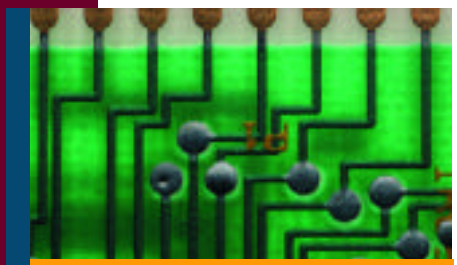

BY MICHAEL H. CARROLL

CICS/ESA V4R1: An Application Programmer's Perspective

CICS V4R1 offers many new enhancements including storage protection of transaction data through transaction isolation, several file control update restrictions, as well as numerous other facility and command changes to facilitate Year 2000 compliance.

I guess I'm getting old when I have to admit to myself that I have worked with all major revisions of CICS from V1R4 right through to V4R1! Many changes have taken place in those intervening years since the early '80s, both from a systems programming and applications programming perspective. A major revision of CICS took place between V2R3 and V3R1, which was perhaps the biggest upheaval to application programming since CICS first appeared. Macro-level programs ceased to exist and command-level for all tasks became the norm. Today, we've moved on again, from V3 to CICS V4R1. This release introduces some more enhancements to CICS. While CICS/ESA V4R1 has been around for awhile, many sites are only now progressing to this release after recovering from the trauma in recent years of moving to V3 from V2.

This article will explore some of the new or changed features of CICS V4R1 that may directly impact the application programming environment. Some of these changes are brought about by "system-type" enhancements while others are dictated by more mundane, but nevertheless necessary improvements such as Year 2000 compliance.

STORAGE PROTECTION AND TRANSACTION ISOLATION

Storage protection insulates CICS code and control blocks from applications, and transaction isolation protects tasks from each other. CICS/ESA 3.3 introduced the facility to exploit the ESA/390 subsystem storage protection facility in a way that enables you to prevent CICS code and control blocks from being accidentally overwritten by your application programs. However, it does not provide protection against deliberate overwriting of CICS code or control blocks. CICS cannot prevent an application from obtaining the necessary access (execution key) to modify CICS storage.

Transaction isolation in CICS V4R1 extends this storage protection to transaction data. Accidental overwrites of the transaction data by an application program of another transaction can affect the reliability and availability of your CICS system and the integrity of the data in the system. Thus, being able to protect against such accidents is an important enhancement.

The use of storage protection through transaction isolation is optional. Your systems programmer can choose whether to use storage protection facilities by means of CICS system initialization parameters, which are described in the *CICS/ESA System Definition Guide*. Some of the benefits of transaction isolation include:

- **Reduces system outages:** Transaction isolation prevents the data corruption and unplanned CICS system outages that normally occur when coding errors in user-key application programs cause the storage of user-key transactions to be accidentally overwritten. Prevention of accidental transaction data overwrites significantly improves the reliability and availability of CICS regions.
- **Protects application data from being overwritten by other errant applications.**
- **Protects CICS from being passed invalid addresses that might cause**

Figure 1: New Keyword for EXEC CICS READ UPDATE Command

TOKEN(data-value) specifies as a binary value a unique request identifier for a READ, used to control multiple UPDATE operations on a data set. This is an output value provided by file control to the requesting task.

data-value is an area defined as

```
                    COBOL    PIC S9(8) COMP
           PL/1     FIXED BIN(31)
           C/C++    int
```

Figure 2: Using EXEC CICS ASKTIME and EXEC CICS FORMATTIME

```
EXEC CICS ASKTIME ABSTIME(asktime)

The format of "asktime" is:

          COBOL    :          PIC S9(15) COMP-3
          C        :          char data_area[8];
          PL/I     :          FIXED DEC(15)
          ASM      :          PL8

EXEC CICS FORMATTIME ABSTIME(asktime)
     YYYYDDD(data-area)      | YYYYMMDD(data-area)     | YYYYDDMM(data-area)  |
     YYDDD(data-area)        | YYMMDD(data-area)       | YYDDMM(data-area)    |
     DDMMYYYY(data-area)     | DDMMYY(data-area)       |
     MMDDYYYY(data-area)     | MMDDYY(data-area)       |
     DATE(data-area)         | DATEFORM(data-area)     |
     DATESEP(data-area)      | DAYCOUNT(data-area)     |
     DAYOFWEEK(data-area)    | DAYOFMONTH(data-area)   |
     MONTHOFYEAR(data-area)  | YEAR(data-area)         |
     TIME(data-area)         | TIMESEP(data-value)     |

See CICS manuals for full details.
```

Figure 3: EXEC CICS ASSIGN ASRAKEY

ASRAKEY(cvda) returns the execution key at the time of the last ASRA, ASRB, AICA, or AEYD abend, if any. The CVDA values on the ASRAKEY option are as follows:

- CICSEXECKEY is returned if the task was executing in CICS-key at the time of the last ASRA, ASRB, AICA, or AEYD abend. Note that all programs execute in CICS key if CICS subsystem storage protection is not active.
- USEREXECKEY is returned if the task was executing in user-key at the time of the last ASRA, ASRB, AICA, or AEYD abend.
- NONCICS is returned if the execution key at the time of the last abend was not one of the CICS keys; that is, not key 8 or key 9.
- NOTAPPLIC is returned if there has not been an ASRA, ASRB, AICA, or AEYD abend.

storage violations: This essentially means CICS checks the ownership of addresses passed to it via EXEC CICS commands.

- **Aids application development:** This ensures possible future storage violations can be identified at testing time before a program moves into the production environment.

To turn on transaction isolation, the ISOLATE parameter in the transaction resource definition must be set to YES. This is only effective if storage protection is switched on for the entire region. CICS will abend a task that attempts to overwrite CICS storage or another application's storage. The abend code is AEYD. Check the *CICS/ESA Application Program-mers Guide* for more detailed information.

FILE CONTROL UPDATE RESTRICTIONS

From the early days of CICS, the inability to update multiple records simultaneously from the same file has been a serious limitation. Well, with CICS V4R1, this restriction is lifted. Whereas only one EXEC CICS READ UPDATE could be used previously, a new addition to the command allows multiple outstanding update reads against the same VSAM file. To achieve this, a new keyword has been added, namely TOKEN(data-value). The official definition of this new keyword is outlined in Figure 1.

The TOKEN keyword can be used on READ UPDATE, REWRITE, DELETE, and UNLOCK commands. READ for UPDATE can be used without TOKEN if only one such outstanding update is required. Therefore, existing applications coded with this in mind do not have to be amended. If TOKEN is used, then consistent reference to it should be made in all EXEC CICS commands associated with that READ for UPDATE.

EIBDATE

Most programmers are familiar with the EXEC Interface Block (EIB), which is basically a CICS structure that holds much useful information, including date and time fields. Officially, the method for obtaining the system date and time from CICS is the EXEC CICS ASKTIME command. However, many application developers discovered some time ago that the EIB fields hold the date and time (EIBDATE and EIBTIME respectively). The date is held as a packed length four-byte field and has the format X'00YYDDDF' where 'YYDDD'is the Julian date. At least in releases prior to V4R1 this was the case!

Many good folks, who are conscious of the Year 2000, have been amending code to accommodate the change from 1999 to 2000. To address these issues when using EIBDATE, it's not uncommon to see code such as this (in COBOL):

```
IF EIBDATE > 60000
    ADD 1900000 TO EIBDATE GIVING NEW-DATE
ELSE
    ADD 2000000 TO EIBDATE GIVING NEW-DATE.
```

Well, this assumes EIBDATE will remain unchanged. In fact, with CICS V4R1 this assumption is wrong! EIBDATE's new format under CICS V4R1 is X'0CYYDDDF', where 'C' is a century indicator, which, when 19 is added to it, gives the correct century. Therefore, 'C'will be zero in 1998 and 1999, one in 2000 through 2099,two in 2100 through 2199, etc. Strictly speaking, then, the previous lines of code should be rewritten as follows :

```
ADD 1900000 TO EIBDATE GIVING NEW-DATE.
```

As it happens, because EIBDATE will always exceed 60000 after midnight on

Figure 4: EXEC CICS ASSIGN ASRASPC

ASRASPC(cvda) returns the type of space in control at the time of the last ASRA, ASRB, AICA, or AEYD abend, if any. The CVDA values on the ASRASPC option are:

- SUBSPACE is returned if the task was executing in either its own subspace or the common subspace at the time of the last ASRA, ASRB, AICA, or AEYD abend.
- BASESPACE is returned if the task was executing in the base space at the time of the last ASRA, ASRB, AICA, or AEYD abend.

Note that all tasks execute in base space if transaction isolation is not active.

NOTAPPLIC is returned if there has not been an ASRA, ASRB, AICA, or AEYD abend.

Figure 5: EXEC CICS ASSIGN INVOKINGPROG

INVOKINGPROG(data-area) returns the eight-character name of the application program that used the LINK or XCTL command to link or transfer control to the current program.

If you issue the ASSIGN INVOKINGPROG command in a remote program that was invoked by a distributed program link (DPL) command, CICS returns the name of the program that issued the DPL command.

If you issue the ASSIGN INVOKINGPROG command in an application program at the highest level, CICS returns eight blanks.

Figure 6: EXEC CICS ASSIGN RETURNPROG

RETURNPROG(data-area) returns the eight-character name of the program to which control is to be returned when the current program has finished executing. The values returned depend on how the current program was given control, see the manual for full details.

December 31, 1999, the code will work without change. You might want to amend this, however, so that at least it makes sense to someone maintaining this code in years to come!

Of course, EIBDATE shouldn't really be used directly in this way at any rate. IBM supply an EXEC CICS FORMATTIME command that can return a date/time value in a variety of formats, including some new ones under CICS V4R1 that take the century into account. Figure 2 outlines this command and its companion EXEC CICS ASKTIME.

CEDF ENHANCEMENTS

CICS V4R1 introduces a new facility to Execution Diagnostic Facility (EDF) where the ability to invoke the CECI transaction within a CEDF session is now possible. Under CECI you can execute commands that are not present in the program to gain additional information or change the execution environment. This was a user-requested modification to CEDF.

OTHER EXEC CICS COMMAND CHANGES

Other than the file control commands discussed earlier, there have been some minor enhancements to other EXEC CICS commands also in CICS V4R1:

DEFAULT and ALTERNATE options added to:

- CONVERSE
- SEND
- SEND CONTROL
- SEND MAP
- SEND TEXT
- SEND TEXT NOEDIT

This allows an alternate screen size to be specified for these commands.

There are new options for ASSIGN:

- ASRAKEY (see Figure 3)
- ASRASPC (see Figure 4)
- ASRASTG (see manual)
- INVOKINGPROG (see Figure 5)
- RETURNPROG (see Figure 6)

There are many other minor changes listed in the documentation. See references at the end of this article for details on where to find all the relevant information.

IBM continues to build and enhance CICS. Indeed, many of the changes in V4R1 have been the result of requests and consultation with CICS developers and systems programmers. CICS is now a multi-platform system but the daddy of them all, CICS/ESA continues to transform itself as we march toward a new century.

OS/VS COBOL PROGRAMS: STORAGE PROTECTION

Migration of OS/VS COBOL programs from a pre-CICS/ESA V3R3 environment may cause possible storage protection exceptions. This will only happen if the storage protection facility, discussed earlier, is active. 0C4 abend may occur where restricted OS/VS COBOL language statements result in an MVS GETMAIN call. The abends will appear to be occurring in the COBOL library routines, not in the program itself. For example, the OS/VS COBOL verb "INSPECT" might be a possible candidate. Upgrading/recompiling the program to a later release of COBOL may be the only solution in this case.

COMPILERS SUPPORTED

CICS/ESA 4.1 supports the following assembler, COBOL, PL/1 and C/C++ compilers :

- MVS Assembler H, Version 2 (5668-962) and HLASM
- VS COBOL II (5668-958)
- IBM COBOL for MVS and VM V1R2 (5668-197)
- OS PL/1 Optimizing Compiler V1R5 (5734-PL1)
- OS PL/1 Optimizing Compiler V2R1 (5668-198) and later
- C (5668-040) and C library (5688-039) Version 1.2 C (5668-187) and C library (5688-188) Version 2.1
- C/C++ for MVS/ESA Version 3 compiler (5668-121) with LE/370 runtime libraries (5688-198)

CICS V4R1 also maintains execution-time support for application programs compiled by the following unsupported COBOL compilers:

- Full American National Standard COBOL V4 (5734-CB2)
- OS/VS COBOL (5740-CB1)

Execution-time support is no longer available for application programs compiled by the old OS/VS COBOL compilers 360-CB-545 and 5734-CB1.

SUMMARY

IBM continues to build and enhance CICS. Indeed, many of the changes in V4R1 have been the result of requests and consultation with CICS developers and systems programmers. CICS is now a multi-platform system but the daddy of them all, CICS/ESA (previously CICS/VS) continues to transform itself as we march toward a new century.

REFERENCES

CICS/ESA Application Programming Guide Version 4 Release 1, Document Number SC33-1169-00.

CICS/ESA Application Programming Reference Version 4 Release 1, Document Number SC33-1170-00. ts

NaSPA member Michael H. Carroll has 19 years of experience in the data processing industry. He's worked in both manufacturing and financial businesses, performing systems and applications programming for mainframe and client/server environments. Michael is a consultant currently working on Y2K projects for a large financial institution in Ireland.