Subject: Re: SOCKET and POST forms
Posted by Ken Mankoff on Wed, 19 Jun 2002 18:20:03 GMT
View Forum Message <> Reply to Message

On Wed, 19 Jun 2002, Ken Mankoff wrote:
> Does anyone know how to use SOCKET to fill out a POST form?

OK, I figured it out.

If anyone needs to do this in the future, send me an email. Remove the
part about my feelings towards spam from the email above.

Ken Mankoff

Subject: Re: SOCKET and POST forms
Posted by R.Bauer on Thu, 20 Jun 2002 07:17:05 GMT
View Forum Message <> Reply to Message

Ken Mankoff wrote:
>
> On Wed, 19 Jun 2002, Ken Mankoff wrote:
>> Does anyone know how to use SOCKET to fill out a POST form?
>
> OK, I figured it out.
>
> If anyone needs to do this in the future, send me an email. Remove the
> part about my feelings towards spam from the email above.
>
> Ken Mankoff

Dear Ken,

please can you post the solution!
I believe this should be saved by David as a special Tip.

regards

Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de

---------- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml
========== =======

Subject: Re: SOCKET and POST forms
Posted by Ken Mankoff on Thu, 20 Jun 2002 15:52:30 GMT
View Forum Message <> Reply to Message

On Thu, 20 Jun 2002, Reimar Bauer wrote:
> Ken Mankoff wrote:
>>
>> On Wed, 19 Jun 2002, Ken Mankoff wrote:
>>>
>>> Does anyone know how to use SOCKET to fill out a POST form?
>>>
>>
>> OK, I figured it out.
>> If anyone needs to do this in the future, send me an email. Remove the
>> part about my feelings towards spam from the email above.
>>
>
> please can you post the solution!
> I believe this should be saved by David as a special Tip.
>

OK, here is the code.

(Andrew: I have updated it a bit from what I emailed you last night
mostly just documentation, and a better implementation of the "5"
section of the code, so use this version unless you just wanted the
POST code)

Below the code and all the comments, I have duplicated the POST
section of the code, so if that is all you are interested in, scroll
to the very bottom...

```
;+
; NAME:
;   TLE_OIG
;
; PURPOSE:
;   This function retrieves the latest (or last 5) Two Line Element
;   (TLE) from the NASA Orbital Information Group (OIG) website.
;
; CATEGORY:
;   SNOE, Satellite, Orbit, Operations, Flight
```

```
;
; CALLING SEQUENCE:
;   Result = TLE_OIG()
;
; OPTIONAL INPUTS:
;   *SAT_NUMBER: The Satellite Object Number. [DEFAULT: 25233 (snoe)]
;              This can either be a single number, or an array of
;              numbers.
;   USERNAME:   The username of the OIG account
;   PASSWORD:   The password of the OIG account
;
; KEYWORD PARAMETERS:
;   *FIVE:      Set this keyword to return the last 5 TLEs. If set,
;              the output is an array of 5 structures. The 0th index
;              in the array is the oldest TLE. The 4th element is the
;              newest (last) TLE.
;   VERBOSE:    Set this keyword to display debugging information
;   HELP:       Set this keyword to display the IDL help for this routine
;
; OUTPUTS:
;   This function returns a structure of the TLE elements, broken down
;   according to the OIG website "TLE Format". In addition, two fields
;   of the structure named 'line1' and 'line2' contain the two lines
;   of the TLE as strings.
;
;   *NOTE on SAT_NUMBER and /FIVE:
;   a) If SAT_NUMBER is a scalar and FIVE is NOT set then:
;      The output is a structure.
;   b) If SAT_NUMBER is a scalar and FIVE is set then:
;      The output is an array of 5 structures.
;   c) If SAT_NUMBER is an array, and FIVE is NOT set then:
;      The output is an array, 1 structure per element in SAT_NUMBER.
;   d) If SAT_NUMBER is an array, and FIVE is set, then:
;      The output is an array, [5, N_ELEMENTS( SAT_NUMBER )]
;
;   The output structure is:
;   struct = { name='TLE':
;              name:            ; The name of the satellite
;              catalog_no:      ; The catolog number (input to this program)
;              security_class:
;              internationalID:
;              YYDDD_FOD:       ; Year, Day-of-Year, Fraction-of-Day
;              time1_deriv:
;              time2_deriv:
;              time2_deriv_exp:
;              bstar_drag:
;              bstar_drag_exp:
;              ephemeris_type:
```

```
;             element_number:
;             chksum1_MOD_10:
;             inclination:
;             raan:             ; Right ascension of ascending node
;             eccentricity:
;             arg_OF_perigee:
;             mean_anomaly:
;             revs_per_day:
;             rev_num_epoch:
;             chksum2_MOD_10:
;             line1:            ; The first line of the TLE
;             line2:            ; The second line of the TLE
;
; RESTRICTIONS:
;   Needs network access
;
; PROCEDURE:
;   * See http://oig1.gsfc.nasa.gov
;   * See code
;   * General procedure is to access TLEs from the OIG home page. The
;     OIG homepage is unlpeasant to navigate, in that each page is
;     created with an embedded hash key that is used to generate the
;     next page (with its unique embedded hash key, etc...) You can
;     see this by going to the home page, and clicking on "View
;     Source" in your browser. Follow a few links and repeat.
;
;     Anyway, the code reads in a page looking for the "Link Name"
;     (i.e. what you would see and click on in your browser). It then
;     parses the line for the embedded hash key, and requestes the
;     next page using that key. This procedure is repeated for each
;     page.
;
;     Finally, the requested Satellite Object Number is passed to the
;     OIG site, and the TLEs are returned. This page is read in,
;     parsed into a structure, and then the structure is returned to
;     the user and the program ends.
;
;     The parsing is based upon a document published on the OIG
;     website titled "TLE Format (Standard and Obsolete)" This
;     document is reproduced in its entirety at the bottom of the
;     procedure.
;
; EXAMPLE:
;   tle = TLE_OIG()             ; get the latest SNOE TLE
;   tle = TLE_OIG( /FIVE )      ; get the last 5 SNOE TLEs
;
;   ;;; To get the last TLE for the ISS, using an OIG account with
;   ;;; username of 'foo' and a password of 'bar', type this:
```

```
;   tle = TLE_OIG( sat=25544, user='foo', pass='bar )
;   help, tle, /st
;   r = 42241.1 / (tle.revs_per_day^(2/3.))
;   alt = r - 6378.1
;
;   ;;; To compare SNOE and ISS for the last five TLEs:
;   IDL> tle = TLE_OIG( sat=[25544,25233], /FIVE )
;   IDL> help, a
;   A               STRUCT    = -> ISS (ZARYA) Array[5, 2]
;   FOR i=0,4 DO print, a[i,0].revs_per_day, a[i,1].revs_per_day
;   ;;; NOTE that TLES are not at the same timescale
;   FOR i=0,4 DO print, a[i,0].YRDOY_FODddddd, a[i,1].YRDOY_FODddddd
;
; MODIFICATION HISTORY:
;  Written by: Ken Mankoff, 2002-06-19, LASP
;-

FUNCTION tle_oig, $
                sat_number = sat_number, $
                username = username, $
                password = password, $
                five=five, $
                verbose=verbose, $
                help=help

IF ( keyword_set( help ) ) THEN BEGIN
   DOC_LIBRARY, 'tle_oig'
   return, -1
ENDIF

;;; check to make sure inputs are valid.
IF ( n_elements( username ) EQ 0 ) THEN user = 'mankoff' ELSE user = username
IF ( n_elements( password ) EQ 0 ) THEN pass = 'passwd' ELSE pass = password
IF ( n_elements( sat_number ) EQ 0 ) $
  THEN satn = '25233' ELSE satn = STRTRIM( sat_number, 2 )

;;; IF multiple satellite IDs requested, then just call yourself 5
;;; times and build up an array.
IF ( n_elements( satn ) GT 1 ) THEN BEGIN
   r = TLE_OIG( s=satn[0], u=user, p=pass, f=five, v=verbose )
   FOR i = 1, n_elements( satn )-1 DO $
     r = [ [r], [ TLE_OIG( s=satn[i], u=user, p=pass, f=five, v=verbose ) ] ]
   return, REFORM( r )
ENDIF


line = ' '                    ; empty line to read in HTML
oig = 'oig1.gsfc.nasa.gov'      ; BASE HREF
```

```
;;; Go to the FRONT OIG home page, get the hash key for the
;;; "Main Home Page"
IF ( keyword_set( verbose ) ) THEN print, "Contacting NASA OIG Webpage..."
SOCKET, lun, oig, 80, /GET
PRINTF, lun, 'GET http://' + oig + '/scripts/foxweb.exe/app01'
WHILE ( NOT STREGEX( line, 'OIG Main Page', /BOOLEAN ) ) DO $
  readf, lun, line
hash = STREGEX( line, "'.*'", /EXTRACT )

;;; Fetch the Main Page, and get the hash key for the "Registered User
;;; Login" page
IF ( keyword_set( verbose ) ) THEN print, "Getting 'Main Page'..."
URL = 'GET http://' + oig + STRMID( hash, 1, STRLEN( hash )-2 )
FREE_LUN, lun
SOCKET, lun, oig, 80, /GET
PRINTF, lun, URL
WHILE ( NOT STREGEX( line, 'Registered User Login', /BOOLEAN ) ) DO $
  readf, lun, line
hash = STREGEX( line, "'.*'", /EXTRACT )

;;; Fetch the "Registered Login Page", and get the hash key to submit
;;; the form
IF ( keyword_set( verbose ) ) THEN $
  print, "Getting 'Registered User Login' Page..."
URL = 'GET http://' + oig + STRMID( hash, 1, STRLEN( hash )-2 )
FREE_LUN, lun
SOCKET, lun, oig, 80, /GET
PRINTF, lun, URL
WHILE ( NOT STREGEX( line, "'tdac'", /BOOLEAN ) ) DO readf, lun, line
hash = STREGEX( line, 'VALUE=".*"', /EXTRACT )
hash = STRMID( hash, 7 ) & hash = STRMID( hash, 0, STRLEN( hash )-1 )

;;; POST the form data (the hash key, the username, and the password)
IF ( keyword_set( verbose ) ) THEN print, "Logging In..."
FREE_LUN, lun
SOCKET, lun, oig, 80, /GET
PRINTF, lun, 'POST http://' + oig + '/scripts/foxweb.exe/loginok@app01?'
PRINTF, lun, 'tdac=' + hash
PRINTF, lun, 'ffv01=' + user
PRINTF, lun, 'ffv02=' + pass
FREE_LUN, lun
;;; NOTE: Rather than using a FREE_LUN and re-opening the stream with
;;; SOCKET, the code should just send an EOF or EOT after the POST
;;; data has been sent. Not sure how to implement this right
;;; now... EOT is 04 (in Hex, and Dec), so maybe just a
;;; "PRINTF, lun, O4"?  Nope... that does not work.
```

```
;;; GET the "Registered User Login OK" page, and the hash
;;; key to 'Continue'
SOCKET, lun, oig, 80, /GET
PRINTF, lun, 'GET http://' + oig + $
       '/scripts/foxweb.exe/loginok@app01?' + $
       'tdac=' + hash + $
       '&ffv01=' + user + $
       '&ffv02=' + pass
WHILE( NOT STREGEX( line, 'Continue', /BOOLEAN ) ) DO readf, lun, line
hash = STREGEX( line, '".*"', /EXTRACT )

;;; Fetch the "User Home Page", and get the hash key for
;;; "TLE Query" link
IF ( keyword_set( verbose ) ) THEN print, "Getting 'User Home Page'..."
URL = 'GET http://' + oig + STRMID( hash, 1, STRLEN( hash )-2 )
FREE_LUN, lun
SOCKET, lun, oig, 80, /GET
PRINTF, lun, URL
WHILE( NOT STREGEX( line, 'TLE Query', /BOOLEAN ) ) DO READF, lun, line
hash = STREGEX( line, '".*"', /EXTRACT )

;;; Fetch the "TLE Query", and get the hash key to submit the form
IF ( keyword_set( verbose ) ) THEN print, "Getting 'TLE Query' Form..."
URL = 'GET http://' + oig + STRMID( hash, 1, STRLEN( hash )-2 )
FREE_LUN, lun
SOCKET, lun, oig, 80, /GET
PRINTF, lun, URL
WHILE ( NOT STREGEX( line, '"tdac"', /BOOLEAN ) ) DO readf, lun, line
hash = STREGEX( line, 'VALUE=".*"', /EXTRACT )
hash = STRMID( hash, 7 ) & hash = STRMID( hash, 0, STRLEN( hash )-1 )

;;; POST the form data (Satellite Object Number)
IF ( keyword_set( verbose ) ) THEN Print, "Submitting TLE Query"
IF ( keyword_set( five ) ) THEN ffv04 = 'five' ELSE ffv04 = 'last'
FREE_LUN, lun
SOCKET, lun, oig, 80, /GET
PRINTF, lun, 'POST http://' + oig + '/scripts/foxweb.exe/ftleadhocr@app01?'
PRINTF, lun, 'tdac=' + hash
PRINTF, lun, 'ffv01=' + satn
PRINTF, lun, 'ffv02=standard'
PRINTF, lun, 'ffv03=catno'
PRINTF, lun, 'ffv04='+ffv04
FREE_LUN, lun

;;; Read the returned TLE info
IF ( keyword_set( verbose ) ) THEN PRINT, "Reading TLE"
SOCKET, lun, oig, 80, /GET
PRINTF, lun, 'GET http://' + oig + $
```

```
        '/scripts/foxweb.exe/ftleadhocr@app01?' + $
        'tdac=' + hash + $
        '&ffv01=' + satn + $
        '&ffv02=standard' + $
        '&ffv03=catno' + $
        '&ffv04=' + ffv04
;;; read the HTML header and the blank line that follows it that are
;;; above the TLE. Then, read and parse the TLE itself.
WHILE ( NOT STREGEX( line, 'PRE', /BOOLEAN ) ) DO readf, lun, line
readf, lun, line ; read blank line

five: ;;; loop to here if reading in 5 TLEs
name = ' ' & readf, lun, name   ; read satellite name
line1 = ' ' & readf, lun, line1 ; read the 1st TLE line
line2 = ' ' & readf, lun, line2 ; read the 2nd TLE line

;;; now, put the TLE info into a structure
r = CREATE_STRUCT( name = 'TLE', $
                   'name', name, $
                   $ ; LINE 1
                   'catalog_no',          LONG( STRMID( line1, 2, 5 ) ), $
                   'security_class',            STRMID( line1, 7, 1 ), $
                   'internationalID',           STRMID( line1, 9, 8 ), $
                   'YYDDD_FOD',         DOUBLE( STRMID( line1, 18, 14 ) ), $
                   'time1_deriv',       DOUBLE( STRMID( line1, 33, 10 ) ), $
                   'time2_deriv',       DOUBLE( STRMID( line1, 44, 6 ) ), $
                   'time2_deriv_exp',   DOUBLE( STRMID( line1, 50, 2 ) ), $
                   'bstar_drag',        DOUBLE( STRMID( line1, 53, 6 ) ), $
                   'bstar_drag_exp',    DOUBLE( STRMID( line1, 59, 2 ) ), $
                   'ephemeris_type',       FIX( STRMID( line1, 62, 1 ) ), $
                   'element_number',      LONG( STRMID( line1, 64, 4 ) ), $
                   'chksum1_MOD_10',       FIX( STRMID( line1, 68, 1 ) ), $
                   $ ; LINE 2
                   'inclination',       DOUBLE( STRMID( line2, 8, 8 ) ), $
                   'raan',              DOUBLE( STRMID( line2, 17, 8 ) ), $
                   'eccentricity',DOUBLE( '.' + STRMID( line2, 26 ,7 ) ), $
                   'arg_OF_perigee',    DOUBLE( STRMID( line2, 34, 8 ) ), $
                   'mean_anomaly',      DOUBLE( STRMID( line2, 43, 8 ) ), $
                   'revs_per_day',      DOUBLE( STRMID( line2, 52, 11 ) ), $
                   'rev_num_epoch',     DOUBLE( STRMID( line2, 63, 5 ) ), $
                   'chksum2_MOD_10',    DOUBLE( STRMID( line2, 68, 1 ) ), $
                   $ ; LINES 1 and 2 unparsed
                   'line1', line1, $
                   'line2', line2 )
IF ( ffv04 EQ 'last' ) THEN BEGIN  ;;; ONLY 1 TLE requested
   FREE_LUN, lun
   return, r
ENDIF ELSE BEGIN ;;; Last 5 TLEs requested (read 4 more!)
```

```
    ;;; NOTE: This is a cheap hack to read in 5 TLEs (4 more) and to
    ;;; use the big structure definition and formatting code above
    ;;; only once.
    IF ( n_elements( counter ) EQ 0 ) THEN BEGIN ; first time through
       counter = 0
       r5 = [ r,r,r,r,r ]
    ENDIF
    r5[ 4-counter ] = r
    counter = counter + 1
    IF ( counter LT 5 ) THEN GOTO, five
ENDELSE
return, r5

END


; Two Line Element as received

; Old TLE format(obsolete)

; SAT ID = 23635
; Latest set =
; 1 23635U 95040B   95215.97504380 -.00000165 +00000-0 +10000-3 0 00018
; 2 23635 004.1663 111.0618 7252147 179.2403 004.8644 02.18377056000000

; New TLE format(standard)

; 23635B
; 1 23635U 95040B   95215.97504380 -.00000165 +00000-0 +10000-3 0 00018
; 2 23635 004.1663 111.0618 7252147 179.2403 004.8644 02.18377056000000


; Break-out of a Two Line Element

; 23635B  =  Name of this two line element set of three lines

; !>Line Number
; !   !>Catalog Number
; !   !  !>Security Classification for this Element Set
; !   !  !    !>International Identification for this Object
; !   !  !    !     !>Two Digit Year
; !   !  !    !     !  !>Day of Year
; !   !  !    !     !  !     !>Fraction of 24 Hour Day
; !   !  !    !     !  !     !     !>Sign of 1st Time Derivative
; !   !  !    !     !  !     !     !    !>1st Time Derivative
; !   !  !    !     !  !     !     !    !     !>Sign of 2nd Time Derivative
; !   !  !    !     !  !     !     !    !     !  !>2nd Time Derivative
; !   !  !    !     !  !     !     !    !     !  !  !>Sign of exponent
```

```
; !  ! !   !    ! !    !    !   !    ! !  !>Exponent 2ndTimeDerivative
; !  ! !   !    ! !    !    !   !    ! !    !>Sign of BSTAR drag term
; !  ! !   !    ! !    !    !   !    ! !    !  !>BSTAR/Drag Term
; !  ! !   !    ! !    !    !   !    ! !    !  !  !>Eign of Exponent
; !  ! !   !    ! !    !    !   !    ! !    !  !  !!>Exponent BstarDrg
; !  ! !   !    ! !    !    !   !    ! !    !  !  !! !>Ephemeris Type
; !  ! !   !    ! ---   !    !   !    ! !    !  !  !! ! !>Element No.
; ! -----! -------- --|||--------- !--------- !-----   !-----!! ! ----!>Checksum
; 1 23635U 95040B   95215.97504380 -.00000165 +00000-0 +10000-3 0 00018 = Line 1

; 2 23635 004.1663 111.0618 7252147 179.2403 004.8644 02.18377056000000 = Line 2

;  12345678901234567890123456789012345678901234567890123456789012345678901234567890 123456789
; ! ----- ---.---- ---.---- ------- ---.---- ---.---- --.--------|||||!>Checksum
; !   !   Incl     RAAN     Ecc     AoP      MA       RpD        -----
; !   !   !        !        !       !        !        !          !>Rev # @ Epoch
; !   !   !        !        !       !        !        !>Revolutions Per Day
; !   !   !        !        !       !        !>Mean Anomaly
; !   !   !        !        !       !>Argument of Perigee
; !   !   !        !        !>Eccentricity, with assumed decimal point leading
; !   !   !        !>Right Asencion of Ascending Node
; !   !   !>Inclination
; !   !>Catalog Number
; !>Line Number


; Line 1

;    Column Numbers     Number of
;    First     Last     Characters     Description
;      1         1          1          Line No. Identification
;      3         7          5          Catalog No.
;      8         8          1          Security Classification
;     10        17          8           International Identification
;     19        32         14           YRDOY.FODddddd
;     34        34          1          Sign of first time derivative
;     35        43          9          1st Time Derative
;     45        45          1          Sign of 2nd Time Derivative
;     46        50          5          2nd Time Derivative
;     51        51          1          Sign of 2nd Time Derivative Exponent
;     52        52          1          Exponent of 2nd Time Derivative
;     54        54          1          Sign of Bstar/Drag Term
;     55        59          5          Bstar/Drag Term
;     60        60          1          Sign of Exponent of Bstar/Drag Term
;     61        61          1          Exponent of Bstar/Drag Term
;     63        63          1          Ephemeris Type
;     65        68          4          Element Number
;     69        69          1          Check Sum, Modulo 10
```

```
; Line 2

;    Column Numbers     Number of
;    First     Last     Characters     Description
;      1         1          1          Line No. Identification
;      3         7          5          Catalog No.
;      9        16          8           Inclination
;     18        25          8            Right Asencion of Ascending Node
;     27        33          7            Eccentricity with assumed leading decimal
;     35        42          8            Argument of the Perigee
;     44        51          8            Mean Anomaly
;     53        63         11             Revolutions per Day
;     64        68          5            Revolution Number at Epoch
;     69        69          1            Check Sum Modulo 10

; More Definitive definitions;

; Sign digit where used; Only negative values are flagged with a "minus",
;      positive values have a "space".

; First time  derivative of the mean motion or ballistic coefficient (depending
; on Ephemeris Type).  Revolution per day-squared or meters-squared per
; kilogram.

; "All orbital elements (TLE) are referred to the mean equator and equinox of
; date."

; END OF TLE_OIG.PRO
 ;------------------------------------------------------------ -------------
```

OK, here is the meat of the POST segment

```
; open the socket
SOCKET, lun, oig, 80, /GET
; POST the CGI script you are manipulating
PRINTF, lun, 'POST http://' + oig + '/scripts/foxweb.exe/loginok@app01?'
; POST the key=value pairs of the data you want to submit
PRINTF, lun, 'tdac=' + hash
PRINTF, lun, 'ffv01=' + user
PRINTF, lun, 'ffv02=' + pass
; close the stream (send an EOF somehow)
```

```
FREE_LUN, lun
; re-open the socket. NOTE: if this were a GET form, none of the above
; would happen, and you would just start the code here (test it on
; google and you'll see GETS are very simple)
SOCKET, lun, oig, 80, /GET
; write a GET command (even though this is a POST, it seems to work!)
; based upon the above CGI script and key=value pairs
PRINTF, lun, 'GET http://' + oig + $
      '/scripts/foxweb.exe/loginok@app01?' + $
      'tdac=' + hash + $
      '&ffv01=' + user + $
      '&ffv02=' + pass
; read back the stream from the server just like you always do with SOCKET
WHILE( NOT EOF( lun ) ) DO BEGIN readf, lun, line & print, line
```

Subject: Re: SOCKET and POST forms
Posted by mikef on Fri, 21 Jun 2002 00:21:01 GMT

View Forum Message <> Reply to Message

In article <Pine.LNX.4.44.0206200946230.27237-100000@snoe.colorado.edu>,
Ken Mankoff <mankoff@I.HATE.SPAM.cs.colorado.edu> wrote:
>
...
> FREE_LUN, lun
> ;;; NOTE: Rather than using a FREE_LUN and re-opening the stream with
> ;;; SOCKET, the code should just send an EOF or EOT after the POST
> ;;; data has been sent. Not sure how to implement this right
> ;;; now... EOT is 04 (in Hex, and Dec), so maybe just a
> ;;; "PRINTF, lun, O4"? Nope... that does not work.
>

Try "PRINTF, lun, String(O4b)"
and use IDL's conversion of bytes in strings.
--
--
Mike Fitzgibbon MRFitz@ns.arizona.edu
UofAz, LPL phone:(520)626-4791
Systems Programmer, Pr. fax: (520)621-6783

Subject: Re: SOCKET and POST forms
Posted by Ken Mankoff on Fri, 21 Jun 2002 15:34:11 GMT

View Forum Message <> Reply to Message

On Fri, 21 Jun 2002, Mike Fitzgibbon wrote:

> Ken Mankoff <mankoff@I.HATE.SPAM.cs.colorado.edu> wrote:
>>
> ...
>> FREE_LUN, lun
>> ;;; NOTE: Rather than using a FREE_LUN and re-opening the stream with
>> ;;; SOCKET, the code should just send an EOF or EOT after the POST
>> ;;; data has been sent. Not sure how to implement this right
>> ;;; now... EOT is 04 (in Hex, and Dec), so maybe just a
>> ;;; "PRINTF, lun, O4"? Nope... that does not work.
>>
>
> Try "PRINTF, lun, String(O4b)"
> and use IDL's conversion of bytes in strings.
>

Well the IDL code does print EOF (or EOT). Thank you. But it does not do what I thought it might do in the comment.

But that entire comment was just conjecture on my part. I have no idea exactly what the HTTP protocol is to deal with POST requests, but it seemed like closing the stream and re-opening it with a GET request is not the right way to do it... I got it working by trial-and-error coding, and assumed that an EOF would be a better solution

But doing the above PRINT, STRING(04B) command does not work, so maybe I got it right the first time.

The main problem is that there is no documentation on this. Docs are either written for the HTML form writer, in which case the POST vs GET method does not matter, so there is just a 1 sentence explanation about the 2 methods, or, docs are written for the CGI script writer, in which case there is are a few lines difference in order to parse the query (and maybe a few different ENV vars), but thats it. But what my code did was to do the actual POSTing (which is normally abstracted from FORM programmer and done by the browser), and there is no documentation specifying what bytes/requests need te be sent in what order to do a POST.

I think maybe the HTML language specifications might cover this, but I don't feel like digging that deep since it appears to work...

-k.