LiveCode Lessons (copy)

Enter a keyword or topic to search

Search

Last Updated
Nov 26, 2018
Print Article

Other Resources

Getting Started with LiveCode
Set Up and Running with LiveCode
Getting Started with LiveCode Development

LiveCode Lessons
How To - Step By Step Guides To Tasks In LiveCode
How To - LiveCode Server Tasks
How To - LiveCode Mobile Tasks
How To - LiveCode Scripts Scripts
How To - LiveCode Marketplace Products
How to Purchase and License LiveCode

Tutorials
Creating a Video Library Application

Data Grid
LiveCode Data Grid
Data Grid Tips & Tricks
Converting the Stock Program

LiveCode Releases
LiveCode 9.0
LiveCode 8.1
LiveCode 8

Comments
Lizard error: internal for this article

Sending Emails From LiveCode Server Scripts

A common task among web scripts is the sending of emails, be it for verifying user accounts or confirming actions. This lesson covers the sending of emails from LiveCode Server scripts.

Note: This lesson only applies on OSX and Linux servers.

Calling Sendmail From LiveCode

The easiest way to send emails from a LiveCode Server script is to shell out to a command line application. In this example, we will use sendmail. Sendmail can be called from the command line in the following manner:

```
[user@user ~]$ sendmail to@address.com -f from@address.com
```

Users can then input the email's subject and message body. We will create a LiveCode command that will wrap sendmail, calling it via the shell function, allowing users to send emails from LiveCode Server scripts. Our command will have the following prototype:

```
-- mail
--
-- Emails the given message to the recipients specified.
-- Each address passed can have a name attached in the form "name <address>".
-- Addresses can be passed as comma separated lists.
-- Attachments can be added by passing an array (integer indexed or otherwise),
-- with each attachment itself being an array.
--
-- pTo - The addresses to send the message to
-- pSubj - The message subject
-- pMsg - The message body
-- pFrom - The address of the message sender
-- pCc - Any cc addresses
-- pBcc - Any Bcc addresses
-- pHtml - Boolean, if the message is to be sent as html
-- pAtts - Array of all attachments to send, each attachment of the form:
--  "name: the name of the attachment
--  "path: the absolute path to the attachment
--  "type: the mime type of the attachment, defaults to
--  application/octet-stream
--
command mail pTo, pSubj, pMsg, pFrom, pCc, pBcc, pHtml, pAtts
```

First of all, we look at the actual calling of sendmail. As mentioned above, we will call it via the shell function. Instead of the user typing the mail message in after calling sendmail, we will pipe our message directly to sendmail.

```
put shell("echo" && wrapQShellEscape(pMsg) && "| /usr/sbin/sendmail" && wrapQShellEscape(pTo)) && "-f" && wrapQShellEscape(pFrom))
```

Note the use of new functions "wrapQ" and "shellEscape". These functions format strings for use in the shell function. "wrapQ" just pre and post fixes the passed string with quotation marks while "shellEscape" escapes any special characters.

```
-- escape shell characters: use this function before passing data to the shell
function shellEscape pText
  repeat for each char tChar in "\$ " & quote
    replace tChar with "\" & tChar in pText
  end repeat
  return pText
end shellEscape
-- wrap quotes around text
function wrapQ pText
  return quote & pText & quote
end wrapQ
```

Formatting Email Headers

So, our command now sends bare bones emails. In order for subject, from, to and cc parameters to work correctly, we must add the appropriate headers to our message. To do this, we just need to prefix our message with the following header data:

```
From: from@address.com
To: to@address.com
Cc: cc@address.com
Subject: Message Subject
```

We will do this by creating a new variable for placing the header and message data into:

```
local tMsg
put "From:" && pFrom & return & "To:" && pTo & return & "Subject:" && pSubj & return into tMsg
if pCc is not empty then
  put "Cc:" && pCc & return after tMsg
end if
put pMsg & return after tMsg
```

Note that the BCC addresses are not handled here. Instead, we just send a copy of the message to the BCC addresses:

```
if pBcc is not empty
  get shell("echo" && wrapQShellEscape(pMsg) && "| /usr/sbin/sendmail" && \
    wrapQShellEscape(pBcc) && "-f" && wrapQShellEscape(pFrom))
end if
```

The next parameter in our command's prototype is "pHtml". This will be a boolean value that will determine if the email is to be sent as plain text or html. To do this, we just add to our header (before we add the message body) the content type:

```
if pHtml is true then
  put "Content-Type: text/html;" & return & return after tMsg
else
  put "Content-Type: text/plain;" & return & return after tMsg
end if
```

Adding Attachments

Our final parameter handles any attachments. We intend attachments to be passed in the form of an array, with an element for each attachment. Each attachment will be an array itself, with three elements, the attachments name, path and mime type. An attachment array will take the following form:

```
put "/homefeed_file.txt" into tAtts[1]["path"]
put "text_file.txt" into tAtts[1]["name"]
put "text/plain" into tAtts[1]["type"]
put "/homefeed_file.pdf" into tAtts[2]["path"]
put "pdf_file.txt" into tAtts[2]["name"]
put "application/pdf" into tAtts[2]["type"]
```

For more information about mime type visit http://www.w3schools.com/tags/att_script_type.asp. In order to send these attachments, we must encode them out from the body of our email. We do this by defining the email as being multipart. That entails that the email body will have multiple sections, one being our message body, the remainder are attachments. Each section will be split by a unique boundary string. We do this by stating in our email will be multipart and defining our boundary string in the header:

```
if pAtts is an array then
  local tBoundary
  put "boundary" & the seconds into tBoundary
  put "MIME-Version: 1.0" & return & "Content-Type: multipart/mixed; boundary=" & \
    wrapQ(tBoundary) & return & "--" & tBoundary & return after tMsg
end if
```

Next we need to add our attachments to the message body. We do this by adding the base 64 encoded contents of the attachment on to the end of our message, remembering to separate each with our boundary string:

```
if pAtts is an array then
  put "--" & tBoundary & return after tMsg
  repeat for each element tAtt in pAtts
    put "Content-Type:" && tAtt["type"] & "; name=" & wrapQ(tAtt["name"]) & "; " & \
    return & "Content-Transfer-Encoding: base64;" & return & return & \
    base64Encode(URL ("binfile:" & tAtt["path"])) & return & "--" & tBoundary & \
    return after tMsg
  end repeat
end if
```

The Complete Command

So that's it, our command is now complete. With all our sections added in, it should look something like the following:

```
-- mail
--
-- Emails the given message to the recipients specified.
-- Each address passed can have a name attached in the form "name <address>".
-- Addresses can be passed as comma separated lists.
-- Attachments can be added by passing an array (integer indexed or otherwise),
-- with each attachment itself being an array.
--
-- pTo - The addresses to send the message to
-- pSubj - The message subject
-- pMsg - The message body
-- pFrom - The address of the message sender
-- pCc - Any cc addresses
-- pBcc - Any Bcc addresses
-- pHtml - Boolean, if the message is to be sent as html
-- pAtts - Array of all attachments to send, each attachment of the form:
--  "name: the name of the attachment
--  "path: the absolute path to the attachment
--  "type: the mime type of the attachment, defaults to
--  application/octet-stream
--
command mail pTo, pSubj, pMsg, pFrom, pCc, pBcc, pHtml, pAtts
  local tMsg
  -- build the message header, adding the from, to and subject details
  -- we also put any cc addresses in here, but not bcc (bcc addresses hidden)
  put "From:" && pFrom & return & "To:" && pTo & return & "Subject:" && pSubj & \
    return into tMsg & pCc is not empty then
    put "Cc:" && pCc & return after tMsg
  end if
  -- if there are any attachments, we must send this email as multipart
  -- with the message body and each attachment forming a part
  -- we do this by specifying the message as multipart and generating a unique boundary
  if pAtts is an array then
    local tBoundary
    put "boundary" & the seconds into tBoundary
    put "MIME-Version: 1.0" & return & "Content-Type: multipart/mixed; boundary=" & \
      wrapQ(tBoundary) & return & "--" & tBoundary & return after tMsg
    end if
  -- add the actual message body, setting the content type appropriatly
  if pHtml is true then
    put "Content-Type: text/html;" & return & return after tMsg
  else
    put "Content-Type: text/plain;" & return & return after tMsg
  end if
  put pMsg & return after tMsg
  -- add each attachment as a new part of the message, separting using
  -- the generated boundary
  if pAtts is an array then
    put "--" & tBoundary & return after tMsg
    repeat for each element tAtt in pAtts
    if there is a file tAtt["path"] then
      if tAtt["type"] is empty then
        put "application/octet-stream"
      else
        put tAtt["type"]
      end if
      put "Content-Type:" && tAtt["name"] & "; name=" & wrapQ(tAtt["name"]) & "; " & \
      return & "Content-Transfer-Encoding: base64;" & return & return & \
      base64Encode(URL ("binfile:" & tAtt["path"])) & return & "--" & \
      tBoundary & return after tMsg
    end if
  end repeat
  end if
  -- send the mail by piping the message we have just built to the sendmail command
  -- we must also send a copy of the message to the bcc addresses
  put shell("echo" && wrapQShellEscape(tMsg) && "| /usr/sbin/sendmail" && \
    wrapQShellEscape(pTo) && "-f" && wrapQShellEscape(pFrom)) && \
  if pBcc is not empty then
    put shell("echo" && wrapQShellEscape(tMsg) && "| /usr/sbin/sendmail" && \
    wrapQShellEscape(pBcc) && "-f" && wrapQShellEscape(pFrom))
  end if
  end mail
```

Prev: How to display errors when using LiveCode Server
Next: How do I pass information to LiveCode Server scripts?

6 Comments

Shedo Surashu Wednesday Feb 19 2019 at 05:03 PM

nice ^^

Tim Selander Sunday Jul 25 2010 at 07:24 AM

Hi, I'm a newbie at me and knows zero about Unix, shell and sendmail commands. Having a bit of a hard time following this. I don't need attachments, but do need to add these headers:

Content-Type: text/plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit

I have a variation of this script working (with help of Sarah Reichelt) but can you show me what the script would look like to add these two headers? Thanks.

Mark Wieder Sunday Jul 25 2010 at 09:17 AM

Change the section of code that sets the headers to read:

```
if pHtml is true then
  put "Content-Type: text/html; charset=iso-2022-jp" & return after tMsg
else
  put "Content-Type: text/plain; charset=iso-2022-jp" & return after tMsg
end if
put "Content-Transfer-Encoding: 7bit" & return after tMsg
```

Torsten Sunday Nov 25 2018 at 12:10 PM

Correction needed: in the sentence "We do this by define a the email as being multipart. " the "a" is not intended :-)

Elanor Buchanan Monday Nov 26 2018 at 01:49 AM

Hi Torsten

Thanks for bringing that to our attention, I have corrected the text in the lesson.

Elanor

Add your comment

Name* Email*

Comment*

Subscribe E-Mail me when someone replies to this comment

Are you human?

I'm not a robot
reCAPTCHA
Privacy - Terms

Submit Comment