: Converts local time to Universal Coordinated Time (UTC) or UTC to local time based on the specified parameters. This function uses the daylight savings settings in the executing machine to compute daylight savings time, if required. Syntax : DateConvert(conversion-type, date) conversion-type There are two conversion types: "local2Utc" and "utc2Local." The former converts local time to UTC time. The later converts UTC time to local time. date Any ColdFusion date and time string. In order to create a ColdFusion date and time, use CreateDateTime.	Exist (Yes/No/Partial): Partial Class: java.util.TimeZone Method: public abstract void setRawOffset(int offsetMillis) Sets the base time zone offset to GMT. This is the offset to add *to* UTC to get local time. Parameters: OffsetMillis - the given base time zone offset to GMT. Proposed Solution in case of Non or Partial Existence:	M
Name: GetLocale Description: Returns the locale for the current request. Locales are	Exist (Yes/No/Partial): Yes Class: java.util.Locale Method: public static Locale getDefault() Class: java.text.DateFormat	L

determined by the native operating system. A locale is an encapsulation of the set of attributes that govern the display and formatting of international date, time, number, and currency values. Syntax : GetLocale()	Method: public static final DateFormat getDateTimeInstance(int dateStyle,int timeStyle,Locale aLocale) Proposed Solution in case of Non or Partial Existence:	
Name: GetTimeZoneInfo Description: Syntax : GetTimeZoneInfo() Returns a structure containing time zone information for the machine on which this function is executed. The structure contains four elements.	Exist (Yes/No/Partial): Yes Class: java.util.Date Method: public int getTimezoneOffset() Proposed Solution in case of Non or Partial Existence:	L
Name: LSCurrencyFormat Description: Returns a currency value using the locale convention. Default value is "local." Syntax : LSCurrencyFormat(number [, type]) number The currency value. Type Currency type. Valid arguments are: none -- (For example, 10.00) local -- (Default. For example, $10.00) international -- (For example, USD10.00)	Exist (Yes/No/Partial): Yes Class: java.text.DecimalFormatSymbols Method: public String getCurrencySymbol() Proposed Solution in case of Non or Partial Existence:	L
Name: LSDateFormat Description: Formats the date portion of a date/time value using the locale convention. Like DateFormat LSDateFormat returns a formatted date/time value. If no mask is specified, LSDateFormat returns a date value using the locale-specific format. Syntax : LSDateFormat(date [, mask])	Exist (Yes/No/Partial): Yes Class: java.text.DateFormat, java.text.SimpleDateFormat Method: format() in combinaion with getDateInstance() or getInstance() of java.text.DateFormat class. The methods getDateInstance() or getInstance() also take Locale as one of their parameters. Method: public abstract StringBuffer format(Date date,StringBuffer toAppendTo,FieldPosition fieldPosition) public final String format(Date date) Proposed Solution in case of Non or Partial Existence:	L
Name: LSEuroCurrencyFormat Description: Returns a currency value using the convention of the locale and the Euro as the currency symbol. Default value is "local." Note: The locale is set with the SetLocale function. Syntax : LSEuroCurrencyFormat(currency-number [, type])	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Java(1.2)API does not support EuroCurencyFormat	H
Name: LSIsCurrency Description: Checks whether a string is a locale-specific currency string. Returns TRUE if *string* is a currency string, FALSE otherwise. Syntax : LSIsCurrency(string)	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: NumberFormat.getInstance().format(number) (this method can be used for Locale-specific numeber formats) or new DecimalFormat(pattern).foramt(number) (this method can be used for user-defined patterns)	L

String: The locale-specific currency string.	Proposed Solution in case of Non or Partial Existence:	
Name: LSIsDate Description: Like the IsDate function, LSIsDate returns TRUE if *string* can be converted to a date/time value in the current locale, FALSE otherwise. 5.2.8.1.1 Syntax : LSIsDate(string)	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: NumberFormat.getInstance().format(number) (this method can be used for Locale-specific numeber formats) or new DecimalFormat(pattern).foramt(number) (this method can be used for user-defined patterns) Proposed Solution in case of Non or Partial Existence:	L
Name: LSIsNumeric Description: Like the IsNumeric function, LSIsNumeric returns TRUE if string can be converted to a number in the current locale; otherwise, FALSE. Syntax : LSIsNumeric(string)	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: public Number parse(String text)throws ParseException Throws: ParseException - if the specified string is invalid. Proposed Solution in case of Non or Partial Existence:	L
Name : LSNumberFormat Description: Formats a number using the locale convention. If mask is omitted, the number is formatted as an integer. Syntax : LSNumberFormat(number [, mask])	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: NumberFormat.getInstance().format(number) (this method can be used for Locale-specific numeber formats) or new DecimalFormat(pattern).foramt(number) (this method can be used for user-defined patterns) Syntax: public final String format(double number) public final String format(long number) Proposed Solution in case of Non or Partial Existence:	L
Name: LSParseCurreny Description: Converts a locale-specific currency string to a number. Attempts conversion through each of the three default currency formats (none, local, international). Returns the number matching the value of string. Syntax : LSParseCurrency(string) string : The locale-specific string you want to convert to a number.	Exist (Yes/No/Partial): Yes Class: java.text Method: parse(String text) Proposed Solution in case of Non or Partial Existence:	L
Name: LSParseDateTime Description: A locale-specific version of the ParseDateTime function, except that there is no option for POP date/time object parsing. Returns a date/time object. Syntax : LSParseDateTime(date-time-string) date-time-string : String being converted to date/time object. This string must be in a form that is readable in the current locale setting. By default the locale is set to English (US).	Exist (Yes/No/Partial): Yes Class: java.text.DateFormat Method: public abstract Date parse(String text,ParsePosition pos) Proposed Solution in case of Non or Partial Existence:	L
Name: LSParseEuroCurrency Description:	Exist (Yes/No/Partial): No Class: Method:	H

Converts a locale-specific currency string that contains the Euro symbol (□) or sign (EUR) to a number. Attempts conversion through each of the three default currency formats (none, local, international). Returns the number matching the value of string. Syntax : LSParseEuroCurrency(currency-string) currency-string : The locale-specific string you want to convert to a number.	Proposed Solution in case of Non or Partial Existence: Java(1.2)API does not support EuroCurrency	
Name: LSParseNumber Description: Converts a locale-specific string to a number. Returns the number matching the value of string. Syntax : LSParseNumber(string) string : String being converted to a number.	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: public abstract Number parse(String text,ParsePosition parsePosition) Proposed Solution in case of Non or Partial Existence:	L
Name: LSTimeFormat Description: Returns a custom-formatted time value using the locale convention. Syntax : LSTimeFormat(time [, mask])	Exist (Yes/No/Partial): Yes Class: java.text.DateFormat Method: public abstract StringBuffer format(Date date,StringBuffer to AppendTo, FieldPosition fieldPosition) Formats a Date into a date/time string. Proposed Solution in case of Non or Partial Existence:	L
Name: SetLocale Description: Sets the locale to the specified new locale for the current session. Note: SetLocale returns the old locale in case it needs to be restored. Syntax : SetLocale(new_locale) new_locale :The name of the locale you want to set.	Exist (Yes/No/Partial): Yes Class: java.util.Local Method: getDefault() public static Locale getDefault( ) Common method of getting the current default Locale. public static void setDefault (Locale newLocale) Sets the default locale for the whole JVM. SetDefault does not reset the host locale. Construtor: Locale(String language, String country) Locale(String language, String country, String variant) Proposed Solution in case of Non or Partial Existence:	L

Issues: Most of the functions are available are in java. Out of 16, 14 are available(including partial)in java. 2 functions are not avilable in java.These are LSEuroCurrencyFormat , LSParseEuroCurrency.
These functions are related to runtime attributes and Java API (JDK1.2) doesn't provide any mechanism to determine these values.Euro currency is not recognized by JDK1.2.
For this functions new methods should be written in Java.

5.2.9 Category: Dynamic Evaluation Functions

Total CFML Functions = 4
Available Java Functions = 2
Complexity level(Low\Medium\High)- 2L / 0M / 2H / 0C

CFML Functions	JAVA API's	ComplexityLevel
Name: DE Description: Returns its argument with	Exist (Yes/No/Partial): No Class: Method:	H

double quotes wrapped around it and all double quotes inside it escaped. The DE (Delay Evaluation) function prevents the evaluation of a string as an expression when it is passed as an argument to IIf or Evaluate.	Proposed Solution in case of Non or Partial Existence: Partial match for this java.lang.String. toString() which returns an *String* object for whatever argument that is passed.	
Name: Iif Description: The function evaluates its *condition* as a Boolean. If the result is TRUE, it returns the value of Evaluate(*string_expression1*); otherwise, it returns the value of Evaluate(*string_expression2*).The expressions *string_expression1* and *string_expression2* must be string expressions, so that they do not get evaluated immediately as the arguments of Iif. Syntax : IIf(condition, string_expression1, string_expression2)	Exist (Yes/No/Partial): Yes Operator: ternary operator (*condition*?*exp1:exp2*) Class: Method: Proposed Solution in case of Non or Partial Existence:	L
Name: Evaluate Description: The function evaluates all of its arguments, left to right, and returns the result of evaluating the last argument.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence:The Java interpreter executes an expression from left to right.This also seem to be a Cold fusion specific function imlemetation of which is yet to be there in java API.	H
Name: SetVariable Description:The function sets the variable specified by *name* to *value* and returns the new value of the variable. Syntax: SetVariable(name, value)	Exist (Yes/No/Partial): Yes Operator: assignment operator (=) *name* = *value* Class: Method: Proposed Solution in case of Non or Partial Existence:	L

Issues :
DE - The DE (Delay Evaluation) function prevents the evaluation of a string as an expression when it is passed as an argument to IIf or Evaluate.
Evaluate – This function executes the expression passed to it as String and returns the results.The Java interpreter execute an expression when it encounters one.There is no method available for this purpose in the API.
As these again seems to be close-knit with the JVM and without any existing support .Hence this sort vof functionality attainment need lot of effort does not look easily acheivable.

5.2.10 Category: Display & Formatting Functions

Total CFML Functions = 15

Available Java Functions = 11
Complexity level(Low\Medium\High)- 9L / 2M / 4H / 0C

CFML Functions	JAVA API's	ComplexityLevel
Name: DateFormat Description: Returns a formatted date/time value. If no mask is specified, DateFormat function returns date value using the *dd-mmm-yy* format.	Exist (Yes/No/Partial): Yes Class:java.text.SimpleDateFormat and java.text.DateFormat Method: format() in combinaion with getDateInstance() or getInstance() of java.text.DateFormat class. Proposed Solution in case of Non or Partial Existence:	L
Name: DecimalFormat Description: Returns *number* as a string formatted with two decimal places and thousands separator.	Exist (Yes/No/Partial): Yes Class: java.text.DecimalFormat Method: format() Proposed Solution in case of Non or Partial Existence:	L
Name: DollarFormat Description: Returns *number* as a string formatted with two decimal places, thousands separator, dollar sign. Parentheses are used if *number* is negative.	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: format() Proposed Solution in case of Non or Partial Existence:	L
Name: FormaBaseN Description: Converts a *number* to a string in the base specified by *radix*. Syntax : FormatBaseN(number, radix)	Exist (Yes/No/Partial): Yes Class: java.lang.Integer, java.lang.Long, java.lang.Short, java.lang.Double, java.lang.Float, java.lang.Byte Method: java.lang.Integer.toString(), java.lang.Long.toString() The following methods accept the number to be converted as *String* and the radix as *int* java.lang.Short.valueOf(), java.lang.Double.valueOf(), java.lang.Float.valueOf(), java.lang.Byte.valueOf() Proposed Solution in case of Non or Partial Existence:	L
Name: HTMLCodeFormat Description: Returns HTML escaped *string* enclosed in <PRE> and </PRE> tags. All carriage returns are removed from *string*, and all special characters (> < " &) are escaped. Syntax: HTMLCodeFormat(string [, version])	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: To do this the JVM should recognoize the HTHL tags and Java API does not provide any mechanism to parse the HTML tags.	H
Name: LSCurrencyFormat Description: Returns a currency value using the locale convention. Default value is	Exist (Yes/No/Partial): Partial Class: java.text.NumberFormat, java.text.DecimalFormat Method:	M

"local."	NumberFormat.getInstance().format(number) or NumberFormat.getCurrencyInstance().forma t(number) (this method can be used for Locale-specific numeber formats) or new DecimalFormat(pattern).format(number) (this method can be used for user-defined patterns) Proposed Solution in case of Non or Partial Existence:	
Name: LSDateFormat Description: Formats the date portion of a date/time value using the locale convention. Like DateFormat LSDateFormat returns a formatted date/time value. If no mask is specified, LSDateFormat returns a date value using the locale-specific format.	Exist (Yes/No/Partial): Yes Class:java.text.SimpleDateFormat and java.text.DateFormat Method: format() in combinaion with getDateInstance() or getInstance() of java.text.DateFormat class. The methods getDateInstance() or getInstance() also take Locale as one of their parameters. Proposed Solution in case of Non or Partial Existence:	L
Name: LSEuroCurrencyFormat Description: Returns a currency value using the convention of the locale and the Euro as the currency symbol. Default value is "local." The LSEuroCurrencyFormat function can display the Euro symbol (□) only on Euro-enabled computers, such as Windows NT 4.0 SP4, that have Euro-enabled fonts installed. This function is similar to LSCurrencyFormat except that LSEuroCurrencyFormat displays the Euro currency symbol (□) or the international Euro sign (EUR) if you specify the type as local or international, respectively, and the Euro is the accepted currency of the locale.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Current Java API (JDK1.2) doesn't recognize Euro currency.	H
Name: LSNumberFormat Description: Formats a number using the locale convention. If mask is omitted, the number is formatted as an integer.	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat, java.text.DecimalFormat Method: NumberFormat.getInstance().format(number) (this method can be used for Locale-specific numeber formats) or new DecimalFormat(pattern).format(number) (this method can be used for user-defined patterns) Proposed Solution in case of Non or Partial Existence:	L
Name: NumberFormat Description:Creates a custom-formatted	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat, java.text.DecimalFormat	L

number value. If no mask is specified, returns the value as an integer with a thousands separator. Syntax : NumberFormat(number [, mask])	Method: NumberFormat.getInstance().format(number) (this method can be used for Locale-specific numeber formats) or new DecimalFormat(pattern).format(number) (this method can be used for user-defined patterns) Proposed Solution in case of Non or Partial Existence:	
Name: ParagraphFormat Description: Returns *string* with converted single newline characters (CR/LF sequences) into spaces and double newline characters into HTML paragraph markers (<P>).	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: A method can be written that parses the given text that performs the same function as ParagraphFormat.	H
Name: TimeFormat Description: Returns a custom-formatted time value. If no mask is specified, the TimeFormat function returns time value using the *hh:mm tt* format. Syntax : TimeFormat(time [, mask]) time - Any date/time value or string convertible to a time value. mask - A set of masking characters determining the format: * h -- Hours with no leading zero for single-digit hours. (Uses a 12-hour clock.) hh -- Hours with a leading zero for single-digit hours. (Uses a 12-hour clock.) H -- Hours with no leading zero for single-digit hours. (Uses a 24-hour clock.) HH -- Hours with a leading zero for single-digit hours. (Uses a 24-hour clock.) m -- Minutes with no leading zero for single-digit minutes mm -- Minutes with a leading zero for single-digit minutes s -- Seconds with no leading zero for single-digit seconds ss -- Seconds with a leading zero for single-digit seconds t -- Single-character time marker string, such as A or P tt -- Multiple-character time marker string, such as AM or PM When passing a date/time value	Exist (Yes/No/Partial): Yes Class: java.text.SimpleDateFormat and java.text.DateFormat Method: format() in combinaion with getTimeInstance() or getInstance()of java.text.DateFormat class. Proposed Solution in case of Non or Partial Existence:	L

as a string, make sure it is enclosed in quotes. Otherwise, it is interpreted as a number representation of a date/time object, returning undesired results		
Name: YesNoFormat Description:Returns Boolean data as YES or NO.The YesNoFormat function returns all non-zero values as YES and zero values as NO. Syntax : YesNoFormat(*value*) *value*-Any number or Boolean value.	Exist (Yes/No/Partial):Partial Class: java.lang.Boolean Method: Boolean.valueOf() This method takes only a *String*.If the String argument passes is "true" then it returns boolean true or else false. Proposed Solution in case of Non or Partial Existence:	M
Name: LSTimeFormat Description: Returns a custom-formatted time value using the locale convention. Syntax : TimeFormat(time [, mask]) (same as TimeFormat) time - Any date/time value or string convertible to a time value. mask - A set of masking characters determining the format.	Exist (Yes/No/Partial): Yes Class: java.text.SimpleDateFormat and java.text.DateFormat Method: format() in combinaion with getTimeInstance() or getInstance()of java.text.DateFormat class .The methods getTimeInstance() or getInstance() also take Locale as one of the parameters. Proposed Solution in case of Non or Partial Existence:	L
Name:HTMLEditFormat Description:Returns HTML escaped *string*. All carriage returns are removed from *string*, and all special characters (> < " &) are escaped.	Exist (Yes/No/Partial):No Class: Method: Proposed Solution in case of Non or Partial Existence: To do this the JVM should recognoize the HTHL tags and Java API does not provide any mechanism to parse the HTML tags.	H

Issues :
HTMLEditFormat & HTMLCodeFormat - These fuctions return the HTML escaped strings.The former encloses the returned string with <PER></PRE> tags while the later just returns the HTML escaped string.And Java API does not provide any mechanism to parse the HTML tags.We need to see the existing parsers (for xml) and have to develop an equivalent in java which is considerably non-simple and time taking.

LSEuroCurrencyFormat – Current Java API (JDK1.2) doesn't recognize Euro currency.

5.2.11 Category: Decision Functions

Total CFML Functions = 17
Available Java Functions = 13(Including Partial)
Complexity Level(Low\Medium\High)- 12L / 1M / 4H / 0C

CFML Functions	JAVA API's	Complexity
Name:IsArray Description: Returns TRUE if value is an array. Syntax : IsArray(value [, number]) Value : Variable name or array name. Number : Tests if the array has exactly the specified dimension.	Exist (Yes/No/Partial): Yes Class: java.lang.Class Method: public boolean isArray() Determines if this Class object represents an array class. Returns: true if this object represents an array class; false otherwise. OR, int a[]={1,2,3,4}; System.out.println(a instanceof int[]); System.out.println(a.length); for(int i=0;i<a.length;i++){ System.out.println(a[i]); Proposed Solution in case of Non or Partial Existence:	L

Name:IsAuthenticated
 	
Description:
Returns TRUE if the user has been authenticated for any ColdFusion security context. If you specify the name of the security context, IsAuthenticated returns TRUE if the user has been authenticated for the specified ColdFusion security context.
 	
Syntax :IsAuthenticated([security-context-name])
 	
Security-context-name: The security context name.
	Exist (Yes/No/Partial): Yes
	Class: javax.servlet.http
	Interface: HttpSession
	Method: getSession()
	We can achive the required functionality with following custom code.HttpSession's getValue (Constant.AUTHENTIFICATION) and putValue(Constant.AUTHENTIFICATION) can be used.
	Syntax:
	HttpSession session = request.getSession(false);
	String requestedPage = request.getParameter(Constants.REQUEST);
	if (session != null)
	{
	Boolean isAuthenticated = (Boolean) session.getValue(Constants.AUTHENTICATION);
	if (!isAuthenticated.booleanValue())
	{
	unauthenticatedUser(response, requestedPage);
	}
	}
	else
	{
	unauthenticatedUser(response, requestedPage);
	}
	}
	
	Proposed Solution in case of Non or Partial Existence:
		L

Name:IsAuthorized Description: Returns TRUE if the user is authorized to perform the specified action on the specified ColdFusion resource. Syntax: IsAuthorized(resourcetype, resourcename [, action])	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: No such method is there in java. This is a runtime verification and Java does not provide any property or method that will check whether the authorized is performed or not.	H
Name:IsBinary Description: Returns TRUE if value is binary; otherwise, the function returns FALSE.	Exist (Yes/No/Partial): Yes Class: java.lang.Integer Method: public static int parseInt(String s,int binary_value)throws NumberFormatException Proposed Solution in case of Non or Partial Existence:	L

Syntax : IsBinary(value) value : Any value.		
Name:IsDate Description: Returns TRUE if string can be converted to a date/time value; otherwise, FALSE. Note that ColdFusion converts the Boolean return value to its string equivalent, "Yes" and "No." Syntax : IsDate(string) string : Any string value.	Exist (Yes/No/Partial):Yes Class: java.util.Date; java.sql.Date; Method: toString() Class: java.text.DateFormat Method: public abstract StringBuffer format(Date date,StringBuffer toAppendTo,FieldPosition fieldPosition) Formats a Date into a date/time string. Proposed Solution in case of Non or Partial Existence:	L
Name:IsDebugMode Description: Returns TRUE if debugging mode was set via the ColdFusion Administrator and FALSE if debugging mode is disabled. Syntax : IsDebugMode()	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: No such method is available in java. Java API does not provide any mechanism to determine whether debugging is enabled or disabled.	H
Name:IsDefined Description: Evaluates a string value to determine if the variable named in the string value exists. IsDefined returns TRUE if the specified variable is found, FALSE if not found. IsDefined provides an alternative to the ParameterExists function, eliminating the need for cumbersome expressions used to test for the existence of a variable: Evaluate("ParameterExists(#var_name#)") Syntax : IsDefined("variable_name")	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Class: java.lang.reflect.Field Method: getName() Returns the name of the field represented by this Field object. Syntax: public String getName()	H

Name:IsLeapYear
 	
Description:
Returns TRUE if the year is a leap year; otherwise, FALSE.
Syntax : IsLeapYear(year)
year : Number representing the year.
	Exist (Yes/No/Partial): Yes
	Class: java.util.GregorianCalendar
	Method: public boolean isLeapYear(int year)
	OR,
	for(int i=0;i<3000;i++){
	if(i%4==0)
	System.out.println(i +"is a Leap year");
	Else
	System.out.println(i+" is not a Leap year");
	}
	
	Proposed Solution in case of Non or Partial Existence:
		L

Name:IsNumeric Description: Returns TRUE if string can be converted to a number; otherwise, FALSE. Syntax : IsNumeric(string) string : Any string value.	Exist (Yes/No/Partial): Yes Class: java.lang.Integer; Method: parseInt(String s) Parses the string argument as a signed decimal integer. Syntax: public static int parseInt(String s)throws NumberFormatException Parses the string argument as a signed decimal integer. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' ('\u002d') to indicate a negative value. The resulting integer value is returned, exactly as if the argument and the radix 10	L

	were given as arguments to the parseInt(java.lang.String, int) method. Returns: the integer represented by the argument in decimal. Throws: NumberFormatException - if the string does not contain a parsable integer.	
Name:IsNumericDate Description: Evaluates "real value" of date/time object. Returns TRUE if the number represents "real value" of the date/time object; otherwise, FALSE. Syntax : IsNumericDate(number) number : Real number.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Class: java.text.DateFormat Method: public Date parse(String text)throws ParseException Parse a date/time string. Parameters: text - The date/time string to be parsed Returns: A Date, or null if the input could not be parsed Throws: ParseException - If the given string cannot be parsed as a date.	M
Name:IsProtected Description: Returns TRUE if the resource is protected in the security context of the authenticated user. Syntax : IsProtected(resourcetype, resourcename [, action])	Exist (Yes/No/Partial): Yes Class: java.security.GuardedObject Method: public Object getObject()throws SecurityException Syntax: public class GuardedObject extends Object implements Serializable A GuardedObject is an object that is used to protect access to another object. A GuardedObject encapsulates a target object and a Guard object, such that access to the target object is possible only if the Guard object allows it. Once an object is encapsulated by a GuardedObject, access to that object is controlled by the getObject method, which invokes the checkGuard method on the Guard object that is guarding access. If access is not allowed, an exception is thrown. Proposed Solution in case of Non or Partial Existence:	L

Name:IsSimpleValue
 	
Description:
Returns TRUE if value is a string, number, Boolean, or date/time value.
 	
Syntax : IsSimpleValue(value)
value : Variable or expression.
	Exist (Yes/No/Partial): Yes
	Class: java.sql.*;
	Interface: ResultSet
	Method: executeQuary("Select * from tableName")
	getString("...")
	Syntax:
	Class.forName("Driver");
	Connection con=DriverManager.getConnection("jdbc:odbc:DSN,"login name","password");
	Statement st=con.creatStatement();
	ResultSet rs=st.executeQuary("Select * from tableName")
	while(rs.next()){
	rs.getString("Emp_ID");
	rs.getString("FirstName");
		L

	rs.getString("LastName");
	}
	
	Proposed Solution in case of Non or Partial Existence:	

Name:IsStruct
 	
Description:
Returns TRUE if variable is a structure.
Syntax : IsStruct(variable)
variable : Variable name.
	Exist (Yes/No/Partial): Yes
	Class: java.sql.*;
	Interface: ResultSet
	Method: executeQuary("Insert into tableName")
	Syntax:
	Class.forName("Driver");
	Connection con=DriverManager.getConnection("jdbc:
	odbc:DSN,"login name","password");
	PreparedStatement st=con.prepareStatement();
	while(rs.next()){
	rs.setString(1,String);
	rs.setInt(2,"int value");
	rs.setFloat(3,"float value");
	}
	ResultSet rs=ps. ExecuteQuary()
	whule(rs.next()){.....}
	Proposed Solution in case of Non or Partial Existence:
	Class: java.sql
	Interface: public abstract interface Struct
	It has 3 methods:
	getAttributes(): Produces the ordered values of the attributes of the SQL structurec type that this Struct object represents.
	
	getAttributes(Map map): Produces the ordered values of the attributes of the SQL structurec type that this Struct object represents.
	
	getSQLTypeName():Retrieves the SQL type name of the SQL structured type that this Struct object represents.
		L

Name:LSIsCurrency Description: Checks whether a string is a locale-specific currency string. Returns TRUE if *string* is a currency string, FALSE otherwise. 5.2.11.1.1 Syntax :LSIsCurrency(string) string : The locale-specific currency string.	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: NumberFormat.getInstance().format(number) (this method can be used for Locale-specific numeber formats) or new DecimalFormat(pattern).foramt(number) (this method can be used for user-defined patterns) Proposed Solution in case of Non or Partial Existence:	L
Name:LSIsdate Description: Like the IsDate function, LSIsDate returns TRUE if string can be converted to a date/time value in the current locale, FALSE otherwise. Syntax : LSIsDate(string) string : Any string value. Usage : Years less than 100 are interpreted as	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: NumberFormat.getInstance().format(number) (this method can be used for Locale-specific numeber formats) or new DecimalFormat(pattern).foramt(number) (this method can be used for user-defined patterns) Proposed Solution in case of Non or Partial Existence:	L

20th century values.		
Name:LSIsNumeric Description: Like the IsNumeric function, LSIsNumeric returns TRUE if string can be converted to a number in the current locale; otherwise, FALSE. Syntax : LSIsNumeric(string) string : Any string value.	Exist (Yes/No/Partial): Yes Class: java.text.NumberFormat Method: public Number parse(String text)throws ParseException Throws: ParseException - if the specified string is invalid. Proposed Solution in case of Non or Partial Existence.	L
Name:ParameterExist Description: Returns True if the specified parameter has been passed to the current template or has already been created during execution of the current template. Otherwise returns NO. This function is provided for backward compatibility with previous versions of ColdFusion. You should use the function IsDefined instead. Syntax : ParameterExists(parameter) parameter : Any syntactically valid parameter name.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: This is a cold fusion special function for cold fusion. No such speaial method is in Java.	H

Issues:

Most of the functions are available are in java. Out of seventeen , therteen are available(including partial)in java. five functions are not avilable in java.These are IsAuthenticated, IsAuthorized, , IsDebugMode, IsDefined & ParameterExist.These functions are related to runtime attributes and Java API doesn't provide any mechanism to determine these values. For this functions new method should be written in Java.

5.2.12 Category: Date & Time Functions

Total CFML Functions = 32
Available Java Functions = 24 (Including partially matching functions)
Complexity Level (Low/Medium/High) – 8L / 17M / 7H / 0C

CFML Functions	JAVA API's	Complexity
Name: CreateDate Description: Returns a valid date/time object.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Contruct an instance of java.util.GregorianCalendar with the constructor GregorianCalendar(int year, int month, int date). Later getTime() method can be used to get an instance of java.util.Date if required.	M
Name: CreateDateTime Description:	Exist (Yes/No/Partial): Partial Class: Method:	M

Returns a valid date/time object.	Proposed Solution in case of Non or Partial Existence: Contruct an instance of java.util.GregorianCalendar with the constructor GregorianCalendar(int year, int month, int date, int hour, int minute, int second) . Later getTime() method can be used to get an instance of java.util.Date if required.	
Name: CreateODBCDate Description: Returns a date in ODBC date format.	Exist (Yes/No/Partial): Partial Class: java.sql.TimeStamp Method: construtor TimeStamp(int year, int month, int date, int hour, int minute, int second, int nano) or construtor TimeStamp (long time) Proposed Solution in case of Non or Partial Existence:	M
Name: CreateODBCDateTime Description: Returns a date/time object in ODBC timestamp format.	Exist (Yes/No/Partial): Partial Class: java.sql.TimeStamp Method: construtor TimeStamp(int year, int month, int date, int hour, int minute, int second, int nano) or construtor TimeStamp (long time) Proposed Solution in case of Non or Partial Existence:	M
Name: CreateODBCTime Description: Returns a time object in ODBC time format.	Exist (Yes/No/Partial): Partial Class: java.sql.Time Method: construtor Time(int hour, int minute, int second) Proposed Solution in case of Non or Partial Existence:	M
Name: CreateTime Description: CreateTime(*hour*, *minute*, *second*) The date portion of time is set to December 30, 1899	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Contruct an instance of java.util.GregorianCalendar with the constructor GregorianCalendar(int year, int month, int date, int hour, int minute, int second) . Pass 1899, 11, 29 to the first three parameters. Later use the getTime() method to get an instance of java.util.Date	M
Name: CreateTimeSpan Description: Creates a date/time object for adding and subtracting other date/time objects.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Arithmetic on date objects can be done using the add(int field, int amount) method of java.util.GregorianCalendar.	M
Name: DateCompare Description: Performs a full date/time comparison of two dates. Returns -1 if *date1* is less than *date2*; returns 0 if *date1* is equal to *date2*; returns 1 if *date1* is greater than *date2*. See the description of *datePart* for information on specifying the precision of	Exist (Yes/No/Partial): Partial Class: java.util.Date Method: int compareTo(Date anotherDate) Proposed Solution in case of Non or Partial Existence: This method returns a integer which could be anything less than or greater that zero, unlike the Cold Fusion function which returns –1 or +1 when the dates arent equal.	H

the comparison.	The precision of comparision cannot be specifed in case of java. A wrapper function could be written to implement this feature.	
Name: DateConvert Description: Converts local time to Universal Coordinated Time (UTC) or UTC to local time based on the specified parameters. This function uses the daylight savings settings in the executing machine to compute daylight savings time, if required.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: A new method should be written to implement this functionality. Classes GregorianCalendar and TimeZone can be used to convert the local time to utc and viceversa	H
Name: DateDiff Description: Returns the number of intervals in whole units of type *Datepart* by which *Date1* is less than *Date2*.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: A new method should be written to implement this functionality. The date value can be converted to milliseconds form and differences between the two dates can be computed. The difference can be returned in the units that is specified.	H
Name: DateFormat Description: Returns a formatted date/time value. If no mask is specified, DateFormat function returns date value using the *dd-mmm-yy* format.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Use java.text.SimpleDateFormat class. Map the various patterns of Cold Fusion to the patterns available in SimpleDateFormat. Specify the required format using the applyPattern() method. Then on calling the format(), we get the date in a string form in the specified format.	H
Name: DatePart Description: Returns the specified part of a date as an integer.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Create an instance of GregorianCalendar passing the date object. Pass appropriate constants to get the corresponding parts of the date to the get(int part) method.	M
Name: Day Description: Returns the ordinal for the day of the month, ranging from 1 to 31.	Exist (Yes/No/Partial): Yes Class: java.util.GregorianCalendar Method: get(int field) Pass Calendar.DAY_OF_MONTH For this method Proposed Solution in case of Non or Partial Existence:	M
Name: DayOfWeek Description: Returns the ordinal for the day of the week. The day is given as an integer	Exist (Yes/No/Partial): Partial Class: java.util.GregorianCalendar Method: int get(int field) Proposed Solution in case of Non or Partial Existence:	M

ranging from 1 (Sunday) to 7 (Saturday).	The int parameter field should be java.util.Calendar.DAY_OF_WEEK 1 should be added to the returned integer.	
Name: DayOfWeekAsString Description: Returns the day of the week corresponding to *day_of_week*, an integer ranging from 1 (Sunday) to 7 (Saturday).	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Return an appropriate String depending on the value returned by java.util.GregroianCalendar.get(java.util.Calendar.DAY_OF_WEEK).	M
Name: DayOfYear Description: Returns the ordinal for the day of the year.	Exist (Yes/No/Partial): Partial Class: java.util.GregorianCalendar Method: int get(int field) Proposed Solution in case of Non or Partial Existence: The int parameter should be java.util.Calendar.DAY_OF_YEAR in the above class.	M
Name: DaysInMonth Description: Returns the number of days in the specified month (*Date*).	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Create an instance of java.util.GregorianCalendar by using the Date value. Then use the getActualMaximum(Calendar.MONTH) to get the number of days in that particular month	M
Name: DaysInYear Description: Returns the number of days in a year	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Create an instance of java.util.GregorianCalendar using that particular date. Then use the getActualMaximum(Calendar.DAY_OF_YEAR) to get the number of days in that particular year.	M
Name: FirstDayOfMonth Description: Returns the ordinal (the day's number in the year) for the first day of the specified month.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Get the current month and from this find the day of the year for the first day of the month.	H
Name: GetTimeZoneInfo Description: Returns a structure containing time zone information for the machine on which this function is executed	Exist (Yes/No/Partial): Yes Class: java.util.GregorianCalendar Method: getTimeZone() Proposed Solution in case of Non or Partial Existence:	L

Name: Hour Description: Returns the ordinal value for the hour, ranging from 0 to 23.	Exist (Yes/No/Partial): Yes Class: java.util.GregorianCalendar Method: get(int field) Pass Calendar.HOUR for this method Proposed Solution in case of Non or Partial Existence:	L
Name: IsDate Description: Returns TRUE if *string* can be converted to a date/time value; otherwise, FALSE. Note that ColdFusion converts the Boolean return value to its string equivalent, "Yes" and "No."	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Use parse() method of java.text.SimpleDateFormat to parse the given string. This returns a date object, if it is possible to parse the string. The method throws an ParseException if the string cannot be parsed. A wrapper function can be used to return appropriate boolean values	M
Name: IsLeapYear Description: Returns TRUE if the *year* is a leap year; otherwise, FALSE.	Exist (Yes/No/Partial): Yes Class: java.util.GrergorianCalendar Method: boolean isLeapYear(int year) Proposed Solution in case of Non or Partial Existence:	L
Name: IsNumericDate Description: Evaluates "real value" of date/time object. Returns TRUE if the number represents "real value" of the date/time object; otherwise, FALSE.	Exist (Yes/No/Partial): Partial Class: java.text.DateFormat Method: Date parse(String dateString) Proposed Solution in case of Non or Partial Existence: The DateFormat.parse(), method returns a reference to Date object on successfull parsing and it throws an exception if the string couldnt be parsed to a date. A wrapper function should be used to return boolean values.	M
Name: Minute Description: Returns the ordinal for the minute, ranging from 0 to 59.	Exist (Yes/No/Partial): Yes Class: java.util.GregorianCalendar Method: get(int field) Pass Calendar.MINUTE for this method Proposed Solution in case of Non or Partial Existence:	L
Name: Month Description: Returns the ordinal for the month, ranging from 1 (January) to 12 (December).	Exist (Yes/No/Partial): Yes Class: java.util.GregorianCalendar Method: get(int field) Pass Calendar.MONTH for this method Proposed Solution in case of Non or Partial Existence:	L
Name: MonthAsString Description: Returns the name of the month corresponding to *month_number*	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence:	M

	Pass the month number to the function. Then map the integer value to a string Value in a switch statement.	
Name: Now Description: Returns the current date and time as a valid date time object.	Exist (Yes/No/Partial): Yes Class: java.util.Date Method: Constructor Date() Proposed Solution in case of Non or Partial Existence:	L
Name: ParseDateTime Description: Returns a date/time object from a string.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence:We can Use parse() method of java.text.SimpleDateFormat. This throws an exception, if the string passed couldnot be parsed.We can almost achieve the same functionality as of the cfml function.	H
Name: Quarter Description: Returns the number of the quarter, an integer ranging from 1 to 4.	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Get the Month of the date and calculate the quarter using this.A customised logic can be developed for this purpose achieve he functionality.	M
Name: Second Description: For a date/time value, returns the ordinal for the second, an integer from 0 to 59.	Exist (Yes/No/Partial): Yes Class: java.util.GregorianCalendar Method: get(int field) Pass Calendar.SECOND for this method Proposed Solution in case of Non or Partial Existence:	L
Name: Week Description: Returns the ordinal for the week number in a year; an integer ranging from 1 to 53.	Exist (Yes/No/Partial): Yes Class: java.util.GregorianCalendar Method: get(int field) Pass Calendar.WEEK_OF_YEAR for this method Proposed Solution in case of Non or Partial Existence:	L
Name:XMLFormat Description: Returns a string that is safe to use with XML.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: A function can be written which escapes any xml special characters, if they are present in the Date string,so that it becomes XML compatible.	H

Note :
Most of the functionality of the Date and Time functions can be achived using the Date and GregorianCalendar classes of the java.util package.
In case of Cold Fusion Date and Time functions, whenever there is a number representing a year, then the following convention must be followed. This number could be in the range 100-9999. Years from 0 to 29 are interpreted as 21st

century values. Years 30 to 99 are interpreted as 20th century values.There lies a possibility of a sort of hardle,maynot be of greater magnitude.

5.2.13 Category: Authentication Functions

Total CFML Functions = 4
Available Java Functions = 4(Including Partial)
Complexity Level(Low\Medium\High)- 3L / 1M / 0H / 0C

CFML Functions	JAVA API's	Complexity
Name: AuthenticatedContext Description: Returns the name of the security context . Syntax : AuthenticatedContext()	Exist (Yes/No/Partial): Partial Class: Method: Proposed Solution in case of Non or Partial Existence: Class: java.security.Security Method: public static String getProperty(String key) Gets a security property value. First, if there is a security manager, its checkPermission method is called with a java.security.SecurityPermission ("getProperty."+key) permission to see if it's ok to retrieve the specified security property value. Parameters: key – the key of the property being retrieved. Returns: the value of the security property corresponding to key. Throws: SecurityException - if a security manager exists and its SecurityManager.checkPermission(java.security.Permission) method denies access to retrieve the specified security property value Class: java.security.Permission Method: public final String GetName() Returns the name of this Permission. For example, in the case of a java.io.FilePermission, the name will be a pathname. Returns: the name of this Permission.	M
Name: AuthenticatedUser Description: Returns the name of the authenticated user. Syntax : AuthenticatedUser()	Exist (Yes/No/Partial): Yes Class: Interface : HttpServletRequest Method: getRemoteUser() Returns the name of the user making this request, if the user has logged in using HTTP authentication. Syntax: Public abstract interface HttpServletRequest extends ServletRequest Public java.lang.String getRemoteUser() Proposed Solution in case of Non or Partial Existence:	L
Name: IsAuthenticated Description: Returns TRUE if the user has been authenticated for any	Exist (Yes/No/Partial): Yes Class: javax.servlet.http Interface: HttpSession Method: getSession()	L

ColdFusion security context. If you specify the name of the security context, IsAuthenticated returns TRUE if the user has been authenticated for the specified ColdFusion security context.
Syntax : IsAuthenticated ([security-context-name])
	We can achive the required functionality with following custom code.HttpSession's getValue (Constant.AUTHENTIFICATION) and putValue(Constant.AUTHENTIFICATION) can be used.
	Syntax:
	HttpSession session = request.getSession(false);
	String requestedPage = request.getParameter(Constants.REQUEST);
	if (session != null)
	{
	Boolean isAuthenticated = (Boolean) session.getValue(Constants.AUTHENTICATION);
	if (!isAuthenticated.booleanValue())
	{
	unauthenticatedUser(response, requestedPage);
	}
	}
	else
	{
	unauthenticatedUser(response, requestedPage);
	}
	}
	
	Proposed Solution in case of Non or Partial Existence:	

Name:IsAuthorized Description Returns TRUE if the user is authorized to perform the specified action on the specified ColdFusion resource. Syntex : IsAuthorized (resourcetype, resourcename [, action]) resourcetype: String specifying the type of resource: Application ,CFML ,File ,DataSource Component ,Collection ,CustomTag UserObject	Exist (Yes/No/Partial): Yes Class: com.netscape.server.servlet.extension Method: isAuthorized( ) Checks whether the current user has a specified permission. Interface: HttpSession2 interface Syntax: isAuthorized( ) Checks whether the current user has a specified permission. public abstract boolean isAuthorized(String acl,String permission) acl. The access control list in which to check for the permission. permission. The permission to check for. Rule : Before calling isAuthorized( ), the application must create a session. The user must also be logged in with loginSession( ). Return Value : Returns true if the authorization check succeeds; otherwise, returns false. Proposed Solution in case of Non or Partial Existence:	L

Issues:

All the functions are available are in java. Out of four, four are available(including partial)in java .

5.2.14 Category: System Functions

Total CFML Functions = 10
Available Java Functions = 7(including partial matches)
Complexity Level (Low/Medium/High/Critical) – 3L / 3M / 3H / 1C

CFML Functions	JAVA API's	Complexity Level
Name: DirectoryExists Description: Returns YES if the directory specified in the argument does exist; otherwise, it returns NO.	Exist (Yes/No/Partial): Yes Class: java.io.File Method: public boolean sDirectory() Description: Tests whether the file denoted by this abstract pathname is a directory.retuns true if adirectory,false otherwise. Proposed Solution in case of Non or Partial Existence:	L
Name: GetFileFromPath Description: Extracts the filename from a fully specified path.	Exist (Yes/No/Partial): Yes Class: Class:java.net.URL Method: public String getFile() Description:Get the file name of the URL. javax.swing.plaf.metal.MetalFileChooserUI Method: public String getFileName() Description:Get the file name from the path. Proposed Solution in case of Non or Partial Existence:	L
Name: ExpandPath Description: Returns a path equivalent to the *relative_path* appended to the base template path. ExpandPath creates a platform-appropriate path	Exist (Yes/No/Partial): No Class: Description: Proposed Solution in case of Non or Partial Existence: This is a typical CFML Application specific function which has no present implementation in contemporary java API.However we have to write methods to extract the Parent directory name fro a fully qualified pathname and append to the relative path (starts fro the appserver root directory)of the same.	H
Name: GetMetricData Description: On Windows NT, GetMetricData returns all the internal data that is otherwise displayed in the Windows NT PerfMonitor.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence:Some method which can get details in a structured format from windows nt perf monitor should be written to achieve the same functionality.	Critical
Name: FileExists Description: Returns YES if the file specified in the argument does exist; otherwise, it	Exist (Yes/No/Partial): Yes Class: java.io.File Method: public boolean isFile() Description :Tests whether the file denoted by this	L

returns NO	abstract pathname is a normal file Proposed Solution in case of Non or Partial Existence:	
Name: GetTempFile Description: Creates and returns the name of a temporary file in a directory whose name starts with (at most) the first three characters of *prefix*.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence: Exploring Java2 API's getPath() methods and moderate edition can be done	High
Name: GetCurrentTemplatePath Description: Returns the fully specified path of the template containing the call to this function.	Exist (Yes/No/Partial): No Class: java.io.File Method: Proposed Solution in case of Non or Partial Existence: Exploring Java2 API's getPath() methods and moderate edition can be done	M
Name: GetTemplatePath Description: Returns the fully specified path of the base template.	Exist (Yes/No/Partial): No Class: Description: Proposed Solution in case of Non or Partial Existence:Exploring Java2 API's getPath() methods and moderate edition can be done.	M
Name: GetDirectoryFromPath Description: Extracts the directory (with a \ (backslash)) from a fully specified path.	Exist (Yes/No/Partial): No Class: Method: Proposed Solution in case of Non or Partial Existence:No equivalent method is there in the java API at present and a new method has to be written .A look into the Swing.plaf package can be done.	H
Name: SetProfileString Description: Sets the value of a profile entry in an initialization file. This function returns an empty string if the operation succeeds or an error message if the operation fails.	Exist (Yes/No/Partial): No Class: javax.servlet.HttpServletConfig Method:public void init() Proposed Solution in case of Non or Partial Existence:At present support is not availale in Java2 API. Servlet API's Init(),getInitParameters() methods can be looked at to draw a possible way of action.	H

ISSUES/NOTES:

GetMetricData() returns data of the windows performance monitor ,we do not have any mapping for it in java.to extract this system level data we need to new function.
To deal with Function like SetProfileString() where we need to update the profile in an initialization file we have to write scripts which interact and updates the profile of a component.

GetTempFile()-could not be clear about the exact format in which we get the name.

6 Appendix C - Cold Fusion to J2EE operator mappings

	CF Operator	J2EE Equivalent	Comments
1.	&	+	String concatenation
2.	MOD	%	Modulus
3.	^	Pow(base, power)	Exponentiation
4.	\	/	Integer division
5.	IS EQUAL EQ	==	Equal-to
6.	IS NOT NOT EQUAL NEQ	!=	Not equal-to
7.	CONTAINS	indexOf Eg. X.indexOf(Y) != 0	Containment operator for strings
8.	DOES NOT CONTAIN	indexOf Eg. X.indexOf(Y) == 0	Opposite of CONTAINS
9.	GREATER THAN GT	>	Greater-than
10.	LESS THAN LT	<	Less-than
11.	GREATER THAN OR EQUAL TO GTE GE	>=	Greater-than or Equal-to
12.	LESS THAN OR EQUAL TO LTE LE	<=	Less-than or Equal-to
13.	TRUE	No Equivalent	May not be required. Eg. CFIF expression IS TRUE can be translated to IF (expression) in JAVA
14.	FALSE	No Equivalent	May not be required. Eg. CFIF expression IS FALSE can be translated to IF !(expression) in JAVA
15.	NOT	!	NOT
16.	AND	&&	Logical-AND
17.	OR	| |	Logical-OR
18.	XOR	^	Exclusive-OR
19.	EQV	!^	Exclusive-NOR
20.	IMP	Not available	Implies Eg. A IMP B Means IF A then B