Using REXX/VSE to Process Web Requests

BY LEO J. LANGEVIN

In the November 1996 issue of ***Technical Support,*** I wrote a lengthy article on how to develop your own CGI (Common Gateway Interface) program to access VSE data. This process allows an Internet/intranet web request to pass control to a VSE program and the VSE program to generate a dynamic web page and pass this data back to the user. It's a very powerful tool, but some people have found it difficult to implement.

In June 1998, I was at the IBM Technical Conference in Reno, and several people asked me if there was an easier way to perform the same process. They did not want to code the CGI in assembler, so a simplified assembler design was not what they were looking for. Alas, it seems that there aren't as many of us assembler programmers as there used to be.

So out of a few people explaining what they wanted, the company I work for, Connectivity Systems, came up with a solution: a REXX/VSE interface to the HTTP application.

This month's column demonstrates how to construct in four simple steps a quick and easy VSE Web Console program using only a few lines of REXX/VSE code. By the way, this code will only work under VSE/ESA version 2 or higher. For those of you who are not running version 2 of VSE/ESA, you can still perform REXX processing, but you can't use the built-in console functions that are VSE2 dependent. One more thing: You will need to be at service pack "G" or higher of TCP/IP for VSE to be able to run REXX programs.

This month's column demonstrates how to construct in four simple steps a quick and easy VSE Web Console program using only a few lines of REXX/VSE code.

Step 1: Create the web page.

The first thing that you will need to do is write a web page that will invoke the CGI. Figure 1 presents a sample web page that you may want to catalog as "CONSOLE.HTML" in your HTTPD library. The HTTPD library is the one that you defined in your "ROOT=" parameter as part of the HTTPD definition within the TCP/IP for VSE product.

As you can see, the web page is pretty straightforward. It displays a page with a single input field. The user enters the VSE command and the request is passed to a VSE program called VSECOM. VSECOM happens to be a REXX program. For TCP/IP for VSE to know that this is a REXX program, you will need to define the CGI to TCP/IP for VSE configuration file.

Step 2: Define the CGI to TCP/IP for VSE using the following:

```
DEFINE FILE,PUBLIC='VSECOM',DRIVER=VSECOM,TYPE=CGI-REXX
```

And that's all there is to it.

Step 3: Catalog the REXX CGI.

Now as for the CGI program, you would catalog VSECOM.PROC in a sublibrary that is part of the search string within your TCP/IP for VSE partition.

Figure 2 shows the code that I used. In this example, I used the "HTML" function to process HTML lines. This is a packaged function created for TCP/IP for VSE and is part of service pack "G". It will return either a "0000" or "0008" to indicate success or failure. As far as the input parameters, the HTTP daemon will pass the name of the user, the person's password, and the data associated with the request to the REXX program. As you can see, this is retrieved by using the ARG function within REXX.

By using the "parse" function, the REXX CGI obtains the data which follows "&command=". This was the field name that I indicated in line 14 of the original web page. When used, field names are always passed with the data.

Figure 1: Console Web Page Example

```
CATALOG CONSOLE.HTML EOD=/+ REP=Y
<HTML>
<HEAD>
<TITLE>VSE Console Command Processor
</TITLE></HEAD>
<BODY TEXT="#993300" BGCOLOR="#66FF99">
<CENTER>
<H2>
<B><I><FONT COLOR="#000000">
VSE Console Command Processor
</FONT></I></B></H2></CENTER>
<P>
<HR><FORM METHOD=GET  ACTION="VSECOM">
Input:
<INPUT TYPE="text" NAME="command" SIZE=25>
<BR>
<HR>
<BR><FONT COLOR="#000066">
<PRE>Output: (none)</PRE>
</BODY>
</HTML>
/+
```

In addition to REXX being an easier language to learn than Assembler, there is another benefit to using REXX: You don't have to worry about the passed data. Using the old CGI process you needed to convert the data, replace "+" with blanks, and replace "%nn" with an ASCII hex value. The CGI-REXX interface will perform this process for you. You also don't have to indicate the mime type. If you want to override the default of "text\html;", you can do so by putting the mime type in the first line of the returned buffer. Otherwise, just pass HTML statements, and let the CGI processor do the rest.

Unlike the old CGI process, you don't have to be concerned with OPEN/CLOSE processing. Since the REXX program is not re-entrant, it's considered a one-time request. All you do is send back what you want the user to see, and that's what they'll see.

Let's take this current example as a functioning system. When the CGI shown in Figure 2 is invoked, the original input page is redisplayed, but the resulting output is also displayed in the bottom half of the screen. A new input field is shown so that the user can re-enter data. The user never needs to know that the screen is dynamically generated.

Step 4: Invoke the web page.

Let's say that your VSE IP address is 100.100.1.1. You would invoke this program from your web browser by entering the following URL:

```
http://100.100.1.1/CONSOLE.HTML
```

And there you have it — a simple VSE console program. If you use this, you may want to have VSECOM perform some userid and password checking! And, with REXX/VSE, there's no excuse for not allowing remote users to access VSE data. It's cool, it's amazing, and it's easy. ts

NaSPA member Leo J. Langevin has been involved in the VSE community since its inception. He is employed by Connectivity Systems and is the lead developer of NFS for VSE. He can be reached at leo@tcpip4vse.com.

Figure 2: REXX CGI Example

```
CATALOG VSECOM.PROC REP=Y EOD=/+
/*                                  */
/* Get the passed parameters        */
/*                                  */
/* ---------------------------------  */
userid=arg(1)
password=arg(2)
data=arg(3)
parse upper var data request 9 command
/*                                  */
/* Activate the console interface   */
/*                                  */
ADDRESS CONSOLE
'ACTIVATE NAME CGICONR PROFILE REXNORC'
'CART USCHI'
/*                                  */
/* Pass the command to VSE          */
/*                                  */
command
rc = GETMSG(msg.,'RESP','USCHI',,5)
/*                                  */
/* Return the web page headings     */
/*                                  */
x=HTML('<HTML><HEAD><TITLE>')
x=HTML('VSE Console Command Processor')
x=HTML('</TITLE></HEAD>')
x=HTML('<BODY TEXT="#993300" BGCOLOR="#66FF99">')
x=HTML('<CENTER><H2><B><I><FONT COLOR="#000000">')
x=HTML('VSE Console Command Processor')
x=HTML('</FONT></I></B></H2></CENTER><P><HR>')
x=HTML('<FORM METHOD=GET  ACTION="VSECOM">')
x=HTML('Input:')
x=HTML('<INPUT TYPE="text" NAME="COMMAND" SIZE=25>')
x=HTML('<BR><HR>')
x=HTML('<FONT COLOR="#000066"><PRE>')
/*                                  */
/*  Insert the console response     */
/*                                  */
i = 1
do while i <= msg.0
   x=HTML(msg.i)
   i=i+1
   end
/*                                  */
/*  Insert the web footer           */
/*                                  */
x=HTML('</BODY></HTML>')
/*                                  */
/* Deactivate the console and exit  */
/*                                  */
ADDRESS CONSOLE 'DEACTIVATE CGICONR'
exit
/+
```