

Email Tracking Made Easy

Build your own spam system

Lucas Sherwood



How do you do it today?

- **<CFMAIL> everywhere!**
- **Where do they come from?**
 - One common address
 - All Different
 - A dynamic Variable
 - No Idea??
- **How do you set your priorities?**

What do they Look Like?

- **Plain Text?**
- **Fancy HTML**
- **Constant or inconsistent?**
- **Is there any consistency at all?**

Things change, get over it!

- **Can you handle it?**
 - Re-branding
 - Templating



Is there a better way?

- **Centralised email handling**
- **Queuing and prioritising emails**
- **Templates**
- **Confirming emails have been sent**

The Main Queue concept

- **CFC/DB driven**
- **Scheduled task sends the mail**
- **FiFo - First in First out**
- **Send high priority items first**
- **Don't Send before X**
- **Templating**

Step 1 - The DB

- **Keep it simple**
- **Only store what you need to store**
 - Simple starting point...

Basic Email Fields

To
From
CC
BCC
ReplyTo
Subject
bodyText
BodyPlainText
bHTML

Tracking Fields

sequenceID
generatedBy
sendingDomain
dtSent
failCount
dtToSend
testMode



Step 2 - The CFC

- **Four methods is all that is needed**
 - Add Mail Item
 - This adds an email to the queue
 - Get Queued Email
 - Go and get get the next batch of emails to send
 - Send Email
 - Send each email, one at a time
 - Apply template
 - Wraps each email in a template

Add Mail Item method

```
<cffunction name="addMailItem" output="false" returntype="numeric" hint="this adds a mail item to the mail queue">
<cfargument name="emailTo" required="true" type="string">
<cfargument name="emailFrom" required="true" type="string">
<cfargument name="subject" required="true" type="string">
<cfargument name="bodyText" required="true" type="string">
<!-- to send date/Time - required but not enforced -->
<cfargument name="dtToSend" required="false" type="date" default="#now()#">

    <cfscript>
        // add key fields
        arguments.dtAdded = now();
    </cfscript>
    <cfquery name="qInsertMail" datasource="#application.dsn#">
    INSERT INTO emailQueue
    ( emailTo,emailFrom,subject,bodyText,dtToSend,dtAdded )
    VALUES
    (
    <cfqueryParam value="#arguments.emailTo#" cfsqltype="cf_sql_varchar">,
    <cfqueryParam value="#arguments.emailFrom#" cfsqltype="cf_sql_varchar">,
    <cfqueryParam value="#arguments.subject#" cfsqltype="cf_sql_varchar">,
    <cfqueryParam value="#arguments.bodyText#" cfsqltype="cf_sql_longvarchar">,

    <cfqueryParam value="#arguments.dtToSend#" cfsqltype="cf_sql_timestamp">,
    <cfqueryParam value="#arguments.dtAdded#" cfsqltype="cf_sql_timestamp">
    )
    SELECT @@IDENTITY as sequenceID
    </cfquery>
<cfreturn qInsertMail.sequenceID>
</cffunction>
```



Get Queued Mail

```
<cffunction name="getQueuedEmail" returntype="query" output="false">  
<cfargument name="rowCount" required="false" type="numeric" default="50">  
<cfargument name="maxFailCount" required="false" type="numeric" default="5">
```

```
    <cfquery name="qGetMails" datasource="#application.dsn#">  
        SELECT TOP #arguments.rowCount#  
        sequenceID,emailTo,emailFrom,subject,bodyText,dtToSend,dtAdded  
        FROM tabEmailQueue  
        WHERE 1=1  
        AND testMode = 0  
        AND dtSent is NULL  
        AND dtToSend <= GetDate()  
        AND failCount < <cfqueryParam value="#arguments.maxFailCount#" cfsqltype="cf_sql_integer">  
        ORDER BY dtToSend,sequenceID  
    </cfquery>
```

```
<cfreturn qGetMails>  
</cffunction>
```



Send Email method

```
<cffunction name="sendEmail" access="public" returntype="boolean">
  <cfset qGetMails = getQueuedEmail()>
  <cfloop query="qGetMails"><!-- loop through current batch -->
    <cftry><!-- begin email try -->
      <cfoutput>
        <cfset error = false>
        <cfmail to="#qGetMails.emailTo#"
              from="#qGetMails.emailFrom#"
              subject="#qGetMails.subject#">#qGetMails.bodyText#</cfmail>
      </cfoutput>
      <!-- catch result -->
      <cfcatch type="any">
        <cfset error = true>
        <cfset errmsg = cfcatch.message>
      </cfcatch>
    </cftry>
    <!-- log email into DB -->
    <cfif error>
      <cfquery name="qUpdateMail" datasource="#application.dsn#">
        Update tabEmailQueue
        SET failCount = <cfqueryParam value="#int(failCount+1)#">,
            dtToSend = #createODBCDateTime(dateAdd('h',1,now()))#,
            failmsg = <cfqueryParam value="#errmsg#" cfsqltype="cf_sql_varchar">
            WHERE sequenceID = #qGetMails.sequenceID#
      </cfquery>
    </cfif>
    <cfquery name="qUpdateMail" datasource="#application.dsn#">
      Update tabEmailQueue
      SET dtSent = #createODBCDateTime(now())#
      WHERE sequenceID = #qGetMails.sequenceID#
    </cfquery>
  </cfloop>
  <cfreturn true>
</cffunction>
```



Apply Template method

```
<cffunction name="parseEmail" returntype="string" output="false">
<cfargument name="emailBody" required="true" type="string">
<cfargument name="templateName" required="false" type="string">

<cfif isDefined("arguments.templateName") AND arguments.templateName neq "">
  <!-- we have a template, if this works then cool - if not, just return --->
  <cftry>
    <cfset filename =expandPath(arguments.templateName)>
    <cffile action="read" file="#filename#" variable="template">
    <cfcatch>
      <cfthrow message="error reading template file">
    </cfcatch>
  </cftry>
  <!-- if we are this far, then there was not an error! --->
  <cfset arguments.emailBody = Replace(template,'[BODY]',arguments.emailBody,'ALL')>
</cfif>
<cfreturn arguments.emailBody>
</cffunction>
```



Step 3 - The Tag

- **Abstract Add Mail Item() function**
- **Looks and works like CFMAIL**
- **Handles validation**

```
<util:mail to="lucas@redballoondays.com.au"  
  from="system@redballoondays.com.au"  
  subject="test">
```

your mail message here

```
</util:mail>
```



Step 4 - The scheduled task

- **Calls the mail agent**
 - Goes and gets the next email
 - Sends it (in interactive mode)
 - Use CFTRY/CFCATCH to see if it works
 - Log the result
 - And maybe retry the failed emails (to a limit)

```
<cfscript>  
    o = createObject("component","emailManager");  
    o.sendEmail();  
</cfscript>
```



Dodging spam filters

- **Set your mailerid**
 - It is “Macromedia ColdFusion” by default
- **Add missing headers (as per RFC2822)**
 - Reply-to
 - Message-id
- **Maybe consider adding SPF records to your DNS**

Taking it one step further

- **Confirming that your emails have been read**
 - Add a small gif to your email template
 - Make sure that this GIF is not a normal GIF
 - Update the mail item and flag that it has been opened

```

```

