IBM GLOBAL SERVICES

E17

REXX Socket Programming in VSE and z/VM

Stephen Gracin

zSeries Expo | Nov. 1 - 5, 2004

Miami, FL

RETURN TO INDEX

VSE/VM Sockets With REXX

Connect To zVM Linux

Stephen Gracin
Gracinsp@us.ibm.com
ATS – 09/12/2004

Introduction

This Presentation

Discuss Sockets and REXX programs that use Sockets to communicate with each other. There are Socket Servers and Clients and a Perl Client on Linux. Look at the IP network that supports the environment with enough detail so that there will be an understanding of how the IP addresses and PORT numbers relate to each other.

Why REXX

- Easy language to work with and understand
- Universal, available with VSE, zVM, zOS
- There is a REXX complier available
- Can build test cases quickly

Agenda

- What is a Socket
- Programming model
- Socket calls
- VSE Socket Interfaces
- Calls In a program
- Rexx – VSE zVM

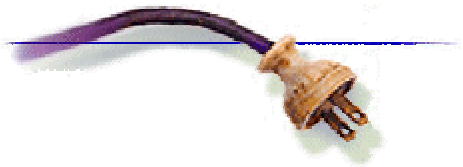
What's A Socket

- A receptacle into which a plug can be inserted.

- In this case a Socket is a software object that connects our application program to a network protocol.

  Opening a Socket allows our application programs to receive and send TCPIP messages by writing and reading from this Socket.

  The application programmer only needs to manage the Socket the operating system manages sending the messages through the network correctly.

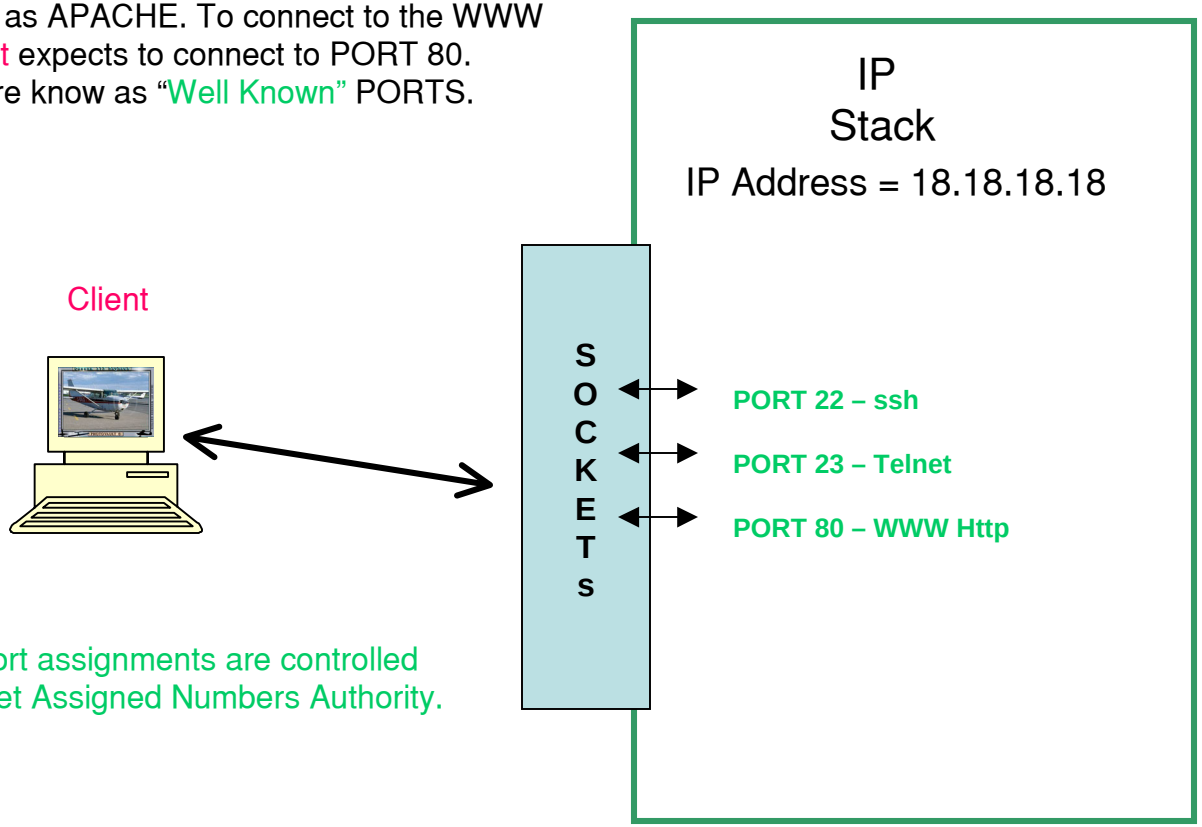


What's A Socket

- A Communications End point.
- A PORT where your program can plug into your IP stack. Sockets are assigned PORT numbers for addressing.
- Client Sockets Connect To Server Sockets
- Server Programs may Use Well-Known PORT Numbers Examples are HTTP (80) , TELNET (23) . To see a list of PORT assignments if you have a Linux system look at the ***/etc/services*** file.

What's A Socket

A client can reach a particular service at A HOST by reaching its SOCKET. At 18.18.18.18 the Socket with PORT 80 is controlled by a Http server such as APACHE. To connect to the WWW a Browser at the client expects to connect to PORT 80. Predefined PORTS are know as "Well Known" PORTS.



"Well Known" port assignments are controlled By IANA, Internet Assigned Numbers Authority.

Socket Calls

VSE/ESA and zVM have REXX Socket functions that look very similar to the *BSD socket interface* for the C programming language. VSE has two interfaces Available. Will concentrate on the C like interfaces.

Primary Socket Calls

These are the primary calls used by Socket communications.

Socket – (Used by both Server & Client) Returns a pointer to the sockets data structure. Used on subsequent calls, there is no PORT or IP address assigned at first.

Bind - (Server) Once the Server has a Socket this assigns a PORT number and IP address to it, creating an end point.

Connect – (Client) Used to establish a TCP connection to a Server, you provide the IP address and PORT number of the Server.

Listen – (Server) Sets a socket into passive mode, it is ready to accept an incoming connection.

Primary Socket Calls

Accept – (Server) Accepts a Connect request. Creates a new socket for the request so the Server can continue to listen on the original socket for more Client requests.

Recv – (Server & Client) TCP only. Receives incoming data when available.

Send – (Server & Client) TCP only. Sends outbound data, interfaces to the local stack.

Close – (Server & Client) After sending/receiving all data terminate Server/Client communications and deallocate the socket.

Utility Calls

These are calls to help manage the environment

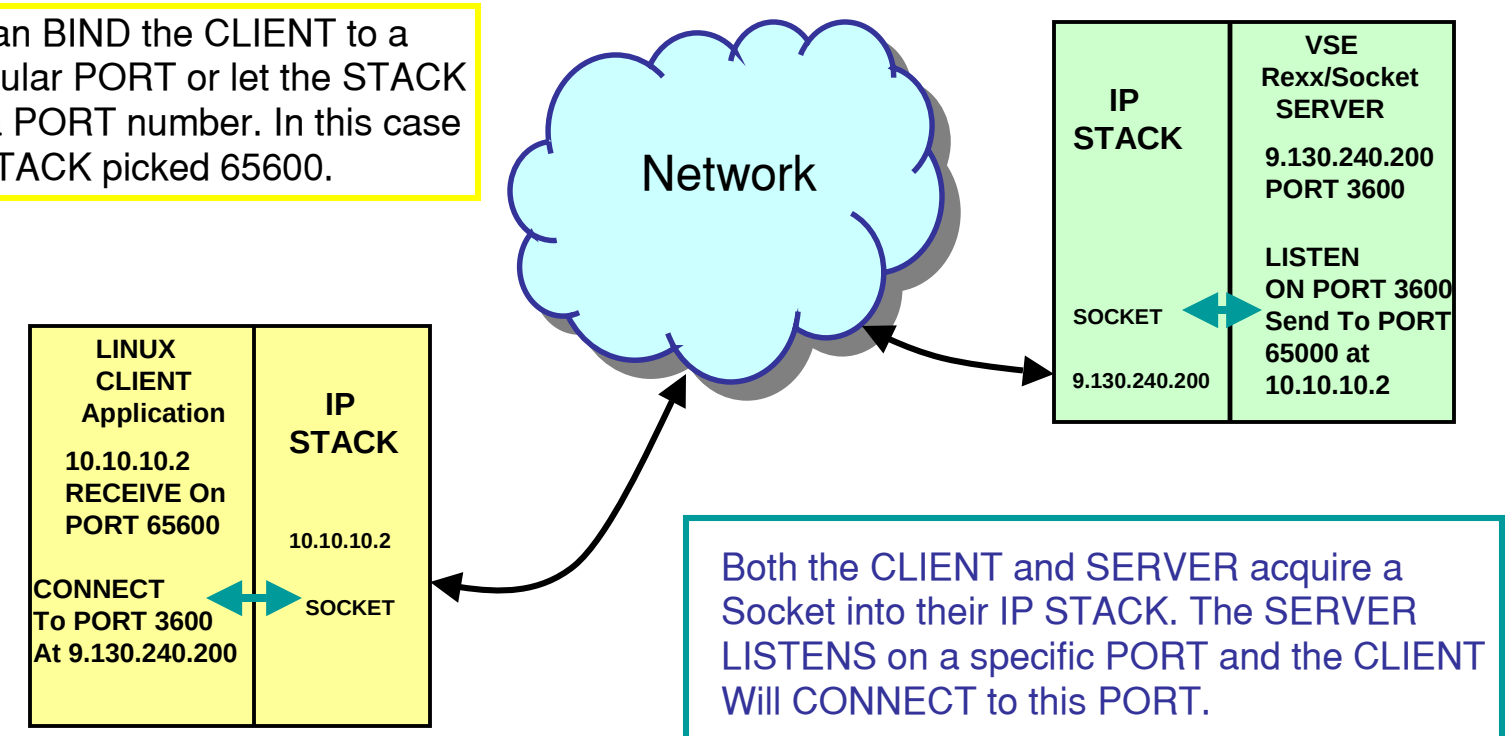
- Information lookup, IP addresses, PORT numbers , protocols.
- Format IP Addresses
- Socket options and configuration.
- Multiple open sockets, file system calls, signals, events.

Note: Socket and Utility calls may vary from operating system to system as to what options are available. Check the documentation for the specific stack you are working with.

Make a Connection

Client To Server

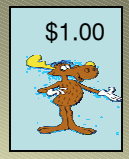
We can BIND the CLIENT to a Particular PORT or let the STACK pick a PORT number. In this case the STACK picked 65600.



Both the CLIENT and SERVER acquire a Socket into their IP STACK. The SERVER LISTENS on a specific PORT and the CLIENT Will CONNECT to this PORT.

Connections

Client
10.10.10.2
PORT 65600



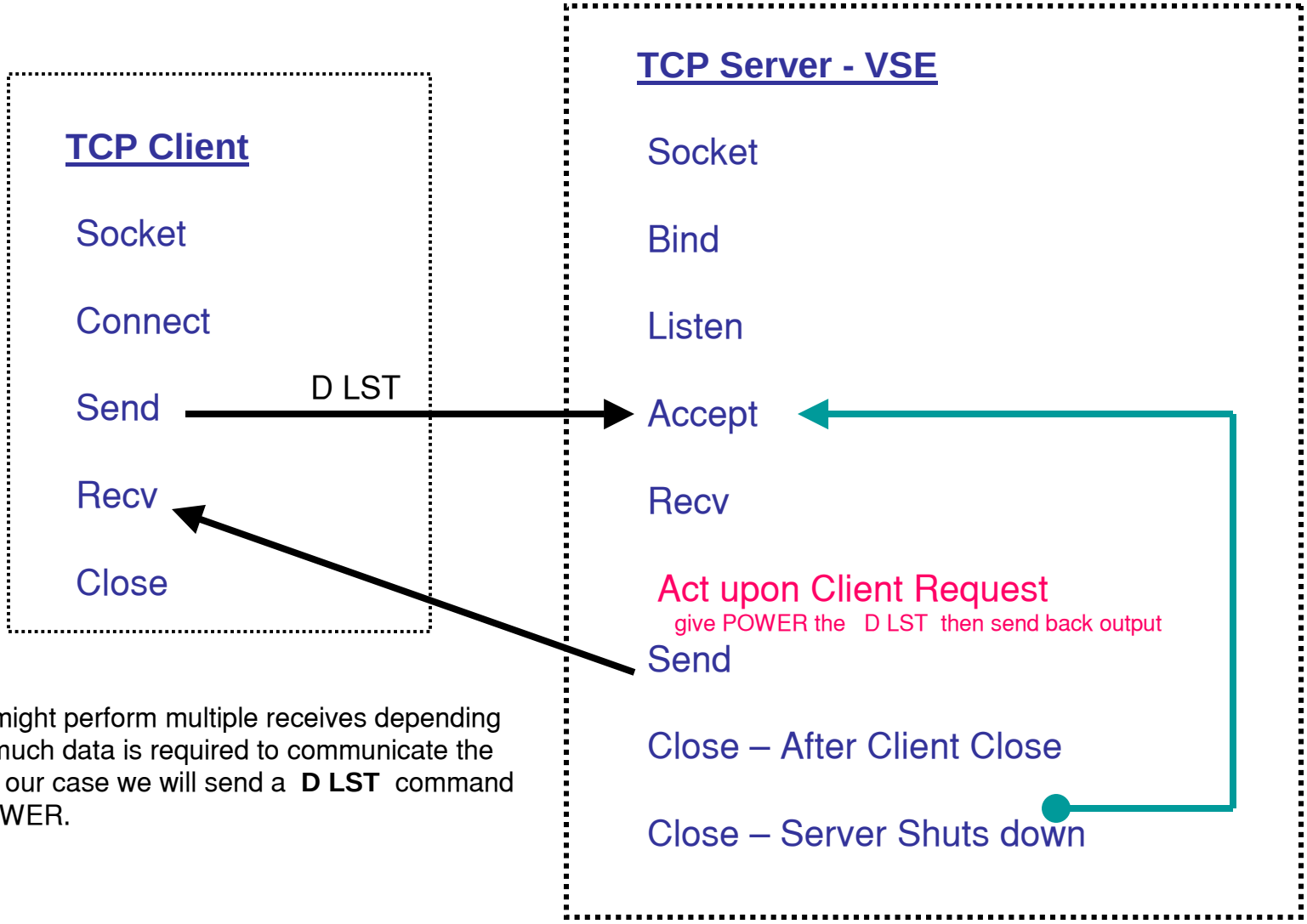
Server
9.130.240.200
PORT 3600

Information needed to send a message and
Receive a response .

The Model

- Servers are usually started first and keep running.
- Clients are started after the server, usually on demand
- Servers listen for a connection request, clients originate a connection request
- Servers may be single or multiple threaded, clients are almost always single threaded
- One server handles requests from many clients

Socket Calls In a Program



The Client might perform multiple receives depending Upon how much data is required to communicate the Request. In our case we will send a **D LST** command To VSE/POWER.

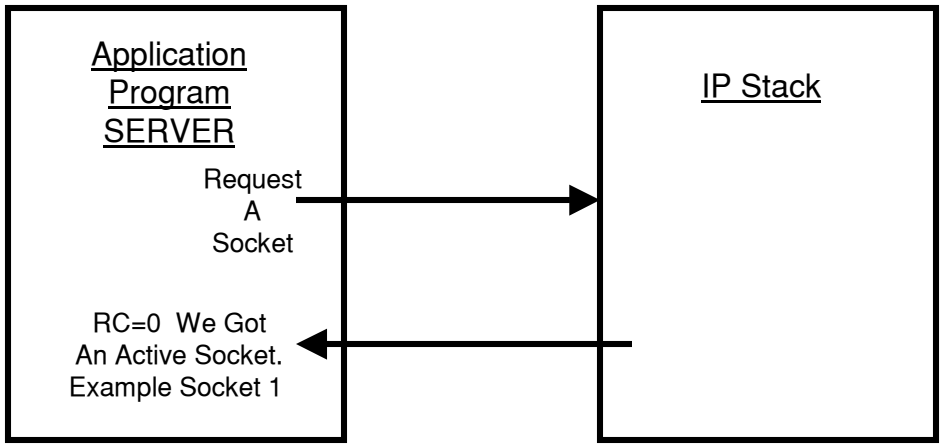
Socket Calls In a Program

Made With REXX

“Plug into” The Network

Get a SOCKET to plug into our TCPIP stack, something both Servers and Clients will do.

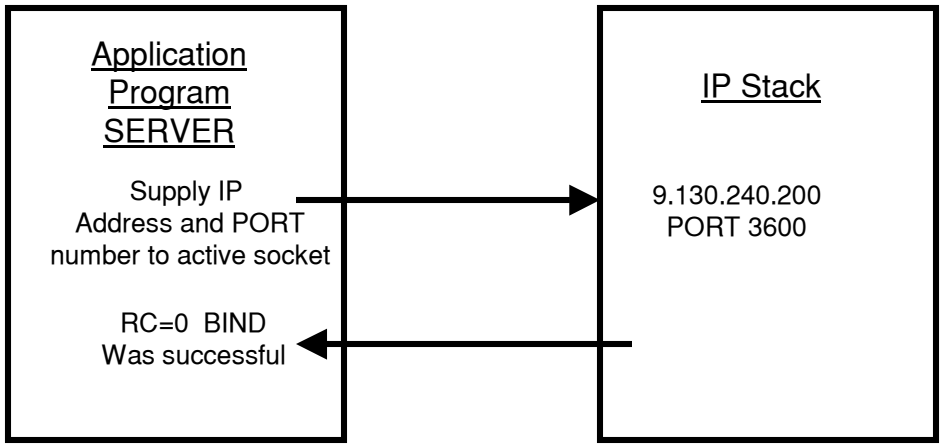
```
SOCKET('SOCKET','AF_INET','SOCK_STREAM','IPPROTO_TCP');
```



BIND To Network

BIND our socket to a IP Address and PORT number. We will be found in the network at this IP address and in the stack and this port number.

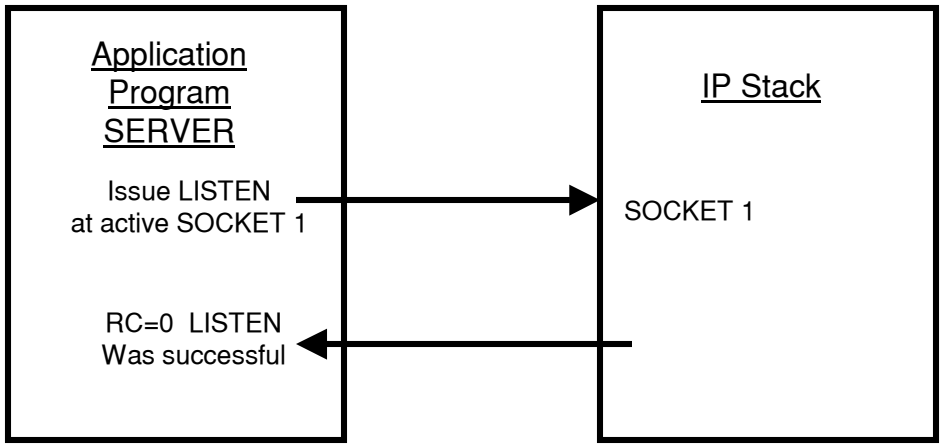
```
SOCKET('BIND',ACTIVESOCKET,' 2 ' PORTID HOSTADDR);
```



LISTEN For Request

LISTEN sets up our socket to passively wait for an inbound request.

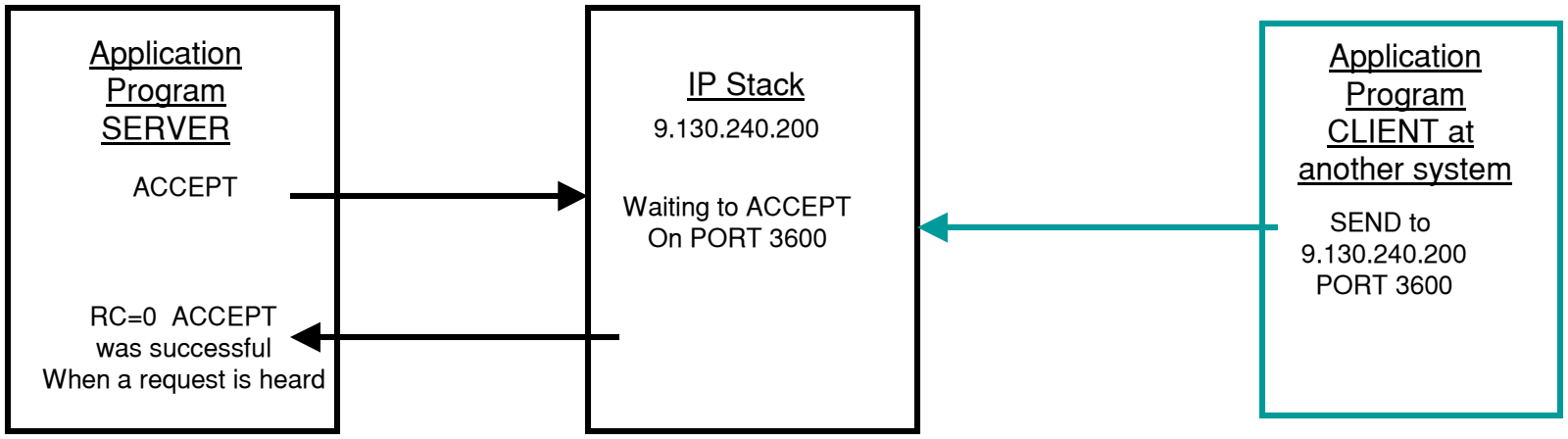
```
SOCKET('LISTEN',ACTIVESOCKET, 10 );
```



ACCEPT Inbound

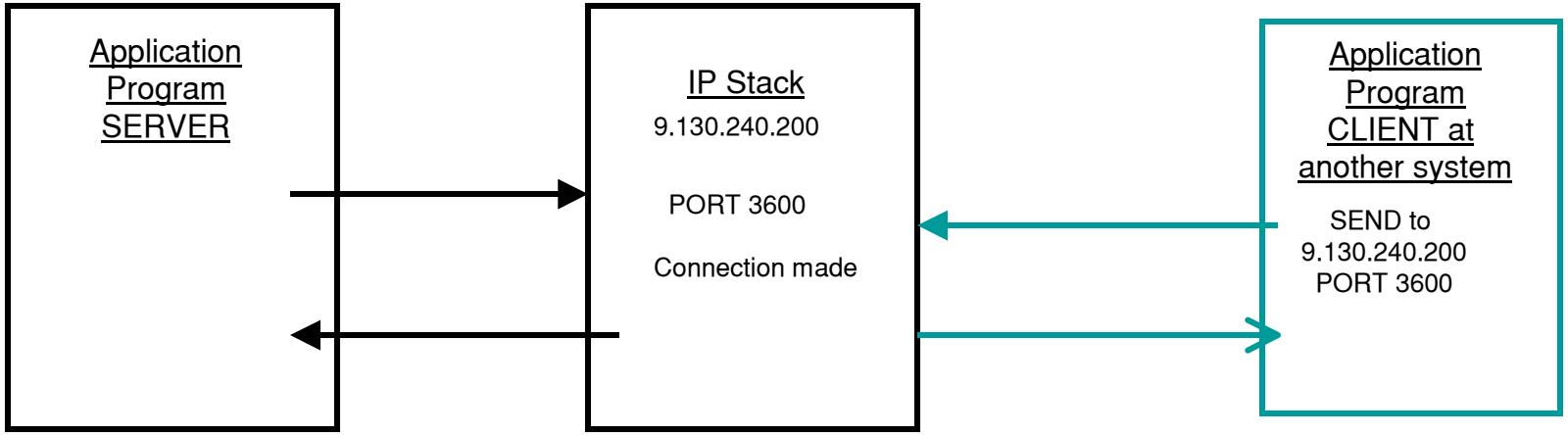
ACCEPT an inbound request , the server will sit here waiting on an inbound request to occur.

```
SOCKET('ACCEPT',ACTIVESOCKET);
```



Request Received

The request was heard and a connection made the application program SERVER is in communication with CLIENT program on another system, somewhere on the WWW. The CLIENT can send request for some action and the SERVER will send results back.



REXX

Socket Access

- In VSE/ESA and zVM

  There is support for multiple programming languages to access the Socket interface. The access can be made by C , Assembler or REXX in VSE/ESA.

  In zVM it can be C or REXX socket calls, there is a PASCAL interface to TCPIP.

- Will use only the REXX interface for examples.

Socket Calls With REXX

- VSE can have two REXX Socket interfaces available. One provided by TCPIP For VSE by CSI and the other requires VSE/LE be installed as it uses C programs to interface with IP. To use both requires some tailoring.

- zVM/REXX has one Socket interface it looks like the VSE/LE interface.

- The VSE/LE REXX Socket calls and the zVM calls are almost identical, they follow the BSD C programming language Calls, compare VSE LE SERVER to the zVM SOCKSVR. The TCPIP For VSE REXX Calls are not like the BSD C calls.

VSE/REXX

- SOCKET.PHASE provided by TCPIP For VSE in PRD1.BASE that allows VSE/REXX to employ a socket interface if LE is not installed.

- If both are installed and you wish to use the VSE/REXX C implementation You must get ARXPARMS.Z from PRD1.BASE make changes then assemble. See the VSE/REXX Reference and be sure PQ90235 is applied.

- This allows a REXX program Call either ***SOCKET*** or ***SOCKEN***. When a program Calls ***SOCKET*** the TCPIP for VSE interface is used, when ***SOCKEN*** is Called the LE interface C programs to REXX are used.

Socket Calls With VSE/REXX

```
CALL SOCKEN 'INITIALIZE','SERVSE1';
```

This call is handled by IBM supplied LE C interface

```
RC=SOCKET(TYPE,'OPEN',LOCP,,,SYSID,TIMEOUT,ASYNC,MODE);
```

This call is handled by TCPIP For VSE Socket Phase

Socket Calls With VSE/REXX

Two socket servers running in VSE/ESA.

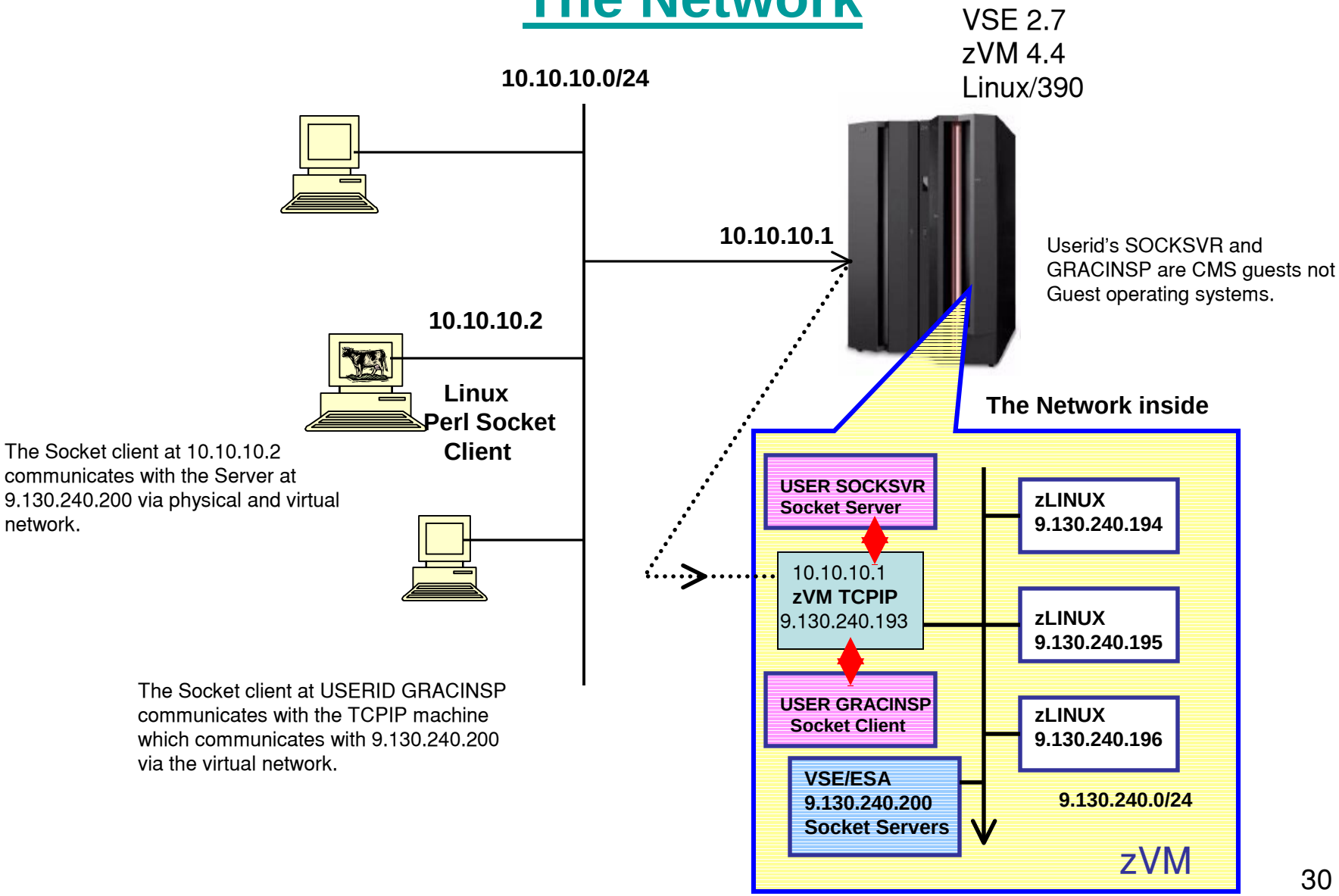
```
F9 0009 // JOB SVASCII
        DATE 07/14/2004, CLOCK 11/38/02
F9 0009 SOC015I Rexx Sockets 01.05 A  (Prod) - 12/23/02 : 13.13
F9 0009 SOC016I Copyright (c) 1998, Connectivity Systems, Inc.
FA 0001 1Q47I   FA SVEBCDIC 07425 FROM (GRAC) , TIME=11:38:04
FA 0010 // JOB SVEBCDIC
        DATE 07/14/2004, CLOCK 11/38/04
FA 0010 0 3600 9.130.240.200
```

Partition F9 is the *TCPIP For VSE* interface (socket) partition FA uses the *LE* interface (socken), in this case.

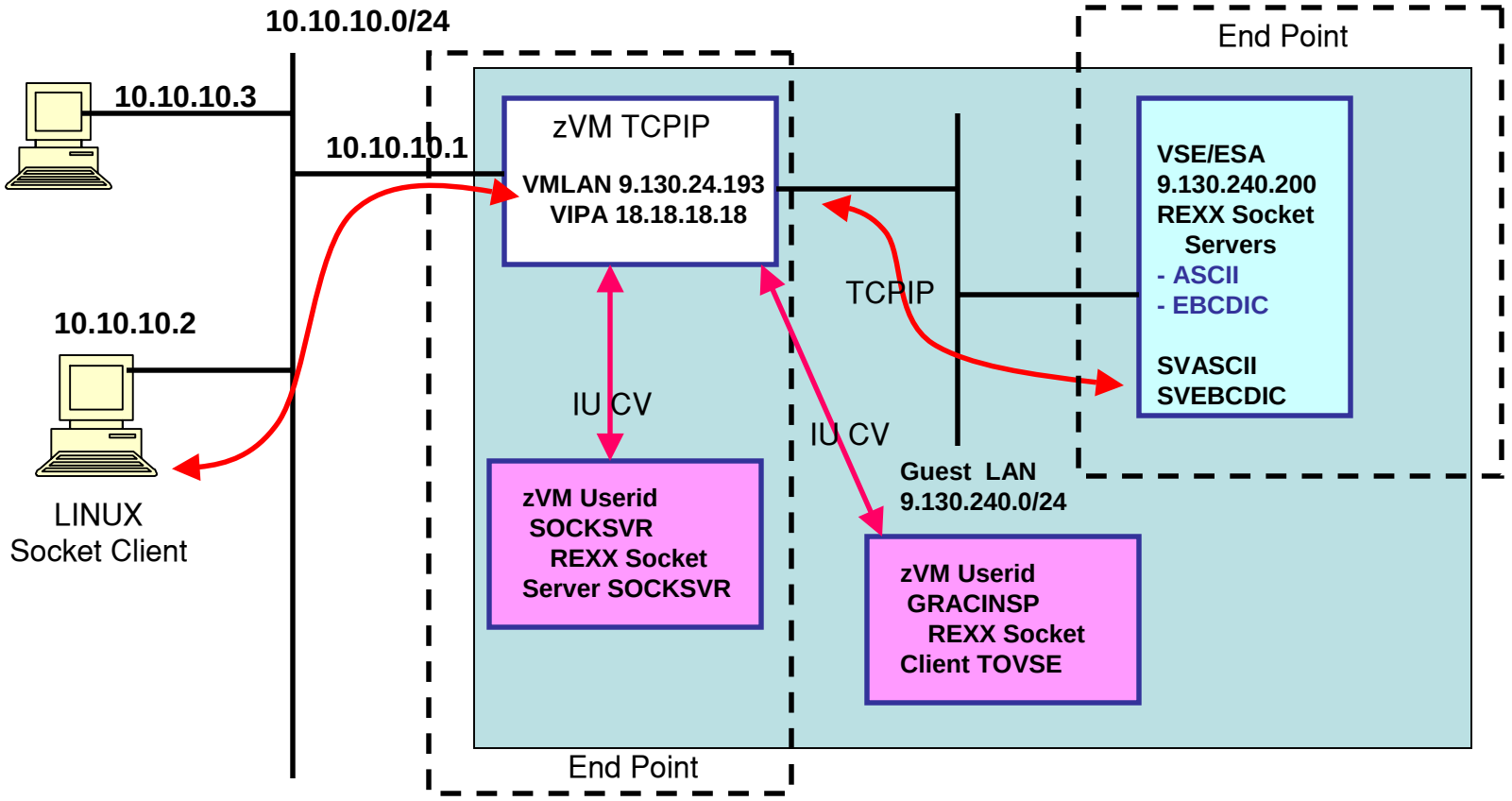
The Network

Where our Servers and Clients are..

The Network


VSE 2.7
zVM 4.4
Linux/390
10.10.10.0/24
10.10.10.1
Userid's SOCKSVR and GRACINSP are CMS guests not Guest operating systems.
10.10.10.2
Linux Perl Socket Client
The Socket client at 10.10.10.2 communicates with the Server at 9.130.240.200 via physical and virtual network.
The Network inside
USER SOCKSVR Socket Server
10.10.10.1 zVM TCPIP 9.130.240.193
USER GRACINSP Socket Client
VSE/ESA 9.130.240.200 Socket Servers
zLINUX 9.130.240.194
zLINUX 9.130.240.195
zLINUX 9.130.240.196
9.130.240.0/24
zVM
The Socket client at USERID GRACINSP communicates with the TCPIP machine which communicates with 9.130.240.200 via the virtual network.

The Network

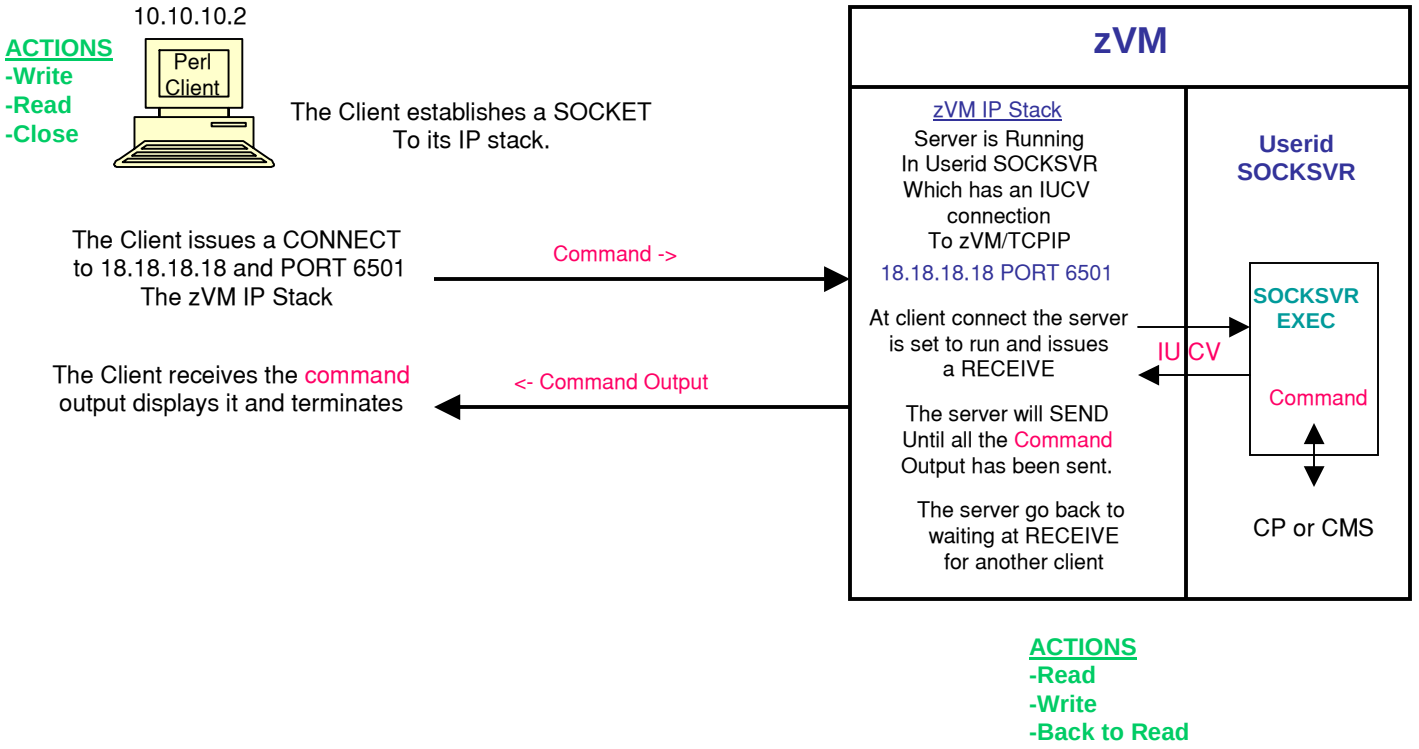


A Socket Client at 10.10.10.2 uses IP to communicate to zVM/TCPIP which in turn uses IUCV to communicate with the REXX Socket Server running in zVM Userid SOCKSVR, zVM is the End Point. The same Socket Client uses IP all the way to VSE/ESA REXX Socket Server, to VSE/ESA on a Guest LAN the zVM/TCPIP stack is its router.

VM Socket Server

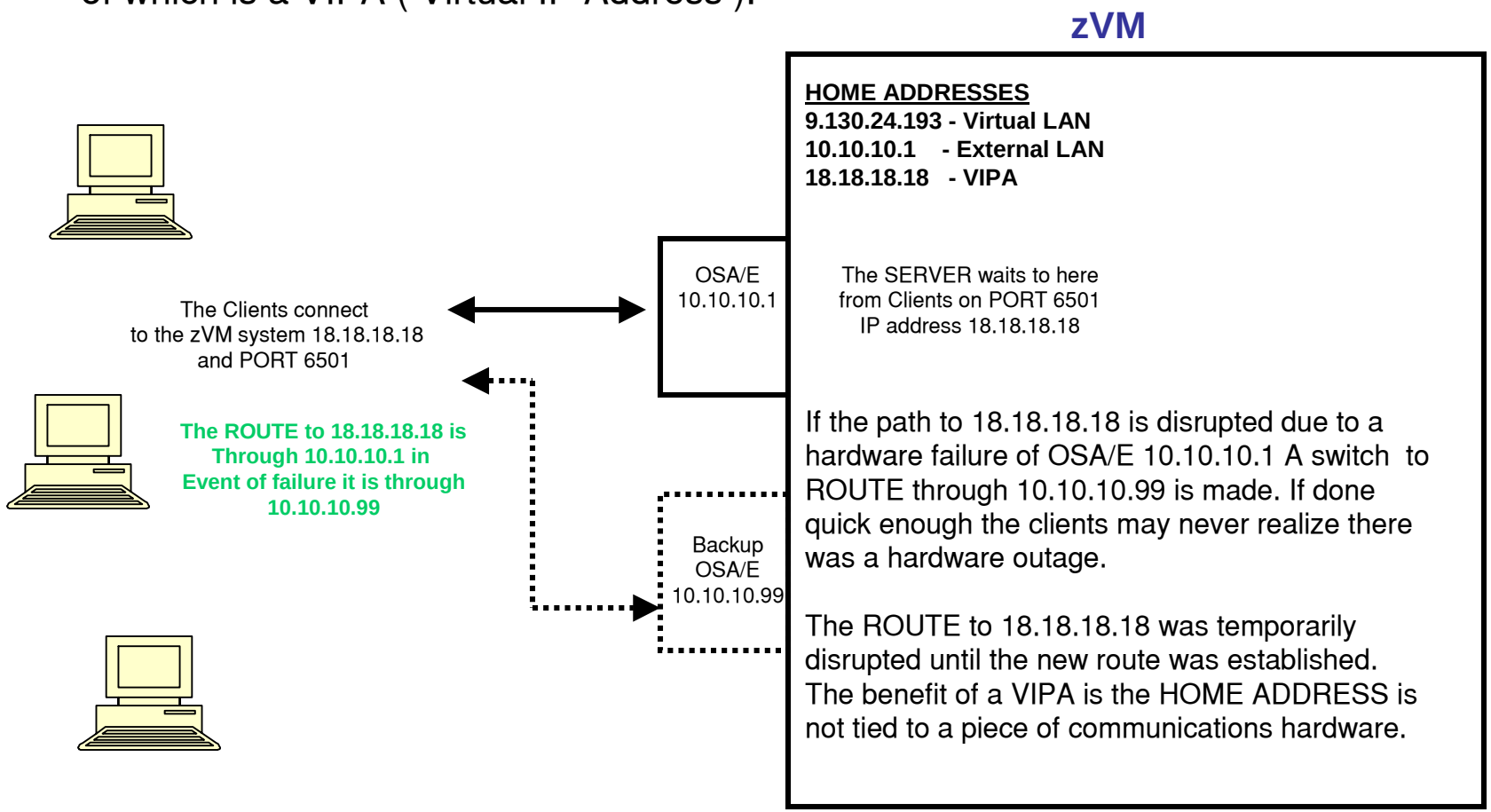
zVM Socket Server

A Perl socket client on Linux will connect to a Socket Server using the TCPIP For VSE REXX interface. To send and execute a VSE/POWER command and send the results back to Linux.



VM Has a VIPA

zVM is the End Point, the zVM stack will have multiple HOME ADDRESSES one of which is a VIPA (Virtual IP Address).



VSE Servers

VSE Socket Servers

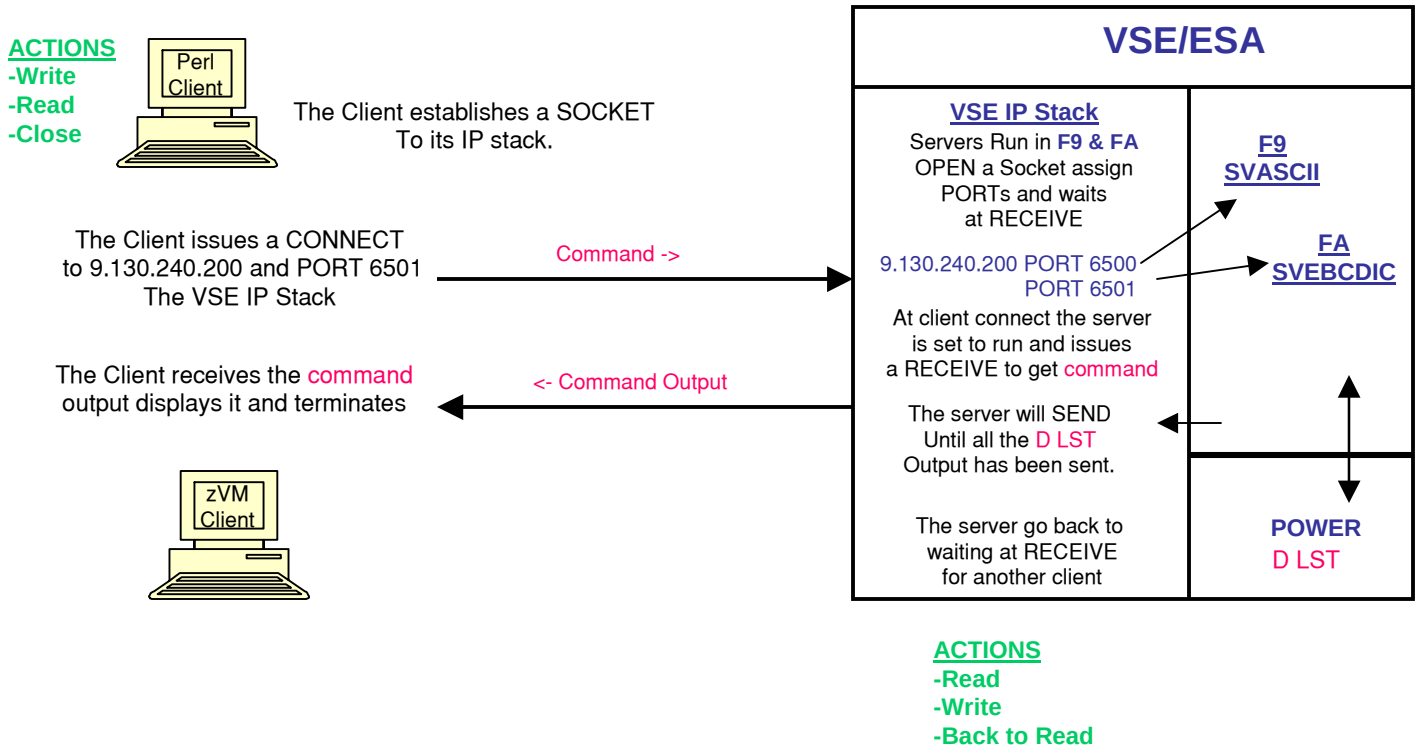
Client machines will send a command to VSE/POWER and receive the command output
And display it. Have two different client types that can CONNECT to VSE they are ASCII client
in this case Linux or an EBCDIC client in this case zVM.

So a scheme to separate the traffic so that proper translation is done as necessary. The
Procedure will be two separate Servers at IP address 9.130.240.200 . Each Server obtains a
Socket and BINDs it with different PORT numbers. 3600 for ascii Server SVASCII 3601 for ebcdic
Server SVEBCDIC.

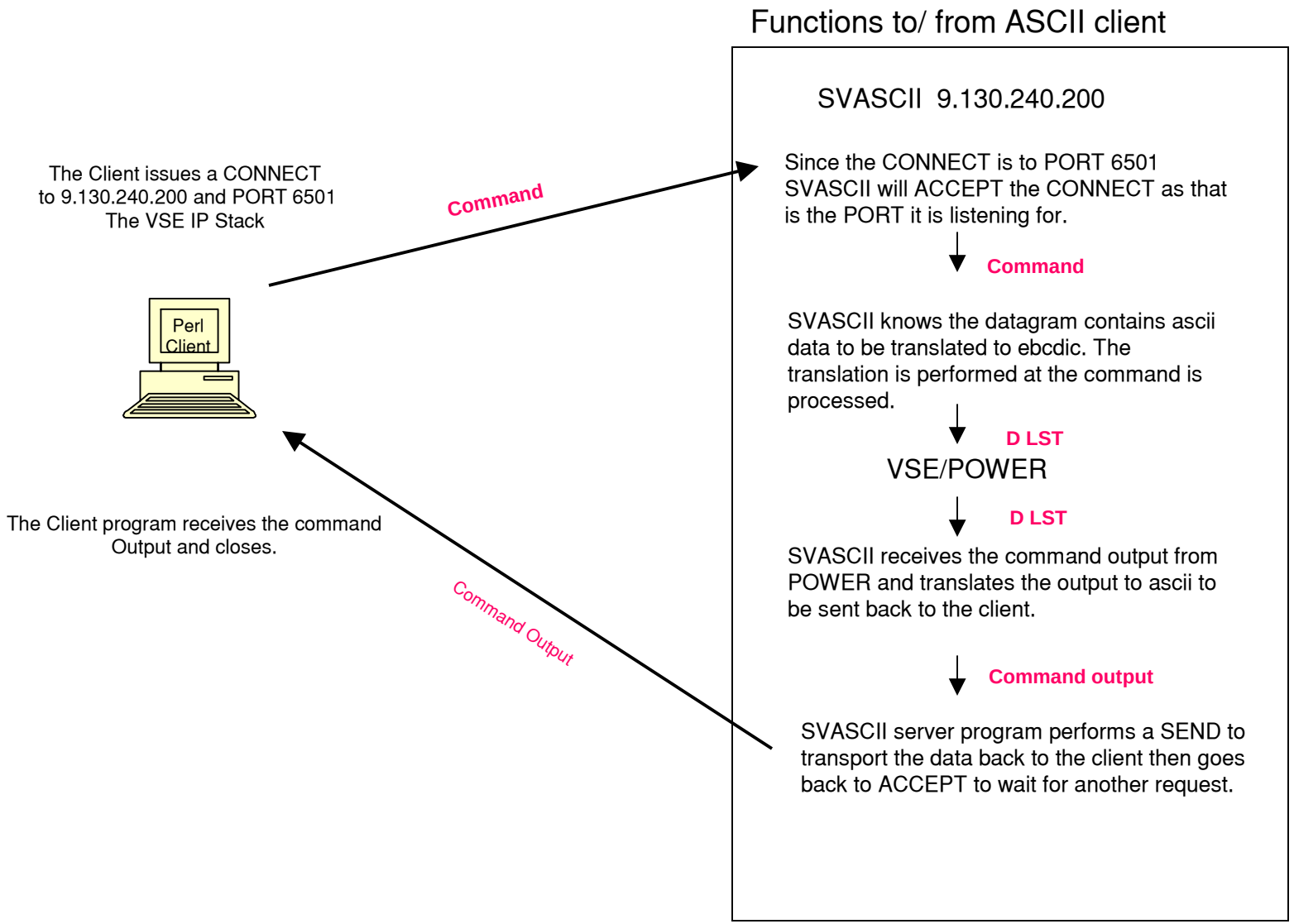
This scheme of two servers with different PORT numbers is not the only way to accommodate
ascii and ebcdic clients.

VSE Socket Servers

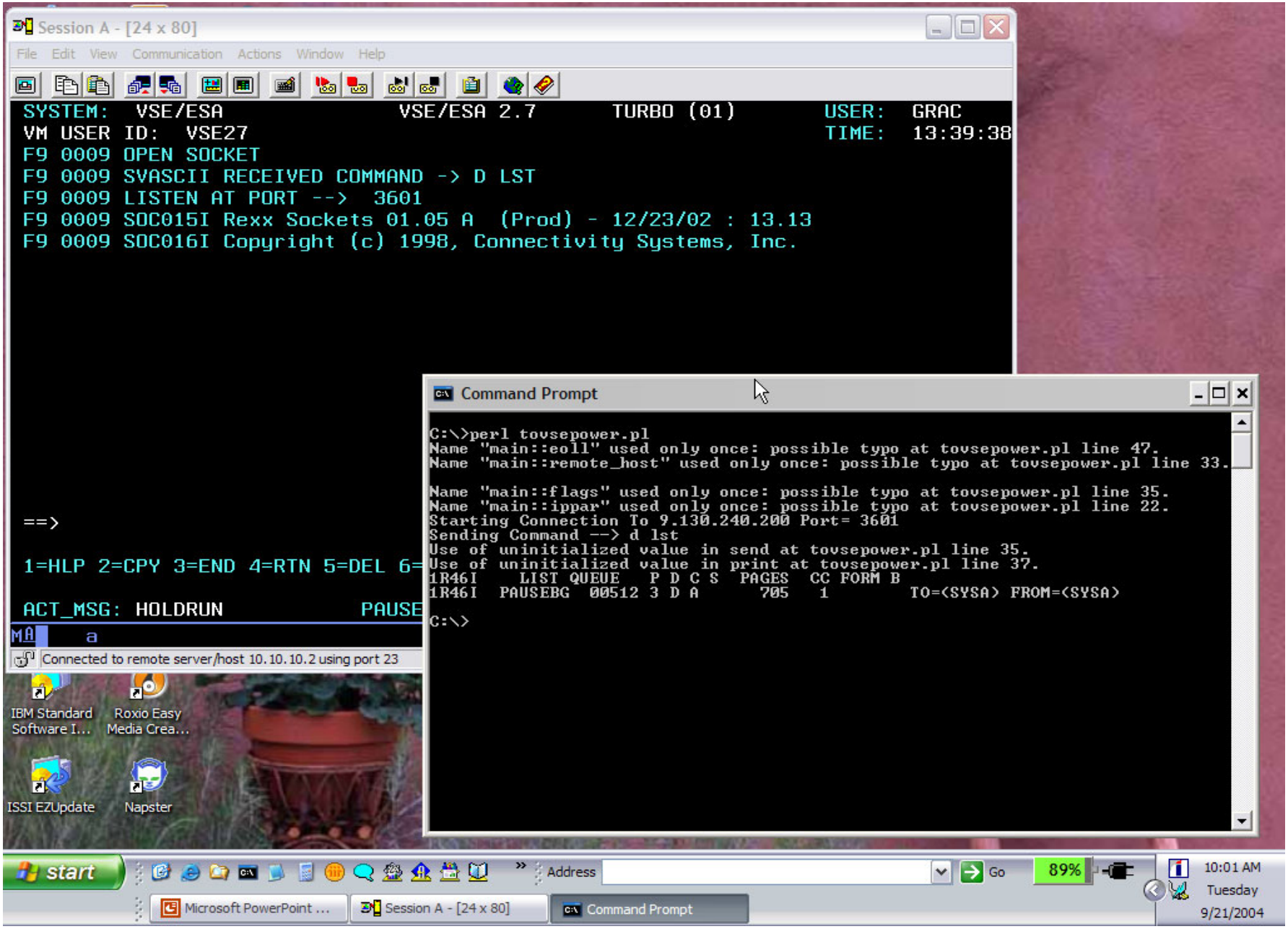
A Perl socket client on Linux will connect to a Socket Server using the TCPIP For VSE REXX interface. To send and execute a VSE/POWER command and send the results back to Linux.



VSE Socket Servers



VSE Socket Servers



Clients & Servers

VSE LE C Interface Server

```
* $$ JOB JNM=SVEBCDIC,CLASS=O
// JOB CATALOG SVEBCDIC
// EXEC LIBR,PARM='MSHP'
A S=IJSYSRS.SYSLIB
CATALOG    SVEBCDIC.PROC       REPLACE=YES
/* ============================================ */
/* THIS EXEC WILL EXPECT EBCDIC DATA            */
/* AS IT LISTENS ON PORT 3600                   */
/* SPG 7/9/2004                                 */
/* ============================================ */
TRACE O;
CALL ASSGN 'STDOUT',SYSLOG;
PORTID="3600";
/* ============================================ */
/*  I N I T A L I Z E                           */
/* ============================================ */
/*  INITALIZE WILL RETURN:                      */
/*   RC - RETURN CODE                           */
/*   SUBTASK - MY NAME =  SERVSE1               */
/*   USE   SOCKEN   NOT   SOCKET                */
/*   AS THIS IS THE IBM REXX C INTERFACE        */
/*   SOCKETSALLOC - NUMBER OF SOCKETS PREALLOCAT */
/*   MACHINENAME - THIS IS USERID OF IP (TCPIP) */
/*                                              */
/*   PASSED: MY SUBTASK NAME ( SERVSE1 )        */
/* ============================================ */
CALL SOCKEN 'INITIALIZE','SERVSE1';
/* ============================================ */
/*  S O C K E T                                 */
/* ============================================ */
/*  SOCKET WILL RETURN:                         */
/*      RC - RETURN CODE                        */
/*      ACTIVESOCKET - WHICH SOCKET NUMBER IS   */
/*                     ACTIVE                   */
/*                                              */
/*  PASSED: AF_INET - USE DOTTED IP ADDRRESSES  */
/*          SOCK_STREAM - STREAM                */
/*          IPPROTO_TCP - PROTOCOL IS TCP       */
/* ============================================ */
SOCKETDATA=SOCKEN('SOCKET','AF_INET','SOCK_STREAM','IPPROTO_TCP');
PARSE VALUE SOCKETDATA WITH RC ACTIVESOCKET;
  IF RC /= O THEN DO;
   SAY 'RETURN CODE CAME BACK =' RC;
   SAY 'ON   SOCKET ';
SAY 'EXITING PROGRAM';
  CALL TERMSOCKET;
END;
/* ============================================ */
/* WILL WANT THIS FOR A FYI ON SCREEN           */
/* ============================================ */
HOSTADDRS=SOCKEN('GETHOSTID');
HOSTADDR=WORD(HOSTADDRS,2);
/* ============================================ */
/*  B I N D                                     */
/* ============================================ */
/*  BIND WILL RETURN:   RC - RETURN CODE        */
/*                                              */
/*  PASSED: ACTIVESOCKET                        */
/*          DOMAIN - FOR AF_INET IT'S 2         */
/*          PORTID - PORT NUMBER I WILL USE     */
/*          HOSTADDRS - MY IP ADDRESS           */
/* ============================================ */
BINDDATA=SOCKEN('BIND',ACTIVESOCKET,' 2 ' PORTID HOSTADDR);
SAY ACTIVESOCKET PORTID HOSTADDR
IF BINDDATA /= O THEN DO;
SAY;
  SAY 'SOCKET BIND  FAILED .....';
  SAY;
  SAY 'EXITING PROGRAM';
  SIGNAL TERMALL;
  END;
/* ============================================ */
/* L I S T E N                                  */
/* ============================================ */
/* LISTEN WILL RETURN RC                        */
/*                                              */
/*   PASSED: SOCKET TO LISTEN ON - ACTIVESOCKET */
/* ============================================ */
LISTENDATA=SOCKEN('LISTEN',ACTIVESOCKET, 10 );
  SAY LISTENDATA;
  IF LISTENDATA /= O THEN DO;
  SAY;
  SAY ' SOCKET LISTEN FAILED .. ';
  SAY;
  SAY ' FUNCTION RETURNED ' LISTENDATA;
  SAY;
```

VSE LE C Interface Server

```
SIGNAL TERMSOCKET;
  END;
/* ============================================ */
/* A C C E P T                                  */
/* ============================================ */
/* ACCEPT RETURNS: RC                           */
/*        NEW_SOCKET - SOCKET CLIENT IS HEARD ON */
/*        DOMAIN                                */
/*        PORT - PORT CLIENT IS USING           */
/*        IPADDRESS - ADDRESS OF CLIENT         */
/*                                              */
/* WILL WAIT HERE FOR INBOUND DATA              */
/* ============================================ */
REACCEPT:
PASS='INITIAL';
SAY 'ADDRESS USED FOR SOCKET = 'HOSTADDR;
RC = 0;
ACCEPTDATA=SOCKEN('ACCEPT',ACTIVESOCKET);
PARSE VALUE ACCEPTDATA WITH RC NEW_SOCKET DOMAIN PORT IPADDRESS;
SAY  RC NEW_SOCKET DOMAIN PORT IPADDRESS;
 IF RC /= 0 THEN DO;
 SAY;
   SAY ' SOCKET ACCEPT FAILED .. ';
   SAY;
   SAY ' FUNCTION RETURNED ' ACCEPTDATA;
   SAY;
   SIGNAL TERMALL;
  END;
 /* ============================================ */
 /* R E C V                                      */
 /* ============================================ */
 /* RECV RETURNS: RC                             */
 /*               READLENGTH - LENGTH OF DATA    */
 /*               READDATA - DATA                */
 /* ============================================ */
 REREAD:
 RC = 0;
 READDATA=SOCKEN('RECV',NEW_SOCKET);
 PARSE VALUE READDATA WITH RC READLENGTH READDATA;
  IF RC /= 0 THEN DO;
      IF PASS='INITIAL' THEN DO;
       SAY;
```

```
SAY ' SOCKET READ FAILED .. ';
      SAY;
      SAY ' FUNCTION RETURNED ' READDATA;
      SAY;
      SIGNAL TERMALL;
     END;
 END;
 IF RC /= 0 THEN DO;
      SAY;
      SAY ' NO MORE DATA FROM.. 'IPADDRESS ' PORT ='
PORT;
      SAY;
 END;
IF RC = 0 THEN PASS='NOTINITIAL';
/* ============================================ */
/* IF RC NOT EQUAL 0 THEN THE OTHER END CLOSED  */
/* THIS SOCKET CONNECTION SO CLOSE THIS END     */
/* AND GO BACK AND ACCEPT AGAIN.                */
/* ============================================ */
  IF RC /= 0 THEN DO;
   CLOSEDATA=SOCKEN('CLOSE', NEW_SOCKET);
  END;
 IF RC /= 0 THEN SIGNAL REACCEPT;
 /* ============================================ */
 /* SEND COMMAND TO POWER                        */
 /* ============================================ */
  RINC=1;
  CMD_UC=TRANSLATE(READDATA);
  IF CMD_UC /= "SHUTDOWNSHUTDOWN" THEN DO;
  SAY "SVASCII RECEIVED COMMAND -> "CMD_UC;
  END;
  IF CMD_UC = "SHUTDOWNSHUTDOWN" THEN SIGNAL TERMALL;
  PTRAP=OUTTRAP(PRET.);
  ADDRESS POWER CMD_UC;
       IF RC /= 0 THEN SAY ERROR;
       DO I=1 TO PRET.0
       REC.RINC=PRET.I
       RINC=RINC+1;
       END;
 /* ============================================ */
  RCTR=RINC;
  RINC=1;
 DO (RCTR-1);        /* SEND THEM ALL            */
```

VSE LE C Interface Server

```
 BUFFER=REC.RINC;
 BUFFER=INSERT(BUFFER," <=EOL=>");
/* ============================================= */
/* SEND                                          */
/* ============================================= */
    SENDDATA=SOCKEN('SEND',NEW_SOCKET,BUFFER);
    PARSE VALUE SENDDATA WITH RC SENDLENGTH SENDFILE;
     IF RC /= 0 THEN DO;
      SAY;
      SAY ' SOCKET SEND FAILED .. ';
      SAY;
      SAY ' FUNCTION RETURNED ' SENDDATA;
      SAY;
/* ============================================= */
/* CLOSE SOCKET IF A SEND FAILS WHILE WE STILL   */
/*       HAVE DATA.                              */
/* ============================================= */
     CLOSEIT=SOCKEN('CLOSE',NEW_SOCKET);
     END;
  RINC=RINC+1;
END;
/* ============================================= */
/* ALL DATA SENT - CLOSE SO CLIENT KNOWS WE ARE  */
/*                 FINISHED                      */
/* ============================================= */
     CLOSEIT=SOCKEN('CLOSE',NEW_SOCKET);
SIGNAL REREAD;
/* ============================================= */
/* WE'RE OUTTA HERE ..                           */
/* ============================================= */
TERMALL:
CLOSEIT=SOCKEN('CLOSE',ACTIVESOCKET);
TERM=SOCKEN('TERMINATE','SERVSE1');
EXIT;
/+
/*
/&
* $$ EOJ
```

VSE CSI Server

```
 * $$ JOB JNM=SVASCII,CLASS=O
 // JOB CATALOG SVASCII EXEC
 // EXEC LIBR,PARM='MSHP'
 A S=IJSYSRS.SYSLIB
 CATALOG SVASCII.PROC    REPLACE=YES
 /* ========================================== */
 /* THIS EXEC EXPECTS TO RECEIVE ASCII DATA    */
 /* AS IT LISTENS ON PORT 36Ø1                 */
 /* ========================================== */
 /*  I N I T A L I Z E                         */
 /* ========================================== */
 TRACE O;
 CALL ASSGN 'STDOUT',SYSLOG;
 CALL TABLES;
 XTABLE=INSERT(X1,X2);
 XTABLET=INSERT(XTABLE,X3);
 ATABLE=INSERT(A1,A2);
 ATABLET=INSERT(ATABLE,A3);
 CTABLE=INSERT(C1,C2);
 CTABLET=INSERT(CTABLE,C3);
 TYPE='TCP';
 LOCP='3601';
SYSID='OO';
TIMEOUT='3600';
ASYNC='N';
MODE='SERVER';
/* ========================================== */
RECEIVE_DATA:
   RC=SOCKET(TYPE,'OPEN',LOCP,,,SYSID,TIMEOUT,ASYNC,MODE);
  IF RC = O THEN SAY "Open Socket";
  IF RC /= O THEN DO;
  SAY "Did Not Get a Socket -- Terminating";
  SAY "Return Code = "RC;
  EXIT;
  END;
/* ========================================== */
/* CAME IN ON ASCII RECEIVE PORT              */
/* CONVERT FROM ASCII/BINARY TO ASCII/HEX     */
/* ========================================== */
RC=SOCKET(HANDLE,'RECEIVE',TIMEOUT);
CHARX=C2X(BUFFER);
INHEX=' ';
BLEN=LENGTH(BUFFER);
```

```
  /* ========================================== */
  /* CONVERT ASCII HEX TO EBCDIC HEX            */
  /* ========================================== */
  PTR=1;
  APTR=1;
  F_CX=" ";
  DO BLEN;
     FIND_CHAR=SUBSTR(CHARX,PTR,2);
        CTR=1;
         APTR=1;
            DO FOREVER;
            COMP_CHAR=SUBSTR(ATABLET,APTR,2);
                IF FIND_CHAR /= COMP_CHAR THEN DO;
                APTR=APTR+2;
                CTR=CTR+2;
                END;
                  IF FIND_CHAR=COMP_CHAR THEN DO;
                  FOUND_CHAR=SUBSTR(CTABLET,CTR,1);
                  IF F_CX /= " " THEN
F_CX=INSERT(F_CX,FOUND_CHAR);
                  IF F_CX = " " THEN F_CX=FOUND_CHAR;
                  LEAVE;
  END;
               END;
      PTR=PTR+2;
  END;
  /* ==========================================
*/
  /* SEND COMMAND TO VSE/POWER
*/
  /* ==========================================
*/
  RINC=1;
  CMD_UC=TRANSLATE(F_CX);
  IF CMD_UC /= "SHUTDOWNSHUTDOWN" THEN DO;
  SAY "SVASCII RECEIVED COMMAND -> "CMD_UC;
  END;
  IF CMD_UC = "SHUTDOWNSHUTDOWN" THEN SIGNAL SHUTDOWN;
  PTRAP=OUTTRAP(PRET.);
  ADDRESS POWER CMD_UC;
       IF RC /= O THEN SAY ERROR;
       DO I=1 TO PRET.O
       REC.RINC=PRET.I
       RINC=RINC+1;
```

VSE CSI Server

```
 END;
 /* ============================================= */
 /* DATA IS GOING TO AN ASCII MACHINE,CONVERT     */
 /* CHARACTERS TO ASCII BEFORE SENDING ...        */
 /* THIS CONVERSION BY REXX IS GOOD FOR DEMO      */
 /* PURPOSES, LOW DATA VOLUMES. FOR PRODUCTION USE */
 /* ASSEMBLER OR C. BUT AFTER ALL THIS IS A REXX  */
 /* PRESENTATION.                                 */
 /* ============================================= */
 RCTR=RINC;
 RINC=1;
 DO (RCTR-1);          /* SEND THEM ALL           */
 BUFFER=REC.RINC;
 BUFFER=INSERT(BUFFER,"<=EOL=>");
 CHAR2BA=C2X(BUFFER);
 B2BALEN=LENGTH(BUFFER);
 /* ============================================= */
 /* CONVERT FROM HEX EBCDIC TO HEX ASCII          */
 /* ============================================= */
 PTR=1;
 APTR=2;
F_CA=" ";
DO B2BALEN;
   FIND_CHARA=SUBSTR(CHAR2BA,PTR,2);
      CTR=1;
       APTR=1;
          DO FOREVER;
          COMP_CHAR=SUBSTR(XTABLET,APTR,2);
              IF FIND_CHARA /= COMP_CHAR THEN DO;
              APTR=APTR+2;
              CTR=CTR+2;
     /* IF XLATE TABLE IS WRONG/INCOMPLETE DON'T LOOP      */
     /* SET CTR TO POINT AT BLANK, SEND MESSAGE.          */
     /* NOT THE BEST BUT IT'S BETTER THAN A LOOP.         */
              IF CTR > 500 THEN DO
               SAY "======================"
               SAY "FROM SVASCII          "
               SAY "BLANK INSERTED-CHECK TRANSLATE TABLE"
               SAY "UNMATCHED CHARACTER IS EDCDIC "FIND_CHARA;
               SAY "======================"
               FIND_CHARA="40";
               COMP_CHAR="40";
```

```
CTR=1
                END;
               END;
             IF FIND_CHARA=COMP_CHAR THEN DO;
             FOUND_CHARAF=SUBSTR(ATABLET,CTR,2);
             IF F_CA /= " " THEN F_CA=INSERT(F_CA,FOUND_CHARAF);
             IF F_CA = " " THEN F_CA=FOUND_CHARAF;
             LEAVE;
             END;
           END;
    PTR=PTR+2;
END;
/* ============================================= */
/* CONVERT ASCII HEX TO                          */
/* CHARACTERS AND SEND BACK.                     */
/* ============================================= */
ACHARS_TO_SEND=X2C(F_CA);
RC=SOCKET(HANDLE,'SEND',ACHARS_TO_SEND);
RINC=RINC+1;
END;
SHUTDOWN:
RC=SOCKET(HANDLE,'CLOSE');
/* ============================================= */
IF CMD_UC /= "SHUTDOWNSHUTDOWN" THEN DO;
SIGNAL RECEIVE_DATA;    /* GO BACK               */
END;
/* ============================================= */
SAY "RECEIVED SHUTDOWN - SVASCII STOPPING";
EXIT;
RETURN;
TABLES:
C1="  !   # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? "
X1="405A407B5B6C50404D5D5C4E6B404B61F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F"
A1="20212223242526272829 2A2B2C2D2E2F303132333435363738393A3B3C3D3E3F"
C2='A B C D E F G H I J K L M N O P Q R S T U V W X Y Z '
X2='C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9'
A2='41424344454647484 94A4B4C4D4E4F505152535455565758595A'
C3='[ \ ] ^ _ a b c d e f g h i j k l m n o p q r s t u v w x y z '
X3='4040404040818283848586878889919293949596979899A2A3A4A5A6A7A8A9'
A3='5B5C5D5E5F6162636465666768696A6B6C6D6E6F707172737475767778797A'
RETURN;
/+
/*
/&
* $$ EOJ
```

Perl Client

```
#!/usr/bin/perl -w
#
=======================================================
# Linux Side Socket Client Sends a POWER Command To VSE
# Gets a Response Back Parse It then Display Results.
# It Will Be Up To The VSE Server Side To Do The EBCDIC
# to ASCII to EBCDIC Translation.
#
# S.Gracin 10/25/2001
#
=======================================================
$ipaddr='9.130.240.200';
$remote_port='3601';
$senddata='d lst';
# ==================================
print ("Starting Connection To ",$ipaddr, " Port= ",$remote_port,"\n");

use Socket;
socket(to_server, PF_INET, SOCK_STREAM, getprotobyname('tcp'));
# ==================================
# Convert Addr
# ==================================
$to_addr=inet_aton($ipaddr)
       or die "Can't convert $ippar: $!\n";
# ==================================
# Define address of Remote Machine
# ==================================
$portandaddr = sockaddr_in($remote_port, $to_addr);
# ==================================
# Connect
# ==================================
print("Sending Command --> ",$senddata,"\n");

connect(to_server, $portandaddr)
     or die "Can't connect to $remote_host:$remote_port  : $!\n";
# ====== Send Data
send(to_server, $senddata, $flags)
    or die "Error Trying To Send: $!\n";
print($readdata);

      $readdata = <to_server>;
# ==================================
# Get length of $readdata
$totaldatal=length($readdata);
# ==================================
# Find End Of Lines
$eoll=7;
```

```
$here=0;

$totaldatal=length($readdata);

while ($here < $totaldatal) {
  $dataloc=index($readdata,"<=EOL=>",$here);
  $rlength=($dataloc-$here);
    $currentr=substr($readdata,$here,$rlength);
       print($currentr,"\n");
  $here=($here+$rlength+7);
    };


# ==================================
# close connection
# ==================================
close(to_server);
```

zVM Server SOCKSVR

```
/* ============================================ */
/* THIS EXEC WILL EXPECT ASCII DATA             */
/* AS IT LISTENS ON PORT 3601                   */
/* THIS EXEC WILL XLATE ASCII TO EBCDIC         */
/* PROCESS COMMAND AND BACK TO ASCII AND SEND   */
/* ============================================ */
TRACE O;
PORTID="3601";
/* ============================================ */
/*  I N I T A L I Z E                           */
/* ============================================ */
/*  INITALIZE WILL RETURN:                      */
/*   RC - RETURN CODE                           */
/*   SUBTASK - MY NAME =  SERZVM1               */
/*   USE   SOCKET   NOT   SOCKET                */
/*                                              */
/*   PASSED: MY SUBTASK NAME ( SERZVM1 )        */
/* ============================================ */
CALL SOCKET 'INITIALIZE','SERZVM1';
/* ============================================ */
/*  S O C K E T                                 */
/* ============================================ */
/*  SOCKET WILL RETURN:                         */
/*      RC - RETURN CODE                        */
/*      ACTIVESOCKET - WHICH SOCKET NUMBER IS   */
/*                     ACTIVE                   */
/*                                              */
/*  PASSED: AF_INET - USE DOTTED IP ADDRRESSES  */
/*          SOCK_STREAM - STREAM                */
/*          IPPROTO_TCP - PROTOCOL IS TCP       */
/* ============================================ */
SOCKETDATA=SOCKET('SOCKET','AF_INET','SOCK_STREAM','IPPROTO_TCP')
;
PARSE VALUE SOCKETDATA WITH RC ACTIVESOCKET;
  IF RC /= O THEN DO;
   SAY 'RETURN CODE CAME BACK =' RC;
   SAY 'ON   SOCKET ';
   SAY 'EXITING PROGRAM';
CALL TERMSOCKET;
END;
/* ============================================ */
/* WILL WANT THIS FOR A FYI ON SCREEN           */
/* ============================================ */
HOSTADDRS=SOCKET('GETHOSTID');
HOSTADDR=WORD(HOSTADDRS,2);
```

```
HOSTADDRVIPA="18.18.18.18";
HOSTADDR=HOSTADDRVIPA;
/* ============================================ */
/*  B I N D                                     */
/* ============================================ */
/*  BIND WILL RETURN:   RC - RETURN CODE        */
/*                                              */
/*  PASSED: ACTIVESOCKET                        */
/*          DOMAIN - FOR AF_INET IT'S 2         */
/*          PORTID - PORT NUMBER I WILL USE     */
/*          HOSTADDRS - MY IP ADDRESS           */
/* ============================================ */
BINDDATA=SOCKET('BIND',ACTIVESOCKET,' 2 ' PORTID HOSTADDR);
SAY BINDDATA;
IF BINDDATA /= O THEN DO;
  SAY;
  SAY 'SOCKET BIND  FAILED .....';
  SAY;
  SAY 'EXITING PROGRAM';
  SIGNAL TERMALL;
  END;
/* ============================================ */
/* L I S T E N                                  */
/* ============================================ */
/* LISTEN WILL RETURN RC                        */
/*                                              */
/*   PASSED: SOCKET TO LISTEN ON - ACTIVESOCKET */
/* ============================================ */
LISTENDATA=SOCKET('LISTEN',ACTIVESOCKET, 10 );
  IF LISTENDATA /= O THEN DO;
  SAY;
  SAY ' SOCKET LISTEN FAILED .. ';
  SAY;
  SAY ' FUNCTION RETURNED ' LISTENDATA;
  SAY;
  SIGNAL TERMSOCKET;
  END;
/* ============================================ */
/* A C C E P T                                  */
/* ============================================ */
/* ACCEPT RETURNS: RC                           */
/*        NEW_SOCKET - SOCKET CLIENT IS HEARD ON */
/*        DOMAIN                                */
/*        PORT - PORT CLIENT IS USING           */
```

zVM Server SOCKSVR

```
/*        IPADDRESS - ADDRESS OF CLIENT          */
/*                                               */
/* WILL WAIT HERE FOR INBOUND DATA               */
/* ============================================= */
REACCEPT:
PASS='INITIAL';
SAY 'LISTEN AT IP ADDRESS --> 'HOSTADDR;
RC = 0;
ACCEPTDATA=SOCKET('ACCEPT',ACTIVESOCKET);
PARSE VALUE ACCEPTDATA WITH RC NEW_SOCKET DOMAIN PORT IPADDRESS;
/* ================ */
/* ================ */
"VMFCLEAR";
/* ================ */
/* ================ */
/* SAY  RC NEW_SOCKET DOMAIN PORT IPADDRESS;  */
 IF RC /= 0 THEN DO;
  SAY;
  SAY ' SOCKET ACCEPT FAILED .. ';
  SAY;
  SAY ' FUNCTION RETURNED ' ACCEPTDATA;
  SAY;
 SIGNAL TERMALL;
 END;
/* ============================================= */
/* R E C V                                       */
/* ============================================= */
/* RECV RETURNS: RC                              */
/*               READLENGTH - LENGTH OF DATA     */
/*               READDATA - DATA                 */
/* ============================================= */
REREAD:
RC = 0;
READDATA=SOCKET('RECV',NEW_SOCKET);
PARSE VALUE READDATA WITH RC READLENGTH READDATA;
 IF RC /= 0 THEN DO;
    IF PASS='INITIAL' THEN DO;
     SAY;
     SAY ' SOCKET READ FAILED .. ';
     SAY;
     SAY ' FUNCTION RETURNED ' READDATA;
      SAY;
      SIGNAL TERMALL;
      END;
```

```
 END;
 IF RC /= 0 THEN DO;
      SAY;
      SAY ' NO MORE DATA FROM.. 'IPADDRESS ' PORT =' PORT;
      SAY;
 END;
IF RC = 0 THEN PASS='NOTINITIAL';
/* ============================================= */
/* IF RC NOT EQUAL 0 THEN THE OTHER END CLOSED   */
/* THIS SOCKET CONNECTION SO CLOSE THIS END      */
/* AND GO BACK AND ACCEPT AGAIN.                 */
/* ============================================= */
IF RC /= 0 THEN DO;
    CLOSEDATA=SOCKET('CLOSE', NEW_SOCKET);
   END;
      IF RC /= 0 THEN SIGNAL REACCEPT;
/* ============================================= */
/* SEND COMMAND TO ZVM                           */
/* ============================================= */
 RINC=1;
 "PIPE VAR READDATA | XLATE A2E | VAR CMD";
 CMD = TRANSLATE(CMD);
 IF CMD = "SERVERDOWN" THEN DO;
 SAY " RECEIVED COMMAND TO SHUTDOWN SERVER";
 SIGNAL TERMALL;
 END;
 /* ============================================= */
 SAY "RECEIVED COMMAND " CMD;
  CMD"(STACK";
      IF RC /= 0 THEN DO;
      SAY "COMMAND " CMD " IN ERROR";
      RINC=2;
      REC.1="COMMAND " CMD " IN ERROR";
      END;
       IF RC = 0 THEN DO;
        DO UNTIL QUEUED() = 0;
        PULL REC.RINC
        RINC=RINC+1;
        END;
/* ============================================= */
 RCTR=RINC;
 RINC=1;
DO (RCTR-1);        /* SEND THEM ALL             */
 BUFFER=REC.RINC;
```

zVM Server SOCKSVR

```
 BUFFERE=INSERT(BUFFER," <=EOL=>");
"PIPE VAR BUFFERE | XLATE E2A | VAR BUFFER";
/* ============================================ */
/* SEND                                         */
/* ============================================ */
    SENDDATA=SOCKET('SEND',NEW_SOCKET,BUFFER);
    PARSE VALUE SENDDATA WITH RC SENDLENGTH SENDFILE;
     IF RC /= 0 THEN DO;
      SAY;
      SAY ' SOCKET SEND FAILED .. ';
      SAY;
      SAY ' FUNCTION RETURNED ' SENDDATA;
      SAY;
/* ============================================ */
/* CLOSE SOCKET IF A SEND FAILS WHILE WE STILL  */
/*       HAVE DATA.                             */
/* ============================================ */
      CLOSEIT=SOCKET('CLOSE',NEW_SOCKET);
      END;
   RINC=RINC+1;
 END;
 /* ============================================ */
 /* ALL DATA SENT - CLOSE SO CLIENT KNOWS WE ARE */
 /*                 FINISHED                     */
 /* ============================================ */
      CLOSEIT=SOCKET('CLOSE',NEW_SOCKET);
 SIGNAL REREAD;
 /* ============================================ */
      CLOSEIT=SOCKET('CLOSE',NEW_SOCKET);
      END;
   RINC=RINC+1;
 END;
/* ============================================ */
     CLOSEIT=SOCKET('CLOSE',NEW_SOCKET);
     END;
  RINC=RINC+1;
END;
/* ============================================ */
/* ALL DATA SENT - CLOSE SO CLIENT KNOWS WE ARE */
/*                 FINISHED                     */
/* ============================================ */
     CLOSEIT=SOCKET('CLOSE',NEW_SOCKET);
SIGNAL REREAD;
/* ============================================ */
```

```
/* WE'RE OUTTA HERE ..                          */
/* ============================================ */
TERMALL:
CLOSEIT=SOCKET('CLOSE',ACTIVESOCKET);
CLOSEIT=SOCKET('CLOSE',NEW_SOCKET);
TERM=SOCKET('TERMINATE','SERZVM1');
                          EXIT;
```

zVM Client SCLIENT

```
/* ============================================ */
/*  Client Program For VM Side Of Sockets       */
/*  Conversation - Sends EBCDIC Expects To      */
/*  Recieve EBCDIC                              */
/*  S. GRACIN 10/25/2002                        */
/* ============================================ */
trace o;
'vmfclear';
arg cmd;
  cmd_to_pass=cmd;
 if cmd = " " then do;
  cmd_to_pass='D LST';
 end;
connect_to="9.130.240.200"; /*  EBCDIC Machine   */
portid = 3600;              /* EBCDIC Port       */
/* ============================================ */
/*  I N I T A L I Z E                           */
/* ============================================ */
/*  INITALIZE WILL RETURN:                      */
/*   RC - Return Code                           */
/*   SUBTASK - My Name =  zvmclient             */
/*   SOCKETSALLOC - Number Of Sockets Preallocat */
/*   MACHINENAME - This Is Userid Of IP (TCPIP)  */
/*                                               */
/*   PASSED: My SUBTASK Name ( zvmclnt )         */
/* ============================================ */
socketinit=socket('initialize','zvmclnt');
rc = 0;
parse value socketinit with rc subtask socketalloc  machinename;
 IF RC /= 0 then do;
  say;
  say 'Return Code Returned = ' RC;
  say ' On   INITIALIZE     ';
  say 'No Allocation Made ... Exiting Program..';
  exit;
  end;
/* ============================================ */
hostaddrs=socket("gethostid");
defaulthostaddr=word(hostaddrs,2);
/* ============================================ */
/*  S O C K E T                                 */
/* ============================================ */
/*  SOCKET WILL RETURN:                         */
/*      RC - Return Code                        */
/*      ACTIVESOCKET - Which SOCKET Number Is   */
/*                     Active                   */
/*                                               */
/*  PASSED: AF_INET - Use Dotted Decimal address */
/*          SOCK_STREAM - Stream                 */
/*          IPPROTO_TCP - Protocol Is TCP        */
/* ============================================ */
rc = 0;
socketdata=socket('socket','af_inet','sock_stream','ipproto_tcp');
parse value socketdata with rc activesocket;
  if rc /= 0 then do;
   SAY 'Return Code =' RC;
   SAY 'On   Socket ';
   SAY 'Exiting Program';
  call termall;
```

zVM Client SCLIENT

```
end;
/* ============================================ */
/*  B I N D                                     */
/* ============================================ */
RC = 0;
socketbind=socket('bind',activesocket, 2 portid defaulthostaddr);
parse value socketbind with rc socketbinddata;
if rc /= 0 then do;
   SAY "Error On BIND --> "socketbinddata;
   signal termall;
end;
/* ============================================ */
/* F C N T L  -  Blocking Mode                  */
/* ============================================ */
socketfcntl=socket('fcntl',activesocket,'F_SETFL','blocking');
/* ============================================ */
/* C O N N E C T - Connect Active Socket        */
/* ============================================ */
socketconn=socket('connect',activesocket, af_inet portid connect_to);
parse value socketconn with rc conndata;
  if rc /= 0 then do;
  say 'Socket CONNECT Error';
  say;
  say 'Return Code = ' RC ' Connect To ' CONNECT_TO ' Failed';
  say
  signal termall;
  end;
/* ============================================ */
/* W R I T E - Command To VSE                   */
/* ============================================ */
senddata=socket('write',activesocket,cmd_to_pass);
```

```
/* ============================================ */
/*  R E A D - Data From VSE                     */
/* ============================================ */
do forever;
readdata=socket('read',activesocket);
parse value readdata with rc datalength recvdata;
  if datalength = 0 then signal termall;
  do until datalength = 0;
     loc=pos('<=EOL=>',recvdata);
     outrec=left(recvdata,(loc-1));
     say outrec;
     datalength=(datalength-(loc+6));
        if datalength = 0 then leave;
     recvdata=delstr(recvdata,1,(loc+6));
     datalength=length(recvdata);
  end;
end;
termall:
   closedata=socket('close', activesocket);
exit;
```

We Said..

- Many different Programming languages have a Sockets interface.
- VSE/ESA has two possible REXX Socket implementations.
- A program, process, can request and open a Socket into its IP stack.
- A Socket is given a PORT Number so it has an address different than all the other Sockets that might be Open to the stack.
- A Client is usually Write-Read-Close a Server is usually Read-Write-Read
- Some PORT numbers are “Well Known” so be sure what PORT numbers to use.

What We Did Not Discuss

Quite a bit …

- UDP vs. TCP
- OOB
- Flow Control
- Setting Socket Options
- RECEIVE vs. READ , SEND vs. SENDTO
- Design. You do not need two servers to do what we did it is just one way of providing the function.

And more, look through the referenced manuals and url's and have fun .

Reference

- VSE/REXX Reference V6R7 SC33-6642
- TCPIP For VSE Programming Guide
  -http://www.e-vse.com/
- zVM/REXX Reference SC24-6035
- REXX/VSE Enhancement: REXX Sockets API

  http://www-1.ibm.com/servers/eserver/zseries/os/vse/support/rexx/vserxsoc.htm
- The REXX LANGUAGE

  http://www-306.ibm.com/software/awdtools/rexx/language/
- O'Reilly books on Perl Programming

The End

RETURN TO INDEX