BY RUSS EVANS

Using GLUEs and TRUEs in CICS

One of the reasons for the enduring popularity of CICS is its ability to be tailored to fit the needs of each shop. This article briefly describes two of CICS' more powerful customization facilities —Global User Exits (GLUEs) and Task Related User Exits (TRUEs) — and how you can use them to manage events with CICS.

INTRODUCED in the '60s, IBM's CICS has become the Teleprocessing Monitor of choice for companies around the world. While CICS' popularity is due in large part to its scalability and reliability, the amazing customizability of the CICS environment no doubt plays a significant role.

This article will not attempt to describe the standard customization options available with CICS, but rather will briefly describe two of the more powerful facilities: GLobal User Exits (GLUEs) and Task Related User Exits (TRUEs).

THE CICS CUSTOMIZATION GUIDE

The documentation for exits is maintained in the *CICS Customization Guide*. Since the exit points and the facilities available to the exit can change dramatically between CICS releases, you must have the customization guide for the release of CICS in which you intend your exit to run. You will find that it is worth the additional expense of acquiring a hard copy of this manual, as you will want to make notes in the text for future reference.

THE GLOBAL USER EXITS

The primary difference between Global and Task Related exits is the event that causes them to be executed. Global exit programs are executed when an event occurs within CICS (for example, when an abend occurs), and they run as part of CICS; Task Related exit programs are executed when a task requests it, and run as part of the task.

Once you have determined that an exit is required, the essential decision in determining whether to use a GLUE or a TRUE is this: If you are required to manage an event (such as file I/O) within CICS, use a GLUE. If the requirement is to manage all tasks, or a grouping of tasks, use a TRUE.

Because Global exits are event-driven, the customization guide groups them together according to the domain in which they run. This can be confusing, as the event points within a domain may not seem to be logically connected (for example, the program control exit points include XPCREQ, which is invoked prior to an EXEC CICS LINK, and XPCTA, invoked after an abend has been detected). For this reason, I always scan the Alphabetical List of Global Exit Points found at the beginning of the chapter on GLUEs. This list contains a one-line description of each exit, providing a cursory review to identify potential exit points for further research.

When deciding on which exit point to use, one of the critical issues you must confront is what CICS services will be available? The customization guide provides a detailed description of the restrictions for each exit point. For example, the description of the XPCTA exit point indicates that all XPI calls can be used, but no SPI or API calls can be used. However, the XDUCLSE documentation states that only the XPI call WAIT_MVS is valid. It is essential that the programmer be aware of all of the exit point's limitations when deciding among multiple possibilities.

Another critical point to keep in mind when designing a solution that requires a GLUE is the lack of upward compatibility provided by CICS. IBM states that neither source nor object compatibility is guaranteed between releases, and exit points can be changed or even removed. If your installation uses GLUEs, you must research each one as part of new release planning for each new CICS release you install.

THE XPI

CICS provides the exit Programming Interface (XPI), which is an alternative to the command level API for use at Global exit points. The XPI is a macro style language, with the macro name being dependent on the domain providing the service. As usual, all documentation pertaining to the XPI can be found in the customization guide, but the most critical points are:

- exit programs must be written in Assembler language
- exit programs must be written to 31-bit specifications (and return in 31-bit mode)
- exit programs must be fully reentrant (not just quasi-reentrant)
- prior to issuing an XPI call, the contents of UEPSTACK in the DFHEUPAR area must be loaded into register 13
- XPI calls use registers 0, 1, 14, and 15

Most importantly, because Global user exit programs execute as if they were part of the base CICS code, if the exit program abends, issues a wait, or violates any of the restrictions given in the customization guide, it can cause unpredictable results for the CICS region, potentially including region failure.

If the GLUE program will use XPI calls, it must include a DFHEUXIT TYPE=XPI-ENV macro call. This macro builds dsects and equates that are common to all XPI calls; in addition, each XPI macro has an accompanying copy member that you must include in your program's working storage area. The macros use a naming convention of DFHaabbX, with the related copybook being DFHaabbY, where aa is the two-character

domain identifier, and bb is a functional identifier. The macros use a standard coding style, as shown in Figure 1.

Looking at this macro invocation from the beginning, DFHSMMCX is the macro for storage control getmain/freemain. CALL indicates that this invocation should result in an actual getmain/freemain call to CICS (CICS allows the programmer to build the parm list incrementally by omitting the CALL option). CLEAR is used to clear out the DFHSMMCY parm list prior to building the parms for this CALL. IN and OUT are used to distinguish between input and output parms — for example, ADDRESS would be an output parm on a getmain, but an input parm on a freemain — and FUNCTION identifies which function of the macro is to be called.

Note the use of '*' in some of the parms; this is used to tell CICS that the target of the parm is to be the related field in the parm copybook. RESPONSE(*) on this sample command will put the return code into SMMC_RESPONSE, relieving the programmer from having to acquire additional storage for that field. Be careful to check the proper RESPONSE field, as checking SMSR_RESPONSE when the return code is in SMMC_RESPONSE could cause incorrect results. Because the macro requires access to the copybook fields, failure to include the copy member will result in assembly errors.

Some exit points allow the GLUE program to control processing of the event with return code settings. When coding a GLUE, the programmer must identify which return codes values are provided for use by that exit point, and explicitly use one of the valid codes on return:

```
RETURN (14,12),RC=UERCNORM
```

Always use the equated field to set the code, rather than the actual value. These values are not guaranteed to remain consistent from release to release. Note that the incorrect coding of the return code value can cause very unpredictable results.

Some exit points allow the use of a subset of the command level API. It is important to follow the limitations in the manual explicitly.

INFORMATION PROVIDED AT ENTRY

CICS provides some information to your exit program when it is invoked, via a parm area addressed in register 1. Certain information (such as the address and length of the global work area, or GWA) is passed to all exit points; additionally, each exit point receives data tailored to its interests. The general information is described in the customization guide under the DFHEUPAR dsect, but the data areas are generated using the DFHUEXIT macro. Because each exit point is provided unique information in its parm area, the DFHUEXIT macro provides the following parm list to tell it which exit points will be used:

```
DFHUEXIT TYPE=EP,ID=exit-point-id
```

THE GLOBAL WORK AREA

CICS provides for a common GWA to be used for all occurrences of a specific user exit program, or for a GWA shared between multiple programs. To acquire a GWA, include the desired length in the GALENGTH() option of the ENABLE PROGRAM command used to activate the exit. The address and length of the GWA are automatically supplied to the exit program; this information can be acquired by a command level program via the EXTRACT EXIT command.

The GWA can be used for any purpose, but is usually reserved for control information, counters, etc., that are required by the exit program at execution time.

FIGURE 1: AN EXAMPLE OF AN XPI MACRO TO ISSUE A CICS GETMAIN

```
DFHSMMCX CALL,
                CLEAR,
                IN,
                FUNCTION(GETMAIN),
                GET_LENGTH((Rn)),
                SUSPEND(NO),
                INITIAL_IMAGE(X'00'),
                STORAGE_CLASS(USER),
                OUT,
                ADDRESS((Rn)),
                RESPONSE(*),
                REASON(*)
```

CONTROLLING THE EXIT PROGRAM

You must activate the exit program at the desired exit point before CICS will use it. This can be performed using CECI during initial testing, but is more appropriately performed by an initialization program. The exit can be enabled at PLTPI time, although there are restrictions on actually running exit programs before startup has completed.

To activate the exit, use the START option of the ENABLE PROGRAM('PGM-NAME') EXIT('EXIT-POINT') GALENGTH(LEN) command. Note that the GALENGTH is optional; omitting it will result in no GWA being acquired. To terminate the exit program, use the DISABLE PROGRAM('pgm-name ') EXIT('exit-point')/ EXITALL command.

Use of the EXITALL option will disable the program for all exit points, whereas the use of the EXIT option will limit the disable to that exit point only. It is not necessary to terminate most exit programs before shutting down CICS.

TASK RELATED USER EXITS

While TRUEs were originally designed to support non-IBM file formats and Database Management Systems (DBMSes), they now have a variety of uses. Application programs can access the TRUE program directly via the DFHRMCAL macro (or through a high-level language CALL to a stub program that issues the DFHRMCAL). Alternatively, you can set up the TRUE to activate automatically at task start without a DFHRMCAL being issued. Regardless of how the TRUE has been called, the TRUE program itself can request that it be called again at task termination or syncpoint processing.

To request that the TRUE be called as each task is initialized, use the TASKSTART option of the ENABLE command that you use to enable the TRUE. If the TRUE must be activated again at task termination or syncpoint, the TRUE program must update the flags pointed to by the UEPFLAGS parm in DFHEUPAR. UEFMTASK is the task termination indicator, and UEFDSYNC is the syncpoint manager indicator.

Because these are bit flags, they must be ORd in rather than moved.

WRITING THE STUB PROGRAM

The "stub" program is a simple program written in Assembler language that handles the CALL from a high-level language to the TRUE program. The stub is responsible for issuing the DFHRMCAL macro that results in a call to the TRUE program. Note that the DFHRMCAL macro does not issue a BALR 14,15 (the standard call); the stub program is expected to keep the Register 14 value it received from the high-level language program call in place; the TRUE will return directly to the high-level language program after it has completed processing.

The syntax of the DFHRMCAL is quite simple:

```
DFHRMCAL TO=pgm-name
```

WORK AREAS FOR THE TRUE PROGRAM

When the TRUE program is called, CICS provides a full Command Level environment for it, including DFHEISTG. However, this environment is reset for each call to the TRUE from within the same task, so no information can be stored between invocations. To alleviate this restriction, use the TALENGTH option of the ENABLE command to create the equivalent of a TWA for the TRUE that will exist for the life of the task. UEPTAA provides the address of the TWA. The ENABLE command also provides for a GWA for the TRUE, which is created and accessed in the same manner as for Global User Exits.

RESTRICTIONS

While there are significantly more CICS facilities available to the TRUE program than for a GLUE, there are important restrictions. It is important to keep the following restrictions in mind when designing and coding your TRUE:

- You must use DFHEIRET to return if you have coded the DFHEIENT macro (or if it has been added by the translator).
- If your TRUE has been called at task termination, do not update any recoverable resources, as the CICS Syncpoint Manager has already run.
- Since all resources except task storage have been freed before a TRUE is called at task termination, resource level security may not function correctly.
- IBM states that you "must understand fully the circumstances in which the function shipping conversation may be terminated" if the TRUE attempts to access remote resources at task termination. This is misleading, and should simply state that remote resources cannot be used, as the sessions will not be freed after the TRUE completes.

CONCLUSION

Between the two types of user exits they provide a powerful facility for customizing CICS to meet the requirements of your installation. However, they come with a price. Before making the decision to use a GLUE or a TRUE to customize your CICS, you must carefully consider their drawbacks, including the potential to destabilize your regions.

Once you have determined that an exit is required, the essential decision in determining which to use a GLUE or a TRUE events vs. tasks is this: If you are required to manage an event (such as file I/O) within CICS, use a GLUE. If you are required to manage all tasks, or a grouping of tasks, use a TRUE.

Questions or comments? Please email editor@naspa.com.

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 Web site at www.reevans.com.