# WS Web Services

# 14.05 WSCurl Send Mail

### Prerequisites

- Products: Liquid UI WS, Liquid UI Server or Local DLL, Client Software
- Commands: WSCurl, pushbutton()
- File: wscurl.dll

### Purpose

Send email using WSCurl from SAP.

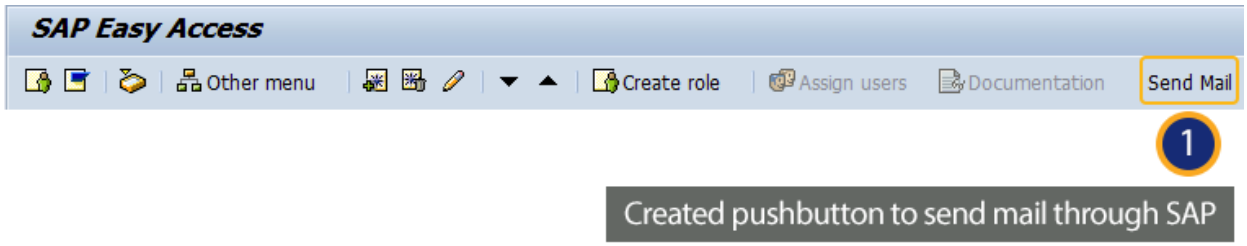
### User Interface

Create the file **SAPLSMTR_NAVIGATION.E0100.sjs** inside your script folder for customizing the **SAP Easy Access** screen.
//Now, let's start adding the script to the above file and **save** it.

### Customization

1. Logon to SAP, and create a toolbar pushbutton with the label Send Mail on the SAP Easy Access screen, as shown in the image below.

   On the SAP screen, a pushbutton with the label "Send Mail" is created.
   pushbutton([TOOLBAR], 'Send Mail', '?', {"process":sendmail});




2. Add the following Liquid UI script to the above file and save it.

```
load('wscurl');
// Function to check if the string value is blank
function isBlank(jvar) {
    if (jvar==void 0 || jvar=="" || jvar==null) {
```

```
        return true;
    } else {
        return false;
    }
}

function sendmail(){
   /* Initialize the Curl object for making the call*/
   var wsCurl = new Curl();


/* This is the URL for your mailserver. Note the use of port 587
 here,
     * instead of the normal SMTP port (25). Port 587 is commonl
y used for
     * secure mail submission (see RFC4403), but you should use
whatever
     * matches your server configuration. */
    // wsCurl.setopt(Curl.CURLOPT_URL, "smtp://smtp.google.com:5
87");
    wsCurl.setopt(Curl.CURLOPT_URL, "smtp://mail.guixt.com:25");


  /* In this example, we'll start with a plain text connection,
and upgrade
     * to Transport Layer Security (TLS) using the STARTTLS comm
and. Be careful
     * of using Curl.USESSL_TRY here, because if TLS upgrade fai
ls, the transfer
     * will continue anyway - see the security discussion in the
 libcurl
     * tutorial for more details. Available 2nd Parameters are C
url.USESSL_NONE(0),
    * Curl.USESSL_TRY(1), Curl.USESSL_CONTROL(2) or Curl.USESSL_
ALL(3)*/
    wsCurl.setopt(Curl.CURLOPT_USE_SSL, Curl.USESSL_ALL);


/* If your server doesn't have a valid certificate, then you can
 disable
     * part of the Transport Layer Security protection by settin
g the
     * CURLOPT_SSL_VERIFYPEER and CURLOPT_SSL_VERIFYHOST options
 to 0 (false).
     *   wsCurl.setopt(Curl.CURLOPT_SSL_VERIFYPEER, 0);
     *   wsCurl.setopt(Curl.CURLOPT_SSL_VERIFYHOST, 0);
     * That is, in general, a bad idea. It is still better than
sending your
     * authentication details in plain text though.
```

```
     * Instead, you should get the issuer certificate (or the ho
st certificate
     * if the certificate is self-
signed) and add it to the set of certificates
     * that are known to libcurl using CURLOPT_CAINFO.
    *    wsCurl.setopt(Curl.CURLOPT_CAINFO, "/path/to/certificat
e.pem");
     */


 /* A common reason for requiring transport security is to prote
ct
     * authentication details (user names and passwords) from be
ing "snooped"
     * on the network. Here is how the user name and password ar
e provided: */
    // wsCurl.setopt(Curl.CURLOPT_USERNAME, "user@example.net");
    // wsCurl.setopt(Curl.CURLOPT_PASSWORD, "P@ssw0rd");
    wsCurl.setopt(Curl.CURLOPT_USERNAME, "benjamin.dasari@guixt.
com");
    wsCurl.s
etopt(Curl.CURLOPT_PASSWORD, " ");      // Enter the Password

    /* value for envelope reverse-
path. '<' '>' are REQUIRED around the email addr*/
    // wsCurl.setopt(Curl.CURLOPT_MAIL_FROM, "<email_of_sender@e
xample.net>");
    wsCurl.setopt(Curl.CURLOPT_MAIL_FROM, "<benjamin.dasari@guix
t.com>");

  /* Add two recipients, in this particular case they correspond
 to the
     * To: and Cc: addressees in the header, but they could be a
ny kind of
     * recipient. */
    // wsCurl.slist_append("<email_of_recipent@example.net>");
    // wsCurl.slist_append("<email_of_cc_recipent@example.net>")
;

   /* Set the curl object to attach the recipients added above
    * The second parameter is a REQUIRED string to maintain the
syntax,
    *  however it has no effect on the execution*/
   // wsCurl.setopt(Curl.CURLOPT_MAIL_RCPT, "recipients");

    /* Since the traffic will be encrypted, it is very useful to
 turn on debug
     * information within libcurl to see what is happening durin
g the transfer.
```

```
     */
    wsCurl.setopt(Curl.CURLOPT_VERBOSE, 1);

   /* Set the buffer header and the data for the e-
mail that is being sent*/
   var email = ["Date: Mon, 29 Nov 2010 21:54:29 +1100\n",
               "To: <umang.desai@guixt.com>\n",
               "From: <benjamin.dasari@guixt.com>\n",
               "Message-ID: ddmmyyyy.hhmm@domain.com\n",
               "Subject: WS embedded SMTP TLS MESSAGE\n",
               "\n", /* empty line to divide headers from body,
see RFC5322 */
               "The body of the message starts here.\n",
               "\n",
               "It could be a lot of lines, could be MIME encode
d, whatever.\n",
               "Check RFC5322.\n", NULL ];


/* In this case, we're using a callback function to specify the
data. You
     * could just use the CURLOPT_READDATA option to specify a v
ar to read from*/
    wsCurl.setopt(Curl.CURLOPT_READDATA, email);

   /* Final step is to call execute to dispatch email*/
   wsCurl.exec();

   /* Clean up the recipient list*/
   wsCurl.slist_free_all();

   /* Close the smtp connection for the mail server*/
   wsCurl.close();

   /* Remove any reference for Garbage Collection*/
   wsCurl= NULL;
}
```

3. Clicking on the Send Mail toolbar pushbutton will send an email from SAP to the desired email address.

# WS Web Services

Unique solution ID: #1820
Author: Poojitha Reddy
Last update: 2023-02-17 06:00