

Contents

- [Enhanced Tags](#)

- [Application](#)
- [cfapplication](#)

- [Data & output](#)
- [cfdocument](#)
- [cfdbinfo](#)
- [cfdump](#)
- [cfhttp](#)
- [cfquery](#)
- [cfflush](#)
- [cfpdf](#)
- [cfselect](#)
- [cfsilent](#)

- [Extensibility](#)

- [cffunction](#)
- [cfimap](#)

- [File management](#)
- [cf file](#)
- [cfdirectory](#)

- [Flow control](#)
- [cfthread](#)
- [cfabort](#)
- [cflock](#)

- [Page processing](#)
- [cfcontent](#)
- [cfhtmlhead](#)
- [cfsetting](#)

Enhanced Tags

Here you will find a description of all tags existing in the CFML standard but which are enhanced in Railo with helpful features. All tags should work like the original definition in the CFML standard. Here only additional functionalities or attributes are described.

Application

This page contains all the application framework tags.

cfapplication

The CFAPPLICATION tag has three new attributes:

- mappings
- customtagpaths
- action

The first two are introduced in Railo 3.0 in order to allow a user the same functionality like Application.cfc which is defining per application mappings or customtags. So what you can do is something like this:

Example:

```
st = struct("myLocalMapping":"/opt/apache/htdocs/myMapping")>
lstCT = "/opt/custom1,/opt/custom">
... mappings="#st#" customtagpaths="#lstCT#"
```

The action attribute allows you to add things to your application context. Normally when you

want to change something in your application context (like timeoutsetting) you need to use all attributes in order to set everything correctly. The action accepts two different actions "update" and "create" which is the default value. The value "update" in the action attribute instructs Railo to replace the settings passed in the other attributes of the tag. So if you want to add a temporary per application mapping you can do that wherever you like:

Example:

```
action="update" mappings="#st#">
```

Data & output

This page contains a description of all changed tags regarding data retrieving and output. cfcache The tag cfcache has been enhanced with a certain functionality. Normally the tag caches a complete page depending on the contents of the url and form scope. In Railo 3.0 you can now use the opening tag CFCACHE with the closing one in order to cache parts of your output. So for instance if you want to cache the header of your application you can do this in the following way:

```
action="content" timespan="#createTimeSpan(0,0,15,0)#">
  "header.cfm">
```

This caches the output of the file header.cfm for 15 minutes. It will be stored as a file in the WEB-INF/railo/cfcache folder and will be read from this file for the next 15 minutes. In addition to the environment state (url, form etc.) you can use the "key" attribute to generate a key like you wish. This key could for instance be something like serialize(request). Please note this differs from CF9.

cfdocument

The popular tag (and) is available since Railo 2.0 and offers some improvements to the well known features. Next to the possibility to define a proxyserver and a corresponding proxyport, there are other attributes, which boost the flexibility of the tag. The attribute "src" and "srcfile" can not only convert text based contents into PDF, but even images and existing PDF documents. Converting PDF's into PDF's might at first glance not make a lot of sence, but the following example should show the options you have using this feature:

Example:

```
filename="ab.pdf">
srcfile="a.pdf" />
srcfile="b.pdf" />
```

This example for instance concatenates two PDF documents into one new, which contains the content of both PDF's.

Create your own index by using the H1 - H6 HTML tags. In addition in Railo 3.0 we introduced the boolean attribute htmlbookmark which converts the H1-H6 headlines into bookmarks of the document. This is a very helpful functionality.

Example:

```
type="application/pdf"><---
--> name="Content-Disposition" value="inline; filename=test.pdf"><---
--> format="pdf" bookmark="true" htmlbookmark="true">
```

Susi

My Susi description text

Peter

Funny

cfdbinfo

Next to the full CFML standard functionality Railo 3.0 supports the new type "users" which returns a list of all database users to the passed database.

cfdump

For the tag the new attribute "eval" was introduced, which can be used instead of the attribute "var".

The attribute "eval" receives a string as an entry, which is interpreted and used as a label for dump.

Example:

```
eval="structCount(form)">
```

is the same as

```
var="#structCount(form)#" label="structCount(form)">
```

Next to supporting all CF8 attributes, CFDUMP now supports the attribute abort="true", which instructs Railo to abort the request after the dump. You can even write it as in the example below:

Example:

```
abort eval="request">
```

In addition as a goodie, if you hover over the label of a dump output, Railo displays the code line and the file where the dump was produced:

cfhttp

This tag has been enhanced with the attribute "addtoken". When this attribute is set to "true", the request cookies "cfid, cftoken, jsessionid" will be added to the request so that a request will be automatically recognized being from the same session. This can be compared with the behaviour of the tag CFLOCATION.

Example:

```
addtoken="yes" url="http://www.railo.ch/en/index.cfm">
```

cfquery

The tag CFQuery comes with a new attribute called "PSQ", which contains a boolean value determining whether in the body of the CFQuery-tags single quotes automatically will be escaped or not. Valid values are TRUE or FALSE.

Example:

```
sql="select * from Users where name='susi's'">  
name="test" PSQ="false">#sql#
```

This can be installed globally by setting it in the Railo administrator under services / datasource.

Even though not documented, the CFML standard CFSTOREDPROC supports the following attributes:

- cachewithin
- cacheafter
- cachename

Railo 3.0 now supports them as well.

cfflush

The tag CFFLUSH can even be used inside a CFTHREAD tag in order to follow the process of a thread. The results in this case can of course interfere with the output generated outside the tag CFTHREAD

cfpdf

The tag CFPDF has a couple of enhancements in comparison to the CFML standard in order to offer more flexibility.

Railo offers you the additional attribute filter that gives you more flexibility when merging several PDF documents from a certain directory. It works exactly as the attribute filter of the tag CFDIRECTORY.

When deleting pages from a PDF document existing bookmarks are deleted as well when they point to a deleted page.

cfselect

The CFSELECT tag now offers the boolean attribute `casesensitive` which defines whether an option is selected based on a case sensitive matching of the value or not.

cfsilent

The tag `cfsilent` comes with a new attribute called `bufferoutput`. The reason for this is that Railo eventually might output the data generated inside a CFSILENT, since when having the following code:

```
This code is visible
```

An abort outputs the data generated so far even though it is encapsulated inside a `cf` tag. Not here comes the usecase for the new attribute. If you do specify the attribute `bufferOutput` and set it to `false`, then the output is not visible. Then Railo writes it into the dev null device. This is not only much faster, but it saves memory as well. We will add a radio button to the Railo Administrator which will allow to set the default behaviour of `bufferOutput` and `output="no"` (this will follow in one of the next patches) inside a `cffunction`. So the code like follows:

```
bufferOutput="false">
This code is not visible
```

Then the line "This code is not visible" will not be displayed. It is a different behaviour than in the CFML standard, but since you can use it or not and since in the Railo Administrator you will be able to define it's default behaviour to meet the standard, it is not considered as incompatible to the CFML standard.

Extensibility

This page contains all the changed tags regarding extensibility.

cffunction

The CFFUNCTION tag now supports the `returnformat="serialize"` attribute. This will return the value of the function formatted in a Railo serialized way (the same as if you would use the function `serialize()` in order to convert the return format).

Example:

```
name="susi" returntype="struct" returnformat="serialize">
```

cfimap

The tag CFIMAP works exactly as the tag CFMAIL and accepts the same attributes. It just uses the IMAP format for connecting to the mailserver.

File management

This page contains the changes of all file or folder related tags.

cffile

The tag `cffile` was enhanced with the action "info", which allows you to retrieve information about a single file, similar to the return value of `dir` for a whole directory. In addition to that the size (width and height) is shown if the target file is an image.

Example:

```
action="info" file="railo.jpg" variable="info">
eval="info">
```

Contents of the variable "info":

```
info.img.width: width of the Image
info.img.height: height of the image
info.attributes: attributes of the file
info.datelastmodified: last change
info.name: filename
info.size: size in bytes
```

Next to supporting resources, especially for the Amazon S3 resource we introduced a new attribute called `acl`. With this attribute you can set the authentication level of a certain file. It can be used like this:

Example:

```
action="copy" from="/opt/apache/htdocs/myfile.txt" destination="/s3/myfile.txt" acl="public-read-write|authenticated-read|public-read|private">
```

Where `/s3` is a mapping defined in the Railo administrator pointing to a Amazon S3 bucket.

cfdirectory

This tag supports resources as well. Especially for the Amazon S3 resource the `CFDIRECTORY` tag supports the new attribute `acl` as well. With this attribute you can set the authentication level of a certain directory. It can be used like this:

Example:

```
action="create" directory="/s3/mydir" acl="public-read-write|authenticated-read|public-read|private">
```

Where `/s3` is a mapping defined in the Railo administrator pointing to a Amazon S3 bucket.

Flow control

Description of all changed flow control tags.

cfthread

The new `CFTHREAD` tag supports the attribute `type`. This attribute can contain either the string `"thread"` which is the default or `"task"`. If you set the type to `"task"` then the code included within the `CFTHREAD` tag is executed in the task manager. In addition to the type attribute the `CFTHREAD` tag supports the attribute `retryinterval` which allows you to set how often a certain task will be repeated before it will be marked as non executable. Read more about the new task manager.

cfabort

This tag has been extended with the attribute `"type"`. With this attribute you can define whether a request is aborted completely (request) or if only the current CFML-template is aborted (page). The default value for this attribute is `"request"`.

Example: Template `index.cfm`

```
Hello
template="main.cfm">
#name#
```

Template `main.cfm`

```
name="Railo">
type="page">
message="your are to far">
```

Output:

```
Hello Railo
```

cflock

Since with the help of the attribute `"throwOnTimeout"` can be prevented that the tag throws an exception, the following return variables have been introduced in Railo: `cflock.succeeded`: Could the lock be applied `cflock.errorText`: Raised exception

Page processing

Description of all changed page processing tags.

cfcontent

The tag `CFCONTENT` introduces full support for download managers. Download managers can request parts of a download and resume or pause running downloads. By using the new attribute `"range"` one can influence the functionality of resumable downloads. The following values are possible:

- No - allows no partial downloads
- Yes - the client (download manager or browser) is notified that partially downloads are possible
- No value - the client will not be especially notified about the possibility of partial downloads. Nevertheless it is possible to resume paused downloads (most download managers try to download in parts anyway since you can always give it a try).

cfhtmlhead

```
<--- read data in html head ---->
action="read" variable="text">
<--- reset data defined for html head ---->
action="reset">
<--- append data to html head (action attribute is optional in this case) ---->
action="append" text="susi">
<--- write data to html head (overwrite existing data) ---->
action="write" text="susi">
```

cfsetting

The attribute "enableCfOutputOnly" takes, like before, next to the boolean value, a new value named "reset". The reason for this is that if you use the command enableCfOutputOnly="true" several times, you don't have to close the setting as many times. You can use "reset" in order to make sure that enableCfOutputOnly was closed.

Example:

```
enableCfOutputOnly="true">
enableCfOutputOnly="true">
enableCfOutputOnly="true">
enableCfOutputOnly="false">
<--- enableCfOutputOnly is still active ---->
Hello
enableCfOutputOnly="reset">
<--- now you can be sure it is deactivated ---->
Railo
```

Output:

Railo