

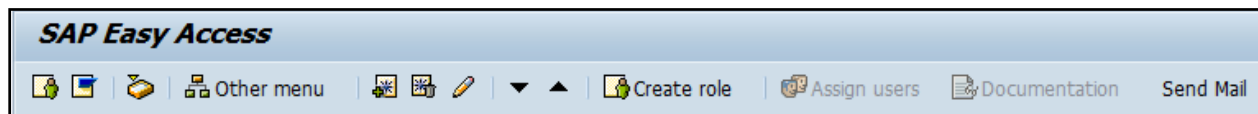
Liquid UI: WSCurl Send Mail

Purpose

Send email using WSCurl from SAP.

User Interface

Log into SAP and on the SAP Easy Access Screen click on the 'Send Mail' toolbar pushbutton.



Liquid UI Code [Script]

User Interface file -

```
SAPLSMTR_NAVIGATION.E0100.sjs
1  load('wscurl');
2
3  // Function to check if the string value is blank
4  function isBlank(jvar) {
5      if (jvar==void 0 || jvar==" " || jvar==null) {
6          return true;
7      } else {
8          return false;
9      }
10 }
11
12 function sendmail(){
13     /* Initialize the Curl object for making the call*/
14     var wsCurl = new Curl();
15
16     /* This is the URL for your mailserver. Note the use of port 587 here,
17     * instead of the normal SMTP port (25). Port 587 is commonly used for
18     * secure mail submission (see RFC4403), but you should use whatever
19     * matches your server configuration. */
20     // wsCurl_setopt(Curl.CURLOPT_URL, "smtp://smtp.google.com:587");
21     wsCurl_setopt(Curl.CURLOPT_URL, "smtp://mail.guixt.com:25");
22
23     /* In this example, we'll start with a plain text connection, and upgrade
24     * to Transport Layer Security (TLS) using the STARTTLS command. Be careful
25     * of using Curl.USESSL_TRY here, because if TLS upgrade fails, the transfer
26     * will continue anyway - see the security discussion in the libcurl
27     * tutorial for more details. Available 2nd Parameters are Curl.USESSL_NONE(0),
28     * Curl.USESSL_TRY(1), Curl.USESSL_CONTROL(2) or Curl.USESSL_ALL(3) */
29     // wsCurl_setopt(Curl.CURLOPT_USE_SSL, Curl.USESSL_ALL);
30
31     /* If your server doesn't have a valid certificate, then you can disable
32     * part of the Transport Layer Security protection by setting the
33     * CURLOPT_SSL_VERIFYPEER and CURLOPT_SSL_VERIFYHOST options to 0 (false).
34     * wsCurl_setopt(Curl.CURLOPT_SSL_VERIFYPEER, 0);
35     * wsCurl_setopt(Curl.CURLOPT_SSL_VERIFYHOST, 0);
36     * That is, in general, a bad idea. It is still better than sending your
37     * authentication details in plain text though.
38     * Instead, you should get the issuer certificate (or the host certificate
39     * if the certificate is self-signed) and add it to the set of certificates
40     * that are known to libcurl using CURLOPT_CAINFO.
41     * wsCurl_setopt(Curl.CURLOPT_CAINFO, "/path/to/certificate.pem");
42     */
43
44     /* A common reason for requiring transport security is to protect
45     * authentication details (user names and passwords) from being "snooped"
46     * on the network. Here is how the user name and password are provided: */
```

```

47 // wsCurl_setopt(Curl.CURLOPT_USERNAME, "user@example.net");
48 // wsCurl_setopt(Curl.CURLOPT_PASSWORD, "P@ssw0rd");
49 wsCurl_setopt(Curl.CURLOPT_USERNAME, "benjamin.dasari@guixt.com");
50 wsCurl_setopt(Curl.CURLOPT_PASSWORD, " "); // Enter the Password
51
52 /* value for envelope reverse-path. '<' '>' are REQUIRED around the email addr*/
53 // wsCurl_setopt(Curl.CURLOPT_MAIL_FROM, "<email_of_sender@example.net>");
54 wsCurl_setopt(Curl.CURLOPT_MAIL_FROM, "<benjamin.dasari@guixt.com>");
55 /* Add two recipients, in this particular case they correspond to the
56  * To: and Cc: addressees in the header, but they could be any kind of
57  * recipient. */
58 // wsCurl_slist_append("<email_of_receipent@example.net>");
59 // wsCurl_slist_append("<email_of_cc_receipent@example.net>");
60
61 /* Set the curl object to attach the recipients added above
62  * The second parameter is a REQUIRED string to maintain the syntax,
63  * however it has no effect on the execution*/
64 // wsCurl_setopt(Curl.CURLOPT_MAIL_RCPT, "recipients");
65
66 /* Since the traffic will be encrypted, it is very useful to turn on debug
67  * information within libcurl to see what is happening during the transfer.
68  */
69 wsCurl_setopt(Curl.CURLOPT_VERBOSE, 1);
70
71 /* Set the buffer header and the data for the e-mail that is being sent*/
72 var email = ["Date: Mon, 29 Nov 2010 21:54:29 +1100\n",
73             "To: <umang.desai@guixt.com>\n",
74             "From: <benjamin.dasari@guixt.com>\n",
75             "Message-ID: ddmmyyyy.hhhmm@domain.com\n",
76             "Subject: WS embedded SMTP TLS MESSAGE\n",
77             "\n", /* empty line to divide headers from body, see RFC5322 */
78             "The body of the message starts here.\n",
79             "\n",
80             "It could be a lot of lines, could be MIME encoded, whatever.\n",
81             "Check RFC5322.\n", NULL ];
82
83 /* In this case, we're using a callback function to specify the data. You
84  * could just use the CURLOPT_READDATA option to specify a var to read from*/
85 wsCurl_setopt(Curl.CURLOPT_READDATA, email);
86
87 /* Final step is to call execute to dispatch email*/
88 wsCurl_exec();
89
90 /* Clean up the recipient list*/
91 wsCurl_slist_free_all();

```

```

93 /* Close the smtp connection for the mail server*/
94 wsCurl.close();
95
96 /* Remove any reference for Garbage Collection*/
97 wsCurl= NULL;
98 }
99
100 // User Interface
101 pushbutton([TOOLBAR], 'Send Mail', '?', {"process":sendmail});

```