



CCAPRINT

A Newsletter Excerpt for Model 204 Users

January 2006

USE OF AND ACCESS TO PRODUCTS AND FEATURES ARE IN ACCORDANCE WITH THE TERMS AND CONDITIONS OF THE USER'S SOFTWARE LICENSE. THE PRESENTATION OF MATERIAL HEREIN DOES NOT, IN ANY MANNER, MODIFY SUCH TERMS AND CONDITIONS.

Increasing System Availability and Performance with REFRESH SUBSYSPROC

By James Damon

It is not uncommon to discover a need to change the source code of a subsystem procedure while the subsystem is active and hundreds, or even thousands, of users are logged in and actively using the subsystem. Unless a procedure group has been defined for the subsystem, all procedures in that subsystem are locked and unavailable for update. In that case you must stop the subsystem and either wait for all users to log out or bump all users of that subsystem. Also, when you stop a subsystem, all previously saved compilations for that subsystem are discarded, thereby increasing system overhead when the subsystem is subsequently restarted.

In releases prior to V6R1.0, you can make changes to procedures in an active subsystem by using procedure groups. In this case, you modify procedures and store them in any unlocked member of the procedure group. When next invoked, the modified procedures are located in an unlocked member, compiled, and evaluated. The original procedure remains in the locked member of the procedure group and is ignored. However, procedures found in unlocked members of a procedure group are always recompiled; the compilation is never saved. If a procedure is frequently included, the overhead involved in numerous recompilations can be significant.

Introducing the REFRESH SUBSYSPROC command

In V6R1.0 a significant enhancement to the process of refreshing procedures was implemented with the REFRESH SUBSYSPROC command. The syntax of the command is shown in Figure 1.

Figure 1. Syntax for refreshing procedures

```
REFRESH SUBSYSPROC procname  
    IN [GROUP | FILE] name  
    [FROM [{[PERM | TEMP] GROUP} | FILE] name2
```

The procedure *procname* is copied from the FROM clause--file or group--to the IN clause--file or group--replacing the existing copy of the procedure with the new copy. The copy affects only *locked* members of a procedure group. Several events occur as part of REFRESH processing:

1. The current, saved compilation for the procedure is deleted and the compiled pages, either in CCATEMP or in CCAAPSY, are released.
2. The APSY in-memory, procedure dictionary is updated to indicate that the new procedure has not been compiled.
3. If the original procedure had previously been made resident under the Resident Request feature, the storage for those resident request pages is released and the procedure is no longer resident.

When the next user invokes the procedure, under any subsystem, the new copy is compiled, saved, and evaluated for that user. If the Resident Request feature is active and the procedure is eligible, it is made resident--provided sufficient RESSIZE memory is available. Figure 2 is an example.

Figure 2. Refreshing a resident procedure

```
REFRESH SUBSYSPROC PROC2 IN TESTPROC FROM TESTZ
*** M204.2666: PROC2 REPLACED IN FILE TESTPROC
*** M204.2665: PROC2 REFRESHED IN SUBSYSTEM SUBSYS1
*** M204.1247: PROCEDURE PROC2 IN SUBSYS1 MADE RESIDENT
```

Note: M204.1247 appears in CCAAUDIT only when the procedure is made resident.

Command Conflicts

If the REFRESH SUBSYSPROC command is issued against a procedure that is currently being evaluated by any user in any subsystem, the messages shown in Figure 3 are issued.

Figure 3. Reporting conflicts with a REFRESH SUBSYSPROC command

```
REFRESH SUBSYSPROC PROC2 IN TESTPROC FROM TESTZ
*** 1 M204.2669: PROCEDURE PROC2 IS IN USE BY SUBSYSTEM SUBSYS1
*** M204.2683: REFRESH SUBSYSPROC COMMAND FAILED
```

The conflict occurs because you cannot refresh a procedure until all users have finished evaluation. If the message M204.2683 error occurs, you may reissue the command until it succeeds. It is possible that some procedures may remain active for long periods of time and may never be refreshable because they are always in use. This could occur, for example, in a procedure that has READ SCREEN statements. In these cases, having a procedure group defined to the subsystem provides the alternative method of making required changes to procedures in unlocked members of the procedure group and avoiding the M204.2669 message.

In Summary

Whenever the need arises to provide emergency fixes or any other kinds of modifications to procedures in active subsystems and you are looking for a solution that does not require continuous recompilation of the procedure in question, consider the REFRESH SUBSYSPROC command. The command is intended to improve both system availability and performance. System availability is improved by eliminating the need to bump users and stop subsystems. Performance is improved by avoiding repeated recompilations of modified procedures.

*© 2006 Computer Corporation of America
500 Old Connecticut Path, Framingham, MA 01701*