Extract Processing in the DFHCSDUP Utility

The demand for access to information in the DFHCSD has prompted the development of solutions that deliver a wealth of reporting features along with the ability to generate desired alterations to the DFHCSD file. In addition to several commercial products, IBM has introduced the EXTRACT command to allow easier access to the contents of the DFHCSD file.

BY KYLE KEENAN

THE migration of CICS resource definitions away from CICS table assemblies has been a long running process that began with the introduction of the resource definition online (RDO) feature in CICS version 1.6 more than a decade ago. While the introduction of RDO has greatly facilitated the ability to define and organize information about CICS resources, IBM has tried to discourage the use of non-IBM programs to read or modify the RDO file (DFHCSD) directly. Originally, the only method available to a user for doing any sort of resource definition cross referencing was to create a print listing of the entire contents of the DFHCSD file. The printed output could then be "eyeballed" for information or used as input for a process that could format the data into a structure that would allow for more complex queries to be entertained.

The DFHCSD file format is a key-sequenced VSAM dataset that houses definitions for all of the different resources, group names to which the resources belong, and lists of groups that can be specified for use in the CICS cold start. The record format of the DFHCSD file must not only accommodate the different record types within the DFHCSD file but must also allow for the addition of new resource types that may be developed or migrated from the old table format. Additionally, the ability to add new keywords within a resource is required. IBM accomplishes this by using a vector type record format that allows a record to self-define some of its own fields. While the record format of the DFHCSD file is very interesting to look at from the point of view of how well it accommodates the hierarchical structure of the RDO concept within one file structure, the idea of trying to become familiar enough with the file format to write a personalized interface program to query the DFHCSD file is a bit more work than most systems programmers would care to get involved in.

The demand for access to information in the DFHCSD has fueled the development of several commercial products that deliver a wealth of reporting features along with the ability to generate desired alterations to the DFHCSD file. CSD/Auditor for CICS, a Windows-based product from Emprise

Figure 1: IBM Sample Programs for Extract Processing

Program names	Languages	Description
DFH$CRFA DFH0CRFC DFH$CRFP	Assembler VS COBOL II PL/I	Produces a cross-reference listing of the resource definitions defined in the group or list you specify on the EXTRACT command.
DFH$FORA DFH0FORC DFH$FORP	Assembler VS COBOL II PL/I	Formats the group or list of resource definitions you specify on the EXTRACT command into a form suitable for the DB2 table load utility.
DFH0CBDC	VS COBOL II	Writes the list or group of resource definitions you specify on the EXTRACT command in the form of DEFINE commands suitable for use as a backup of copy the resources extracted.

Figure 2: Sample Execution JCL for EXTRACT Program

```
//JOBCARD JOB (TECHGRP),'CSDCHECK',CLASS=A,MSGCLASS=A,MSGLEVEL=(2,1)
//STEP1     EXEC PGM=DFHCSDUP,REGION=2M
//STEPLIB   DD DISP=SHR,DSN=cicsvs.system.sdfhload
//          DD DISP=SHR,DSN=userexit.program.loadlib
//DFHCSD    DD DISP=SHR,DSN=your.dfhcsd
//TEMPFILE  DD UNIT=SYSDA,SPACE=(CYL,(10,3),RLSE)
//SORTWORK  DD UNIT=SYSDA,SPACE=(CYL,(10,3),RLSE)
//SYSUT1    DD UNIT=SYSDA,SPACE=(1024,(100,100))
//RDOCARD   DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SYSIN     DD *
      EXTRACT LIST(PRODCICD) USERPROGRAM(CSDCHECK)  OBJECTS
```

Technologies (www.emprisetech.com), downloads the output of a LIST ALL OBJECTS command in the DFHCSDUP utility to create a standalone, PC-based database that is used to allow the viewer to select from a wealth of pre-defined reports. Another software product, CICS-RDO/EC, from SofTouch Systems, Inc. (www.softouch.com), also delivers the capability to create queries and searches for information in the DFHCSD file, and is designed to be executed as a CICS transaction, exploiting the use of the CEDA user interface.

If a CSD inquiry/management product is not available or cannot meet the requirements of a specific task or inquiry, the EXTRACT command in the DFHCSDUP utility in CICS (version 3 or later) offers the ability to deliver to a user program all of the keyword information of a group or a list of groups. IBM introduced the EXTRACT command to allow easier access to the contents of the DFHCSD file. The command delivers all of the particular details of a requested list or group to a user program and gives control to a user program at as many as nine different exit points when processing. This allows the user program to collect the necessary information to be directed into whatever output processing is specified by the user. The user program is limited to 24-bit addressability and also needs to be able to maintain addressability to its own program data areas across multiple calls from the DFHCSDUP utility. When link-editing the user program, a CICS-supplied stub, DFHEXTRA, must be included.

Three IBM sample programs are a valuable source of information on how to code a user exit program. Two of the programs are available in COBOL, assembler and PL/I, while the third is only available in COBOL II. One of the sample programs creates a cross-reference directory of objects and keywords defined in the CSD. Another sample program uses the EXTRACT output to create data for the DB2 table load utility. The third sample, DFH0CBDC (available as COBOL II only), generates DEFINE statements for all of the resources delivered to the user exit that are suitable for input into the DFHCSDUP utility for rebuilding the resource definitions, as shown in Figure 1. Prior to execution, the DFH0CBDC program must be compiled and then link-edited with an included CICS interface stub to the DFHCSDUP program.

Even as autoinstall for resources takes a larger role in the responsibility of CICS resource definition, RDO remains an important repository of information for CICS as well as a strategic feature of present and future CICS releases.

The cross-reference program and the DB2 formatting program are shipped in source code and executable format. *The CICS Customization Guide* details the necessary steps to run these programs. The COBOL-only sample, DFH0CBDC, comes in source code version only. The extra steps necessary to create an executable version of the DFH0CBDC program might prove useful to any business that needs to ship CICS resource definitions offsite in text format. This program may also be used as a backup utility for the DFHCSD file. While an IDCAMS REPRO would be a less time-consuming method for backing up the DFHCSD file, a backup created by the DFH0CBDC would consist of DEFINE statements that could be viewed and modified. If the contents of a DFHCSD file were to become compromised but the nature of the problem could not be detected through CICS startup messages or online CEDA use, the DFH0CBDC program offers some recourse by extracting only those aspects of the resource definition that are used in the DEFINE statement, disregarding all other information. Subsequent use of the DEFINE statements in rebuilding the DFHCSD will cause all information not explicitly in the DEFINE statements to revert back to a default setting. (I once used this method to repair a damaged DFHCSD file that could pass through a CEDA CHECK successfully but at CICS startup would install a number of transactions that caused a CICS region to abend.)

In order to make use of the EXTRACT user exit effectively, it might be worthwhile to create a modified version of one of the IBM sample programs that could be easily

Figure 3: Code to Attach the CICS EXTRACT Stub, DFHEXCI, to the User Exit Program

```
//JOBCARD   JOB  (TECHGRP),'COMPILE',CLASS=A,MSGCLASS=A,MSGLEVEL=(2,1)
//COB2     EXEC  PGM=IGYCRCTL,PARM='OBJECT',REGION=1024K
//STEPLIB    DD  DSNAME=SYS1.COB2.COB2COMP,DISP=SHR
//SYSPRINT   DD  SYSOUT=*
//SYSLIN     DD  DSNAME=&&WORK,UNIT=SYSDA,DISP=(NEW,PASS),
//               SPACE=(TRK,(3,3))
//SYSUT1     DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2     DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3     DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4     DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5     DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT6     DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT7     DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN      DD  DISP=SHR,DSNAME=your.source.lib(CSDCHECK)
//LKED     EXEC  PGM=IEWL,PARM='LIST,XREF,LET',
//               REGION=512K,COND=(7,LT,COB2)
//SYSLIB     DD  DSNAME=SYS1.COB2.COB2LIB,DISP=SHR
//CICSLIB    DD  DSNAME=your.cics.sdfhload,DISP=SHR
//OBJLIB     DD  DSNAME=&&WORK,DISP=(OLD,DELETE)
//SYSLMOD    DD  DSNAME=userexit.program.loadlib,DISP=SHR
//SYSUT1     DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT   DD  SYSOUT=*
//COBLIB     DD  DSN=SYS1.COB2.COB2LIB,DISP=SHR
//SYSLIN     DD  *
 ENTRY       DFHEXTRA
 CHANGE      EXITEP(CSDCHECK)
 INCLUDE     CICSLIB(DFHEXCI)
 INCLUDE     SYSLIB(ILBOSRV)
 INCLUDE     SYSLIB(ILBOCMM)
 INCLUDE     SYSLIB(ILBOBEG)
 INCLUDE     OBJLIB
 NAME        CSDCHECK(R)
//
```

altered when the need arises to make some type of inquiry or modification to the DFHCSD file that could not be addressed through CEDA processing. Once set up, the source code, compile JCL and execute JCL would then be available for quick modification and use. The remainder of this article examines a small COBOL II program, CSDCHECK, and the steps necessary to compile and execute this program via the EXTRACT utility command.

THE CSDCHECK PROGRAM

The CSDCHECK program is modeled after the DFH0CBDC program in that it uses a COBOL EVALUATE statement to determine why control has been passed to the user program and what actions to take. The intent of this program is to search a list for duplicate resource entries and create DELETE statements to eliminate all duplicate entries. (The duplicate entry that appears last in the list will not be deleted.) Although the program attempts to reproduce the type of information that might come from a CEDA CHECK LIST command, for the sake of simplicity, the sample program does not try to replicate the CHECK LIST output.

Control will be passed to the CSDCHECK program at the beginning and end of EXTRACT processing, at the beginning and end of processing a list, group, or object and for every keyword of all objects presented. A resource record is created for every object passed from the DFHCSDUP utility and the resource record file is sorted by resource name and returned to the CSDCHECK program. Whenever a duplicate resource is found, the preceding resource record is used to create a DELETE record. The program does not attempt to reconcile mapset and program definitions with the same name. Neither does it attempt to determine which definition of a file resource defined more than once is the currently installed definition.

The JCL necessary to execute the sample program is based on a JCL stream that would be used to execute the DFHCSDUP utility with the addition of whatever file I/O statements are called for in the user

Figure 4: CSDCHECK

```
CBL NOADV,APOST,SEQ,XREF,VBREF,DUMP,FDUMP,NORES,NORENT,FASTSRT
       IDENTIFICATION DIVISION.
       PROGRAM-ID. CSDCHECK.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT RESOURCE-FILE
                  ASSIGN TEMPFILE
                  ORGANIZATION IS SEQUENTIAL
                  ACCESS IS SEQUENTIAL.
           SELECT DATA-OUT
                  ASSIGN TO RDOCARD
                  ORGANIZATION IS SEQUENTIAL
                  ACCESS IS SEQUENTIAL.
           SELECT SORTFILE
                  ASSIGN TO SORTWORK.
       EJECT.
       DATA DIVISION.
       FILE SECTION.

       FD   RESOURCE-FILE
            RECORD CONTAINS 32 CHARACTERS
            BLOCK CONTAINS 0 RECORDS
            RECORDING MODE IS F
            LABEL RECORDS ARE OMITTED.
       01   RESOURCE-REC.
            02 RESOURCE-GRPNUM              PIC 9(4).
            02 RESOURCE-GRPNAME             PIC X(8).
            02 RESOURCE-OBJNAME             PIC X(8).
            02 RESOURCE-OBJTYPE             PIC X(12).

       FD   DATA-OUT
            RECORD CONTAINS 80 CHARACTERS
            BLOCK CONTAINS 0 RECORDS
            RECORDING MODE IS F
            LABEL RECORDS ARE OMITTED.
       01   OUT-REC                         PIC X(80).

       SD   SORTFILE
            RECORD CONTAINS 32 CHARACTERS.
          01 SORT-RECORD.
            02 SORT-GRPNUM                  PIC 9(4).
            02 SORT-GRPNAME                 PIC X(8).
            02 SORT-OBJECT.
            04 SORT-OBJNAME                 PIC X(8).
            04 SORT-OBJTYPE                 PIC X(12).
       *
        EJECT.
        WORKING-STORAGE SECTION.
       *
       * Call types, as defined by DFHCSDUP.
       *
        77  INITIAL-CALL                   PIC 99 VALUE IS 0.
        77  LIST-START-CALL                PIC 99 VALUE IS 2.
        77  GROUP-START-CALL               PIC 99 VALUE IS 4.
        77  OBJECT-START-CALL              PIC 99 VALUE IS 6.
        77  KEYWORD-CALL                   PIC 99 VALUE IS 8.
        77  OBJECT-END-CALL                PIC 99 VALUE IS 10.
        77  GROUP-END-CALL                 PIC 99 VALUE IS 12.
        77  LIST-END-CALL                  PIC 99 VALUE IS 14.
        77  FINAL-CALL                     PIC 99 VALUE IS 16.
       *
       * A switch to get out of the sort output procedure.
       *
        77  SORT-EOF                       PIC 99 VALUE IS 0.
       *
       * Counter used to number the order of the group in the list.
       *
        01  GROUP-COUNTER                  PIC 9(4).
       *
       * Used for comparison during sort output procedure.
       *
        01  WS-SAVE-GROUP                  PIC X(8).
        01  WS-SAVE-OBJECT.
        02 WS-SAVE-OBJNAME                 PIC X(8)   VALUE SPACE.
        02 WS-SAVE-OBJTYPE                 PIC X(12)  VALUE SPACE.
       *
       *  Record layout for the resource/sort record.
       *
        01  WS-SORT-REC.
```

Continued on next page.

Figure 4 continued from last page.

```
           02 WS-SORT-GRPNUM              PIC 9(4).
           02 WS-SORT-GRPNAME             PIC X(8).
           02 WS-SORT-OBJECT.
           04 WS-SORT-OBJNAME             PIC X(8).
           04 WS-SORT-OBJTYPE             PIC X(12).
      *
      *  Record layout for delete record.
      *

       01  WS-DELETE-LINE.
           02 FILLER                      PIC X(8)   VALUE ' DELETE '.
           02 WS-DELETE-OBJTYPE           PIC X(12).
           02 FILLER                      PIC X(2)   VALUE ' ('.
           02 WS-DELETE-OBJNAME           PIC X(8).
           02 FILLER                      PIC X(8)   VALUE ') GROUP('.
           02 WS-DELETE-GRPNAME           PIC X(8).
           02 FILLER                      PIC X(1)   VALUE ')'.
      *
      * Return code, if an invalid call is made to this program
       77  INVALID-CALL-TYPE              PIC S9(4) COMP VALUE IS 2.
      *
      * ************************************************************ *
       LINKAGE SECTION.
       01  EXIT-FUNCTION-CODE             PIC 99 COMP.
       01  EXIT-WORK-AREA-PTR             POINTER.
       01  EXIT-BACK-TRANS-CMD-PTR        POINTER.
       01  EXIT-LIST-NAME                 PIC X(8).
       01  EXIT-GROUP-NAME                PIC X(8).
       01  EXIT-OBJECT-TYPE               PIC X(12).
       01  EXIT-OBJECT-NAME               PIC X(8).
       01  EXIT-KEYWORD-TYPE              PIC X(12).
       01  EXIT-KEYWORD-LENGTH            PIC 999 COMP.
       01  EXIT-KEYWORD-VALUE.
       03 EXIT-KEYWORD-CHAR               PIC X OCCURS 1 TO 183
                                  DEPENDING ON EXIT-KEYWORD-LENGTH.
       EJECT.
      * ************************************************************ *
      *                                                              *
      *      M A I N L I N E   C O D E   S T A R T S   H E R E       *
      *      _____________________________________________           *
      *                                                              *
      * ************************************************************ *
       PROCEDURE DIVISION USING EXIT-FUNCTION-CODE
                                EXIT-WORK-AREA-PTR
                                EXIT-BACK-TRANS-CMD-PTR
                                EXIT-LIST-NAME
                                EXIT-GROUP-NAME
                                EXIT-OBJECT-TYPE
                                EXIT-OBJECT-NAME
                                EXIT-KEYWORD-TYPE
                                EXIT-KEYWORD-LENGTH
                                EXIT-KEYWORD-VALUE.
       MAIN-LOGIC.
      *
      * Perform appropriate action, according to EXIT-FUNCTION-CODE
      *
           EVALUATE EXIT-FUNCTION-CODE
      *
      * Do nothing on the initial call
      *
             WHEN INITIAL-CALL
               MOVE 0 TO RETURN-CODE
      *
      * Initialize the group counter and open the resource file
      *
             WHEN LIST-START-CALL
               MOVE 0 TO GROUP-COUNTER
               OPEN OUTPUT RESOURCE-FILE
      *
      * Set up current group name and group counter in resource record.
      *
             WHEN GROUP-START-CALL
               ADD 1 TO GROUP-COUNTER
               MOVE EXIT-GROUP-NAME TO WS-SAVE-GROUP
      *
      * Move object name and object type to resource record.
      *
             WHEN OBJECT-START-CALL
               MOVE EXIT-OBJECT-NAME TO RESOURCE-OBJNAME
               MOVE EXIT-OBJECT-TYPE TO RESOURCE-OBJTYPE
      *
      * The keyword exit is not used in this program.
      *
             WHEN KEYWORD-CALL
               MOVE 0 TO RETURN-CODE
      *
      * Write out a resource record.
      *
             WHEN OBJECT-END-CALL
               MOVE GROUP-COUNTER TO RESOURCE-GRPNUM
               MOVE WS-SAVE-GROUP TO RESOURCE-GRPNAME
               WRITE RESOURCE-REC
      *
      * Group end call not used.
      *
             WHEN GROUP-END-CALL
               MOVE 0 TO RETURN-CODE
      *
      * Close the resource file and initialize the save-object value.
      *
             WHEN LIST-END-CALL
               CLOSE RESOURCE-FILE
               OPEN OUTPUT DATA-OUT
               MOVE SPACE TO WS-SAVE-OBJECT
      *
      * Sort the resource file.
      *
             WHEN FINAL-CALL
               SORT SORTFILE
                 ON ASCENDING KEY SORT-OBJNAME
                                  SORT-OBJTYPE
                                  SORT-GRPNUM
                 USING RESOURCE-FILE
                 OUTPUT PROCEDURE FIND-DUPS
      *
      * Signal invalid call type TO DFHCULIS.
      *
             WHEN OTHER MOVE INVALID-CALL-TYPE TO RETURN-CODE
           END-EVALUATE
      *
      * Must use GOBACK so that COBOL data areas are preserved and usable
      * on the next invocation of the program.
      *
           IF EXIT-FUNCTION-CODE = FINAL-CALL THEN
               STOP RUN
           ELSE
               GOBACK.
      *
      *
       FIND-DUPS.
      *
      * Return a record from the sorted file.
      *
           RETURN SORTFILE RECORD
               AT END MOVE 1 TO SORT-EOF.
      *
      * When an object matches the saved object create a delete
      * card for the saved object.
      *
           IF SORT-EOF = 0
               IF WS-SAVE-OBJECT = SORT-OBJECT
                  MOVE WS-SAVE-OBJNAME TO WS-DELETE-OBJNAME
                  MOVE WS-SAVE-OBJTYPE TO WS-DELETE-OBJTYPE
                  MOVE WS-SAVE-GROUP   TO WS-DELETE-GRPNAME
                  WRITE OUT-REC FROM WS-DELETE-LINE.
      *
      * Save current object.
      *
           MOVE SORT-OBJECT  TO WS-SAVE-OBJECT.
           MOVE SORT-GRPNAME TO WS-SAVE-GROUP.
      *
           IF SORT-EOF = 0
               GO TO FIND-DUPS.
      *
       SORT-FINISHED.
      *
          EXIT.
      *
```

exit program. The JCL for CSDCHECK adds three //DD statements for a temporary resource record file, a sort work file, and RDOCARD, a sysout file for the output of the DEFINE statements, as shown in Figure 2. The RDOCARD file is set to sysout for ease of viewing but can easily be changed to become a permanent file. The compile and linkedit of the CSDCHECK program illustrated in Figure 3 shows the method for attaching the CICS EXTRACT stub, DFHEXCI, to the user exit program. The DFHEXCI stub is generated with an entry name of DFHEXTRA and a CSECT name of EXITEP. A CHANGE statement input into the linkedit step will alter the csect name to CSDCHECK. For COBOL programs, the CHANGE statement should point to the name specified in the program id statement of the user program.

Because of the simplicity of the output processing in CSDCHECK, as shown in Figure 4 (and also available for download from the ARTICLES LIB of the NaSCOM Internet site as filename NOV98001.EX1), some people might find it to be a more useful model for creating their own user exit program than the IBM sample programs. The IBM sample programs, however, should not be overlooked for information and ideas on creating a custom user exit for EXTRACT processing. Even as autoinstall for resources takes a larger role in the responsibility of CICS resource definition, RDO remains an important repository of information for CICS as well as a strategic feature of present and future CICS releases. ts

---

Kyle Keenan is a systems programmer at Centura Bank where he specializes in CICS and other IBM network software. He has been an avid (rabid) user of RDO since 1985. He can be reached at kkeenan@centura.com.