Customizing CICS Using the System Programmer Interface

By Russ Evans

This article provides tips on using the CICS System Programmer Interface to customize transactions.

INTRODUCTION

While IBM-supplied exit facilities are extremely powerful, they have significant drawbacks. Unlike exits, the System Programmer Interface (SPI) provides a fully supported, upgradable and easily maintainable method of programmatically interacting with CICS at the system level.

Prior to CICS release 3, customers could inquire on and modify CICS system resources directly, either through the use of CICS macros, or by manually chaining documented control blocks. With the introduction of Object Code Only (OCO) and the redesign of CICS into its current Domain-based structure, both of these methods came to an abrupt end. IBM has attempted to fill in this gap by providing System Programmer extensions to the familiar 'EXEC CICS' API.

One of the primary advantages of the SPI is the familiar look and feel of the command syntax. Just like the standard API, all SPI commands start with EXEC CICS; they use HANDLE CONDITION and NOHANDLE in the same manner, and expect the same data area types. For convenience, the SPI commands are not included in the Application Programmers Reference Manual, but instead are documented in the System Programmers Reference Manual.

Using the SPI, the systems programmer can create programs that perform most of the functions provided by the IBM supplied transactions CEMT and CEDA; in some cases, more information is available to the SPI than can be gathered from the equivalent IBM transaction. This allows the systems programmer to create customized information screens, or user-friendly control transactions.

SYNTAX

Although the SPI commands look the same as API commands, the SPI make heavy use of CICS Value Data Areas (CVDAs) to hold state data about system resources. CVDAs are fullword areas that are set to a binary value that represents a specific state. For example, if the programmer inquires on the open status of a file, CICS might return a CVDA containing 18, which represents "OPEN". In the same manner, to close a file using the SPI, the programmer would pass CICS a CVDA containing 19, representing "CLOSED". The Systems Programmers Reference Manual lists all relevant CVDAs as part of the description of each command; a complete list appears in the index titled "CICS –value data areas used by all commands".

DEBUGGING AND MAINTAINING SPI PROGRAMS

The process of debugging and maintaining programs that use the SPI is much like that used for any CICS program. As with the API, SPI commands are caught by CEDF, and can be executed using CECI; any interactive debugger or CICS dump formatter that supports the API will also support the SPI. In addition, like the API, programs using the SPI may be written in either Assembler or COBOL.

One area that can cause problems is the introduction of new CICS releases: because the SPI is closely tied to the facilities being accessed, it

FIGURE 1: THE KERNEL OF THE NEWCOPY PROGRAM IS THE SET COMMAND

```
EXEC   CICS SET
       PROGRAM(MSGPROG)
       NEWCOPY VERSION(COPYVAL)
       RESP(WS_RESP)
```

The SET PROGRAM(MSGPROG) NEWCOPY simply instructs CICS to issue a newcopy on the program name contained in the variable MSGPROG (as with any other command, a literal can be substituted for the variable by enclosing it in quote marks.)
The parameter VERSION(COPYVAL) is an example of a CVDA. The VERSION keyword indicates that CICS should return a code indicating if this command actually resulted in a new copy. The possible return values are NEWCOPY or OLDCOPY. The result can be tested using the DFHVALUE keyword:

```
CLC    COPYVAL,DFHVALUE(NEWCOPY)      is it a true newcopy?
BE     SENDIT                         yes, send the message
```

Although the CVDA is a binary number, IBM provides a facility that allows the programmer to reference these values using a meaningful word. To use this facility, simply wrap the reserved word DFHVALUE() around the text. This DFHVALUE feature can be used in either Assembly or COBOL. For example, if inquiring on the status of a file, the command

```
IF WS-FILE-CVDA = DFHVALUE(OPEN)
```

will be translated to compare on the actual numeric value:

```
IF WS-FILE-CVDA = +18
```

IBM strongly recommends the use of DFHVALUE in place of hard-coding the specific numeric equivalent. In this manner, value changes can be handled with a simple re-assembly.

Note that the DFHVALUE facility works in exactly the same manner as the DFHRESP. If, through force of habit, a programmer codes

```
IF WS-FILE-CVDA = DFHRESP(OPEN)
```

The translator will produce an error message indicating that OPEN is not a valid value.

is possible that the specific CVDA values returned by one release of CICS will not be returned by a different release. While IBM often continues to support these obsolete values, it is the responsibility of the programmer to review the release guide for each release of CICS and make program modifications as required.

RESTRICTIONS

There are a few restrictions when using the SPI. First, the program must be compiled using the SP translator option. Second, SPI commands cannot be function shipped. The resource being affected must reside within the same region as the program issuing the command.

THE NEWCOPY PROGRAM

One common complaint regarding the CEMT facility is the inability to protect portions of its function (such as PERFORM SHUTDOWN) while providing access to other functions (such as newcopy). The SPI makes it easy to create and maintain programs that provide user-friendly access to specific CEMT facilities. The sample program REENEWC is such a program.

REENEWC is a bare-bones newcopy program. The user simply enters the name of the program to be newcopied, and receives a success/failure message. The programmer could easily add a write to CSSL or other file to create a log of newcopies. One advantage of the REENEWC program over CEMT NEWCOPY is that REENEWC can differentiate between an actual newcopy (where the program now occupies a new spot on the load library) vs. a program refresh. This capability is particularly useful in a testing environment, where a program can be accidentally linkeditted into a load library that comes after the production load library in the DFHRPL concatenation. See FIGURE 1 (on page 16).

Finally, note the use of the RESP option on the SET. RESP functions the same way for SPI commands as for API commands: control is returned to the program following successful or unsuccessful processing of the command. The programmer then interrogates the value returned in WS_RESP to determine if the command was successful.

CONCLUSION

The CICS System Programmer Interface combines powerful capabilities with ease of use and familiarity, allowing an experienced CICS programmer to create customized, site-specific transactions with little effort. Because this facility uses standard CICS Command Level interfaces, the risk of an abend or coding error causing a region wide outage is minimized. The programmer must exercise caution when modifying CICS resources, as the change will impact the entire CICS region.

FIGURE 2: THE WHOIS PROGRAM

IBM supplies transaction CEOT to supply information about the terminal the transaction has been entered on. Unfortunately, CEOT provides update as well as inquiry, so it cannot be made universally available. Once again, the SPI provides an easy way to create an inquiry only equivalent of CEOT; sample program REEWHO provides such a function. By making REEWHO available to all users, centralized help desk personnel can acquire specific terminal information from their callers.
The kernel of the WHOIS program is an Inquire command:

```
EXEC  CICS INQUIRE TERMINAL(EIBTRMID)
       NETNAME(LUNAME)
       OPERID(OPID)
       PRINTER(PRINTTO)
       SECURITY(TERMSEC)
       TERMMODEL(MOD)
       UCTRANST(UCTRAN)
       USERAREALEN(TCTUAL)
       USERID(USERID)
       RESP(WS_RESP)
```

The parameters on the INQUIRE TERMINAL command are a mixture of data areas and CVDAs. For example, TERMINAL(EIBTRMID) passes the name of our terminal to CICS, and NETNAME(LUNAME) requests that CICS return the actual network name of the terminal in data area LUNAME.SECURITY(TERMSEC) asks CICS to return a data value in area TERMSEC indicating the status of terminal preset security (either NOPRESETSEC or PRESETSEC.)

FIGURE 3: THE SYSTEM INQUIRY PROGRAM

The SPI also provides the programmer with the capability of customizing how information is displayed on an inquire. CEMT Inquire provides a wide range of information, but many systems programmers find that they typically look at only selected areas of several different CEMT panels. A simple SPI program can be written to gather all the "usual" data and present it on one screen.
Program REESYST performs this function using the INQUIRE SYSTEM command:

```
EXEC  CICS INQUIRE SYSTEM
       DSALIMIT(DSA)
       EDSALIMIT(EDSA)
       CDSASIZE(CDSA)
       ECDSASIZE(ECDSA)
       ERDSASIZE(ERDSA)
       RDSASIZE(RDSA)
       ESDSASIZE(ESDSA)
       SDSASIZE(SDSA)
       EUDSASIZE(EUDSA)
       UDSASIZE(UDSA)
       JOBNAME(JOBNAME)
       REENTPROTECT(RENT)
       STARTUP(START)
       STOREPROTECT(STORPROT)
       TRANISOLATE(TRANSISO)
       RESP(WS_RESP)
```

Like the INQUIRE TERMINAL command from the previous example, there are a mixture of data areas and CVDAs used in this command, and the options coded are only a subset of those available. By selecting these options, the inquire program is able to display detailed information about current DSA usage along side basic information about the CICS region, thus combining the CEMT I SYS panel with the CEMT I DSA panel.

Assembler source code for the sample programs mentioned in this article can be downloaded from www.reevans.com/download.html.

Russ Evans is the owner of R. E. Evans Consulting LLC, an international firm specializing in highly technical projects in the mainframe environment. He can be reached at russevans@reevans.com, or visit his website at www.reevans.com