ImapWatcher ColdFusion Event Gateway Documentation

This document provides instructions on the configuration and use of the ImapWatcher Event Gateway.

About the ImapWatcher Event Gateway

According to Wikipedia, The Internet Message Access Protocol (IMAP) is one of the two most prevalent Internet standard protocols for e-mail retrieval, the other being the Post Office Protocol (POP). Virtually all modern e-mail clients and mail servers support both protocols as a means of transferring e-mail messages from a server.

The ImapWatcher Event Gateway allows you to create a persistent connection to an IMAP mail server and wait for new messages to come in. As new messages arrive the ImapWatcher Event Gateway can notify your application.

Installation

Installation of the ImapWatcher Event Gateway is very easy. Simply take the ImapWatcherGateway.jar file and copy it under ColdFusion in the Gateway/lib directory. For example, on OS X running JRun, this path would be:

/Applications/JRun4/servers/cfusion/cfusion-ear/cfusion-war/WEB-INF/cfusion/gateway/lib

Your exact path will likely be different.

One you have copied the jar file restart ColdFusion and open up the ColdFusion administration interface. Go into Event Gateways > Gateway types and add this event gateway using these settings:

Type Name: ImapWatcherGateway

Description: Watches an IMAP server for new messages

Java Class: com.alagad.ImapWatcher.ImapWatcherGateway

Startup Timeout: 30 seconds

Click Add Type and the ImapWatcher Event Gateway is now installed.

Configuration

When you want to start using the ImapWatcher Event Gateway you will need to create a configuration file. An example file can be seen /config/ImapWatcher.cfg. This file has the following configurable values:

hostname – This is the IMAP host you are connecting to. For example, imap.gmail.com.
username – This is the username that you will be connecting to IMAP as.

password – This is the user's password.

folder – This is the folder the ImapWatcher Event Gateway will watch for messages. If you are unsure of the name of the folder you can enable logging and restart the gateway. This configuration will list all of the folder names on the IMAP server for you to choose from. For example: Inbox.

searchOption – The ImapWatcher Event Gateway will watch the folder you specify for messages. This configuration value controls what type of messages it will watch for. The options are "unread" or "all". Unread will force the Event Gateway to ignore any messages that have been read (including by the Event Gateway itself). An option of all will cause the Event Gateway to ignore no messages.

regex – One the Event Gateway has found messages based on the searchOption setting, you can further filter them down by making sure they match a regular expression. This regular expression provided will be matched against the subject, body text, and body html of emails. Any matches will trigger an event. Any non-matches will be ignored. Leave this blank or comment it out to avoid using regular expressions.

markAsRead – Indicates if a message should be marked as read once they have been found. Options are true or false.

postEventAction – Indicates what to do one an event has been raised. Options are move, delete or none. Move will move the file to the folder specified in the targetFolder setting. Delete will delete the message by putting it in the trash. None will do nothing with matched messages. (Note, if you set searchOptions to all, and postEventAction to none the EventGateway will repeatedly announce events.)

targetFolder – If postEventAction is set to move, this value indicates the folder to move matched messages to.

functionToCall – This is the function to call on CFC listeners.

Listener CFC

You can set any CFC to listen for events. To see an example listener CFC look in cfml/example.cfc. The CFC must have a function named the same as the functionToCall configuration value. This function must accept one argument named CFEvent of type Struct.

Creating a Gateway Instance

One you have a configuration file and a CFC, you can create an instance of the IMAP Event Gateway. To do this, open your ColdFusion Administrator and go into Event Gateways > Gateway Instances. Provide the following values into the Add ColdFusion Event Gateway Instances form:

Gateway ID: Any name you want. For example, My IMAP Watcher

Gateway Type: Select "ImapWatcherGateway - Watches an IMAP server for new messages"

CFC Path: The full path to your CFC

Configuration File: The full path to your configuration file.

Startup Mode: Automatic

Click Add Gateway Instance. Now you should be able to start the Event Gateway and begin handling events from it.

Handling Events

The CFC you write to handle events will receive one argument named CFEvent. This argument is a structure of data. There is a bunch of gateway-specific information in this structure, but what you will most care about is the EventMessage object which is found at "CFEvent.data.message".

The EventMessage object is a Java object that holds information about the message such as subject, sender, recipients, body, and attachments. You will use this object's API to access information from this message.

EventMessage API

The EventMessage object has the following public functions:

getSubject() – Returns the message subject.

getFrom() – Returns the email address of the sender. (Note this is wrapped in square brackets.)

getToList() – Returns a comma separated list of addresses the message was sent to. (Note, email addresses are wrapped in square brackets and there is a trailing comma). You can always use the listChangeDelims() ColdFusion function to replace and correct these extra characters.

getCCList() – Returns a comma separated list of addresses the message was CCed to. (Note, email addresses are wrapped in square brackets and there is a trailing comma). You can always use the listChangeDelims() ColdFusion function to replace and correct these extra characters.

getText() – Returns the text version of the email.

getHTML() – Returns the HTML version of the email. Note, not all messages will have this.

getAttachments() – Returns a Java ArrayList of Attachment objects on this message. You can use getAttachments().size() to see how many attachments are on a message. See Attachment API below.

Attachment API

Attachments are associated with an EventMessage by an ArrayList of Attachment object.

writeToDisk(directory) – Writes the attachment to disk in the specified directory using the attachment's file name.

writeToDisk(directory, file) – Writes the attachment to disk in the specified directory using the specified file name.

getContentType() – Returns the mime type of the attachment.

getFileName() – Returns the filename of the attachment.

byte[] = getContent() – Returns a Java byte array of the binary attachment content.