An Introduction to Extending VM/ESA with CP Exits

John Hall - Development

Safe Software, Inc.
333 Bahia Vista Drive
Indian Rocks Beach, FL 33785
(+1) 727-517-0718
E-Mail: JohnHall@SafeSoftware.Com
Web: WWW.SafeSoftware.Com

RETURN TO INDEX

Agenda

- Why would I want to extend VM/ESA?
- Getting started with CP Exits
- Useful techniques
- Lots of examples:
  - Hello World
  - Front-ending a CP Diagnose
  - Dynamically adding a CP exit point
  - Adding a new CP command
  - Front-ending a CP command

Example Programs

- Safe Software web site has full program materials
  - www.SafeSoftware.com - Click on “VM Downloads”

- Examples are for demonstration purposes.

Why Would I Extend VM?

- New Product: SafeAccess
  - SFS Migration facility
  - Uses CP Exits to intercept commands that interface with minidisks
- Areas of experience:
  - Adding and front-ending CP commands
  - Adding and front-ending CP Diagnoses
  - Utilizing existing & dynamic CP exit points
  - Writing *lots* of CP code

What is the CP Exit Facility?

- Allows you to dynamically add programs to CP
- Gives you predefined and dynamically added points of control in CP
- Allows you to add new commands, subcommands, or diagnoses
- Allows you to change the behavior of existing commands, subcommands, or diagnoses
- Reference: CP Exit Customization, CP Source code

What are Dynamic CP Exits?

- New in VM/ESA 2.4.0
- You can dynamically define exit points
  - Give it a name
  - A location
  - A set of input parameters
- Appears to be based on the TRSOURCE facility
  - You define input parameters with similar syntax

HELLO WORLD!

- Add new CP command that responds with “Hello, World” when invoked
- Syntax: HELLO
- Response: HCPJAH0950I “Hello, World”
- Example Demonstrates:
  - Adding a new command
  - Adding a program to CP
  - Updating *aaa*MDLAT MACRO
  - Using CP Exit commands
  - Using a Message Repository

HELLO WORLD!

- To add a new program to CP we need
  - Component ID - CP Uses this to identify programs related to your changes
  - *aaa*MDLAT MACRO - Used by assembler to determine linkage to/for your programs
  - *aaaaaa* ASSEMBLE - Program you want loaded
  - *aaaaaa* REPOS - Message repository
    - Optional, but recommended

HELLO WORLD!

- I’ve chosen to use “JAH” as my component ID
- This leads me to create
  - JAHMDLAT MACRO
  - JAHLIB MACLIB (containing JAHMDLAT MACRO)

HELLO WORLD!

- JAHMDLAT MACRO

```
 1            MACRO
 2 &LABEL     JAHMDLAT  &EPNAME
34            MDLATHDR  &EPNAME
35            MDLATENT  JAHELO,MODATTR=(PAG,MP,DYN),CPXLOAD=YES
36            MDLATENT  JAHELOEN,MODATTR=(PAG,MP,DYN),CPXLOAD=YES
42            MDLATTLR
43            MEXIT
44            MEND
```

- Place this into JAHLCL MACLIB
  - TXTLIB GEN JAHLIB JAHMDLAT

HELLO WORLD!

- JAHMSG REPOS
  - This file contains all of the messages issued by my CP Exits
  - File is associated via *applid* on GENMSG command and COMPID on HCPCONSL macro

```
 29 * 0950I Hello, World!
 31 09500101I Informational: "Hello, World!"
 32 09500201E Error: "Hello, World!"
```

- 09500101I - *mmmmffllla*
  - *mmmm* - Message number (HCPCONSL)
  - *ff* - Format (HCPCONSL)
  - *ll* - Line number (continuation)
  - *a* - severity
- GENMSG JAHMSG REPOS A JAH (MARGIN 63 CP
- If you find that you get message headers, but no text, you've forgotten to "ASSOC MESS COMP JAH EPNAME JAHMSG"
  (as, of course, I did)

HELLO WORLD!

- JAHELO ASSEMBLE
  - Program that simply writes back to the issuer “HELLO, World!”
- See JAHELO ASSEMBLE in your packet
- Prolog

```
1           HCPCMPID COMPID=JAH
2           COPY HCPOPTNS
3 JAHELO    HCPPROLG ATTR=(PAGEABLE,REENTERABLE),BASE=(R12)
4 JAHELO    TITLE 'JAHELO - CP Exits 101 - Hello World!'
```

- HCPCMPID=JAH - Identifies component ID
- HCPPROLG - Defines entry program, should match MDLAT

HELLO WORLD!

- CSECT

```
 47 JAHELO   CSECT
 48          HCPUSING PFXPG,R0
 49          HCPUSING SAVBK,R13
 50          EJECT 1
```

- Set global using, etc. here
- HCPENTER sets base register

```
 73 JAHELOEN HCPENTER CALL,SAVE=DYNAMIC
```

SAVE=DYNAMIC gives you a dynamic save area. Local program space is static

- See CP Exit Customization for entry conditions

HELLO WORLD!

- Entry Point - JAHELOEN

```
 72           SPACE 1
 73 JAHELOEN HCPENTER CALL,SAVE=DYNAMIC
 74           SPACE 1
 78           HCPCONSL WRITE,                Write this ever famous
 79                 DATATYPE=NONE,             line to issuer.
 80                 DATA='ABCXYZ123D Hello, World! (non-repos)'
 85           HCPCONSL WRITE,
 86                 COMPID='JAH',               Which component?
 87                 REPOSNUM='MS095002',       Msg 0950E
 88                 DATATYPE=NONE,            Skip terminal editing.
 89                 DESTINATION=TERMINAL       Send to terminal.
```

  - EMSG - editing
  - FULLEMSG - no editing
  - IMSG - Programmer's responsibility

HELLO WORLD!

- Exit

```
 74 EXIT     DS    0H
 75          MVC   SAVER2,=F'4'
 76          HCPEXIT EP=(JAHELOEN),SETCC=0
 77          HCPDROP R0
 78          HCPDROP R13
```

- This playfully always sets RC=4 (via R2)
  - The RC that the user sees is returned in R2
- CP convention is to set RC= msg #, except for -I messages.

HELLO WORLD!

- Assemble

```
GLOBAL MACLIB JAHLIB HCPGPI HCPPSI HCPOM1 HCPOM2
ASM JAHELO
```

- Load

```
cpaccess operator 191 c rr
17:37:22 CPACCESS request for mode C scheduled.
17:37:22 HCPZAC6732I CPACCESS request for OPERATOR's 0191 in
mode C completed.
Ready; T=0.01/0.01 17:37:22
cpxload jahelo text c nocontrol temp
17:37:35 Loaded as identifier 0
Ready; T=0.01/0.03 17:37:35
CPXLOAD JAHMSG TEXT C NOCONTROL TEMP
17:37:35 Loaded as identifier 1
Ready; T=0.01/0.03 17:37:36
cprelease c
17:37:40 CPRELEASE request for disk C scheduled.
17:37:40 HCPZAC6730I CPRELEASE request for disk C completed.
Ready; T=0.01/0.01 17:37:40
```

HELLO WORLD!

- Define, Enable, Run

```
define command hello anytime epname jaheloen privclassany
Ready; T=0.01/0.01 17:38:01
enable cmd hello
Ready; T=0.01/0.01 17:38:06
hello
17:38:08 ABCXYZ123D Hello, World! (non-repo)
17:38:08 JAHELO950E Error: "Hello, World!"
17:38:08 JAHELO950E Error: "Hello, World!"
17:38:08 JAHELO950E Error: "Hello, World!"
17:38:08 JAHELO950I Informational: "Hello, World!"
Ready(00004); T=0.01/0.01 17:38:08
```

- Unload

```
DISASSOC MESS COMP jah language ameng
Ready; T=0.01/0.01 17:40:19
cpxunload id 0
17:41:20 HCPCLP2770I CPXUNLOAD for load ID 0 has been scheduled
17:41:20 HCPCLH2771I CPXUNLOAD for load ID 0 has been completed
Ready; T=0.01/0.01 17:41:20
cpxunload id 1
17:41:20 HCPCLP2770I CPXUNLOAD for load ID 1 has been scheduled
17:41:20 HCPCLH2771I CPXUNLOAD for load ID 1 has been completed
Ready; T=0.01/0.01 17:41:20
```

Front-Ending a CP Diagnose

- Message OPERATOR when new directory activated
- Implement Diagnose 3C handler
- Example Demonstrates
  - Modifying/adding CP Diagnose
  - Modifying Diagnose behavior
  - Calling programs

Front-Ending a CP Diagnose

- Files used:
  - JAHMDLAT MACRO (continued)
  - JAHLIB MACLIB (containing JAHMDLAT MACRO)
  - JAHMSG REPOS (continued)
  - JAHMSG TEXT (from JAHMSG REPOS)
  - JAHD3C ASSEMBLE
  - JAHD3C TEXT (from JAHD3C ASSEMBLE)

Front-Ending a CP Diagnose

- New messages in JAHMSG REPOS
  - 0951I User "$1" issued Diagnose 3C
- JAHD3CUS entry point
  - Implements Diag 3C handler
  - Because the “real” handler doesn’t return success/failure, all we can do here is report on the use of Diag 3C
- JAHD3CUS entry point
  - Writes a message to OPERATOR console for each Diag 3C
  - Calls real Diag 3C handler (HCPUDSDS)
- Enable with “`CP MODIFY DIAG 3C EPNAME JAHD3CUS`”

```
 92 JAHD3CUS HCPENTER CALL,SAVE=DYNAMIC
 93          SPACE 1
 94 *
 95 * Call "real" Diag 3C handler.
 96 *
 97          HCPCALL HCPUDSDS
 98          SPACE 1
 99 *
100 * Generate message
101 *
102          MVC   D3CUID,VMDUSER    Set my userid as msg sub.
103          MVI   D3CDELIM,X'FF'    Set end of sub data mark.
104          SPACE 1
105          HCPCONSL WRITE,         0950I           *
106                COMPID='JAH',                     *
107                REPOSNUM='MS095101',              *
108                DATATYPE=NONE,                    *
109                DESTINATION='OPERATOR',           *
110                SUBDATA=D3CUID
111          SPACE 1
112 *
113 * Exit
114 *
115          HCPEXIT EP=(JAHD3CUS)
201 *-------------------------------------------------------*
202 * Redefine SAVEWRK for JAHD3C                           *
203 *-------------------------------------------------------*
204          ORG   SAVEWRK
205          SPACE 1
206 D3CUID   DS    CL8             Userid bring dir online.
207 D3CDELIM DS    X               Field seperator.
208          SPACE 1
209          DC    0S(L'SAVEWRK-(*-SAVEWRK)) Error if too big
```

Dynamic CP Exit

- Message OPERATOR when new directory activated
  - Put dynamic exit point inside HCPUDSDS in “success” path
- Example Demonstrates
  - Adding dynamic CP Exit point
  - Associating a dynamic exit point with an entry point

Dynamic CP Exit

- Files used:
  - JAHMDLAT MACRO (continued)
  - JAHLIB MACLIB (containing JAHMDLAT MACRO)
  - JAHMSG REPOS (continued)
  - JAHMSG TEXT (from JAHMSG REPOS)
  - JAHD3C ASSEMBLE
  - JAHD3C TEXT (from JAHD3C ASSEMBLE)

Dynamic CP Exit

- New messages in JAHMSG REPOS
  - 0951I New directory online by user "$1"
- JAHD3CEX entry point
  - Writes a message to OPERATOR console for each Diag 3C
- Define Dynamic Exit
  - `CP DEFINE EXIT 8000 AT HCPUDSSD + xxxx iiiiii PARM G0`
    - Passes us Guest CC (R0)
- Point dynamic exit to our new code
  - `CP ASSOCIATE EXIT 8000 EPNAME JAHD3CEX`
- Enable it
  - `CP ENABLE EXIT 8000`

Dynamic CP Exit

- Determining where to place Dynamic Exit Point
  - Identify HCPUDSDS as control point
  - Take care to consider “loss of control”
    - Holding Directory lock?
    - Pick spot after release of lock
  - Assemble HCPUDS
  - Identify offset and instruction
    - HCPUDS + 04C0 8900001C

```
- CP DEFINE EXIT 8000 AT HCPUDSSD + 4C0 8900001C PARM G0
```

```
*
* From HCPUDS ASSEMBLE
*
DSEXIT   DS    0H
         HCPCALL HCPGSVCC          Set guest condition code
         SPACE 1
         LA    R1,SYSDBDLK         Pass the lock address
         HCPCALL HCPLCKRX          Release SYSDBDLK exclusive
         SPACE 1
         SLL   R0,28               Return CC
         SPM   R0                   to caller
         SPACE 1
         L     R1,UDSADDR          Get address of UDSAVE
         HCPRELST BLOCK=(R1)       Release UDSAVE storage
         SPACE 2
         HCPEXIT EP=(HCPUDSDS)
```

We'll pick this instruction because it is after the unlock. We use the contents of R0 to determine "success"

```
158 JAHD3CEX HCPENTER CALL,SAVE=DYNAMIC
159          SPACE 1
160 *
161 * If HCPUDSDS's R0 = 0, then issue message.
162 *
163          L     R0,12(,R1)              Get contents of R0
164          CL    R0,PFX0                 R0=0?
165          BNE   CEX100                    No, just exit.
166          SPACE 1
167 *
168 * We know that we're in the "success" path, so we simply
169 * issue our message and return.
170 *
171          MVC   D3CUID,VMDUSER    Set my userid as msg sub.
172          MVI   D3CDELIM,X'FF'    Set end of sub data mark.
173          SPACE 1
174          HCPCONSL WRITE,              0950I *
175                COMPID='JAH',                *
176                REPOSNUM='MS095102',         *
177                DATATYPE=NONE,               *
178                DESTINATION='OPERATOR',      *
179                SUBDATA=D3CUID
180          SPACE 1
181 *
182 * Exit
183 *
184 CEX100   DS    0H
185          XC    SAVER15,SAVER15          RC=0
186          HCPEXIT EP=(JAHD3CUS)
```

Adding a CP Command

- Excercise
  - Maintain list of “FTP Servers” in CP
  - Report on all LINK commands from “FTP Servers”
- Solution
  - Implement FTPLIST command
  - “front-end” LINK command
- Example Demonstrates:
  - Adding CP command, class checking
  - Modify existing command behavior
  - Adding two programs to CP at same time
  - Calling added programs
  - 2 kinds of subroutines & save areas
  - Allocating/using storage
  - Serializing access to programs/information

Adding a CP Command

- Files used:
  - JAHMDLAT MACRO (continued)
  - JAHLIB MACLIB (containing JAHMDLAT MACRO)
  - JAHMSG REPOS (continued)
  - JAHMSG TEXT (from JAHMSG REPOS)
  - JAHLNK ASSEMBLE (Implements LINK front-end)
  - JAHLNK TEXT (from JAHLNK ASSEMBLE)
  - JAHFTP ASSEMBLE (Implements FTPLIST command)
  - JAHFTP TEXT (from JAHFTP ASSEMBLE)

Adding a CP Command

- Update JAHMDLAT with new programs
- New messages in JAHMSG REPOS
  - 970E User "userid" not in FTP Server list.
  - 0971E User "userid" is already in FTP Server list.
  - 0972I FTP Server list user: userid
  - 0973E FTP Server list is full.
  - 0974I Userid "userid" {added to|removed from} FTP Server list.
  - 0975E FTP Server list is empty.

Adding a CP Command

- Programs
  - JAHLNK ASSEMBLE
    - Implements front end to LINK command
    - Examines each LINK, if “FTP Server”, audits to OPERATOR console
    - Always calls “real” LINK command
  - JAHFTP ASSEMBLE
    - Implements FTPLIST command to maintain the “FTP Server” list
    - Syntax:

```
FTPLIST QRY
        ADD userid
        DEL userid
        AMI
```

Adding a CP Command

- JAHLNKEN
  - Determine if invoker is an “FTP server”

```
89 *-------------------------------------------------------------*
90 * See if invoker is FTPSERVE, skip to CP LINK command if not.*
91 *-------------------------------------------------------------*
92          LA    R1,VMDUSER              Address my id.
93          HCPCALL JAHFTPQU              Go check me out.
94          C     R15,=F'0'               RC=0 means FTP server.
95          BNE   LNK100                    No, skip to end.
96          SPACE 1
```

  - Use HCPCALL to call your program

Adding a CP Command

- JAHLNKEN
  - Allocate storage to save “LINKDATA”

```
103 *
104 * Address the GSDBK.  It contains the entire command string.
105 *
106          L     R8,VMDCFBUF              Address GSDBK
107          HCPUSING GSDBK,R8                and map it.
112          SLR   R1,R1                    Get the size of the
113          ICM   R1,B'0011',GSDDCNT         string in GSDBK.
114          SPACE 1
115          TM    GSDFLAG,GSDBUFWT         Console function to buffer?
116          BNO   LNK010                   No, skip adjustment.
117          S     R1,=F'8'                 Skip over buffer data.
130          HCPGETST LEN=(R1)              Allocate space.
131          LR    R6,R1                    Address storage
```

  - VMDBK points to GSDBK (VMDCFBUF)
  - Uses data in GSDBK to determine how big command string is.

Adding a CP Command

- JAHLNKEN
  - Invoke the “real” command (LINK)

```
226 * Restore registers to entry conditions
227 *
228          LM    R0,R9,SAVER0             Restore entry regs.
229          SPACE 1
230 *
231 * Call the real link.
232 *
233          HCPCALL HCPLNKIN               Call "real" LINK
234          ST    R2,SAVER2                Set RC
```

Adding a CP Command

- JAHLNKEN
  - Release storage

```
215           HCPRELST BLOCK=(R6)              Done with storage.
216           HCPDROP R6
```

  - Pass HCPRELST the address that HCPGETST gave you.
  - CP will abend if you pass a bad address or if you have walked beyond the end of the space that you requested on the HCPGETST.
  - Easy to use because you don't have to worry about the size.

Adding a CP Command

- JAHFTP ASSEMBLE
  - Entry Points
    - FTPLIST command via JAHFTPEN
    - Is a user an “FTP Server” via JAHFTPQU
  - Items of Interest
    - Scanning tokens
    - Two types of subroutine linkage
      - BAS/BR
      - HCPLCALL/HCPLEXIT
    - Component ID Block
      - Allocating
      - Serializing access using CMPBK

Adding a CP Command

- JAHFTP ASSEMBLE
  - Scanning tokens
    - VMDBK points to GSDBK with command string
    - HCPSCCFD scans using the GSDBK from VMDBK
      - Others (HCPS*aaaa*) for other requirements

```
109           HCPCALL HCPSCCFD      Pull off next token.
110           BNZ   PEN300            Missing, default to QRY.
```

  - On return
    - BNZ for nothing returned
    - R0 - Length of next token
    - R1 - Address of next token

Adding a CP Command

- JAHFTP ASSEMBLE
  - Subroutine Linkage - BAS/BR

```
135              BAS    R9,CHKCLASA              Got Class A?
136              BNZ    MSG026E                  Nope
713  CHKCLASA    DS     0H
729              LTR    R15,R15                   Set CC
730              BR     R9                          and return.
```

  - No registers are saved or restored
  - No additional storage allocated
  - Good for commonly used, but very small and non-destructive programs
  - Sug: Don't change anything except linkage, R0, R1, R15 registers

Adding a CP Command

- JAHFTP ASSEMBLE
  - Subroutine Linkage - HCPLCALL/HCPLEXIT

```
163           HCPLCALL FTPFND
584 FTPFND    HCPLENTR
645           ST    R8,SAVER1                Return A(CMPBK)
646           HCPLEXIT                       Return to caller
```

  - Storage allocated for save area (same as HCPCALL)
  - All registered saved/restored
    - Pass back values via SAVER*n*
  - Gets a “new” SAVBK
    - Be careful not to access caller’s SAVBK
  - Sug: Preferred

Adding a CP Command

- JAHFTP ASSEMBLE - Component ID Block
  - Provides you with a GLOBAL data area
    - Accessible across calls via name - “JAH”
    - Lock used to serialize access to points in your code
      - LOCKSHARED or LOCKEXCLUSIVE

```
 589            HCPXSERV LOCKEXCLUSIVE,
 590                  COMPID='JAH'
 606 FND010     DS     0H
 607            HCPXSERV ALLOCATE,      Allocate block with
 608                  COMPID='JAH',     extra space for JAHSECT
 609                  QWORDS=((JAHSECTL+15)/16)
 673            HCPXSERV UNLOCKEXCLUSIVE,
 674                  COMPID='JAH'
```

  - LOCKSHARED unless you are going to change something
  - LOCKEXCLUSIVE *always* waits for LOCKSHARED
  - Be aware that a name collision can occur (Jennifer A. Hall)

Adding a CP Command

- JAHFTP ASSEMBLE - SAVBK
  - Provides you with dynamic storage on entry
    - Save area that your caller's registers are stored in
    - Work area for your use: `SAVEWRK  DS    0XL40`
      - Woe to the dwarf that overruns this!!

```
782           ORG   SAVEWRK
783 USERID    DS    CL8              Remap SAVEWRK area.
784 ENDSUB1   DS    XL1              Indicates end of sub.
785 ACTION    DS    CL12            "added to" or "removed from"
786 ENDSUB2   DS    XL1              Indicates end of sub.
787           DC    0S(40-(*-USERID)) Assembler error if too big
```

  - Very useful when redefined, as above
  - 40 bytes, use them well
  - Sug: use an assembler construct (line 787) to tell you if you are using too much space

Adding a CP Command

- Load as one entity via CP Exit Directives
- CPEXIT LIST file:

```
OPTIONS TEMPORARY NOCONTROL
INCLUDE JAHLNK TEXT
INCLUDE JAHFTP TEXT
INCLUDE JAHMSG TEXT
```

- Command: **`CPXLOAD CPEXIT LIST C`**
- See CP201 EXEC
  - I ran this on OPERATOR on my 2nd level test system

Adding a CP Command

- Running it
  - Operator:

```
09:57:46 JAHLNK960I FTP TRAP: HALL01   <NONE>   LINK OPERATOR 191 1 RR
```

```
ftplist add hall
23:21:54 JAHFTP974I Userid "HALL" added to FTP Server list.
Ready; T=0.01/0.01 23:21:54
ftplist qry
23:22:01 JAHFTP972I FTP Server list user: HALL01
23:22:01 JAHFTP972I FTP Server list user: HALL
Ready; T=0.01/0.01 23:22:01
ftplist del hall01
23:22:04 JAHFTP974I Userid "HALL01" removed from FTP Server list.
Ready; T=0.01/0.01 23:22:04
ftplist qry
23:22:20 JAHFTP972I FTP Server list user: HALL
Ready; T=0.01/0.01 23:22:20
```

Final Thoughts

- Get a test system!!
- QUERY CPXLOAD - List of exits loaded
- QUERY CPCMD *cmdname* - Command info
- CP LOCK SYMBOL *name* - So you can trace it
- Stop by our booth if you'd like to "play" with more exits -- I should have my test system available
- Feel free to contact me…info on title page