234

CICS update

May 2005

In this issue

A xephon PUBLICATION

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *CICS Update*, comprising twelve monthly issues, costs $270.00 in the USA and Canada; £175.00 in the UK; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the December 2000 issue, are available separately to subscribers for $24.00 (£16.00) each including postage.

CICS Update on-line

Code from *CICS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/cics; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *CICS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

List all installed global user exits – revisited

The original source for this program was published in *CICS Update,* issue 182, January 2001, and again in *Mainframe Week,* issue 48 (www.mainframeweek.com/journals/articles/0048/List+all+installed+global+user+exits). It was written by Claus Reis.

With the program CSDISGLU you can list all installed GLUEs (global user exits) in a CICS region and you don't need an OEM product to do it.

I have updated this source to use CICS control blocks and to determine whether the number of CICS exit points matches the number of table entries.

This works well in CICS TS 2.2 and I did some more substantial coding on this and made it CICS version-independent.

```
*ASM XOPTS(CICS,SP)
*********************************************************************
*ORIGINALLLY WRITTEN BY                                             *
*CLAUS REIS, CICS SYSTEMS PROGRAMMER,                               *
*            NUERNBERGER LEBENSVERSICHERUNG AG (GERMANY)            *
* DECEMBER 04,2002 ISSUE OF MAINFRAME WEEK - ISSUE 48               *
*                                                                   *
* COMPLETELY REDONE TO USE CICS CONTROL BLOCKS.                     *
* BY -                                                              *
*BRUCE BENTLEY, SENIOR SYSTEMS PROGRAMMER,                          *
*            FEDERATED MUTUAL INSURANCE MN (USA)                    *
*********************************************************************
*  PROGRAM:        FEDEXITS                                         *
*  TRANSACTION:    CCC2                                             *
*  FUNCTION:       THIS IS A SIMPLE PROGRAM TO DISPLAY THE EXITPGMS  *
*                  BY NAME FOR A SPECIFIC EXIT-POINT IN THE ORDER   *
*                  OF ENABLING = ORDER OF PROCESSING SEQUENCE.      *
*  LIMITATIONS:    CAN ONLY GO 3 DEEP ON ANY ONE EXIT POINT.        *
*                  CAN ONLY DISPLAY 16 EXIT POINTS WITHOUT PAGING.  *
*                  THE FOLLOWING EXIT POINTS ARE "HIDDEN" -
*                  EXIT100  DC   CL8'XINDT1'
*                  EXIT101  DC   CL8'XINDT2'
*                  EXIT102  DC   CL8'XLGWBC'
*                                                                   *
*  SUPPORTED CICS VERSION:                                          *
*  ANY                                                              *
```

```
*                                                                     *
*     THE TERMINAL OUTPUT LOOKS LIKE :                              *
*                                                                     *
*                    EXITPOINT TO EXITPROGRAM X-REFERENCE             *
*     ------------------------------------------------------------    *
*     CICSVER=Ø62Ø     SYSID=YYYY  TRANSID=ZZZZ  PROGRAM=FEDEXITS     *
*     ------------------------------------------------------------    *
*                PROCESSING SEQUENCE  ===>  ENABLING SEQUENCE         *
*     EXIT-POINT PROGRAM1 ENTRY1   PROGRAM2 ENTRY2   PROGRAM3 ENTRY3  *
*       NAME     PROGRAM  ENTRY                                       *
*       ....     .......  .......                                     *
*                                                                     *
*     NAME...       = EXIT POINT NAME                                 *
*     PROGRAM1-5    = PROGRAM NAME AT EXIT POINT                      *
*                     THE SEQUENCE GIVES THE ENABLE SEQUENCE, WHICH   *
*                     IS THE PROCESSING SEQUENCE TOO.                 *
*     ENTRY1-3      = ENTRYPOINT NAME IF ENABLED WITH ENTRYNAME        *
*                     IF NOT ENABLED WITH ENTRYNAME IT DEFAULTS TO    *
*                     THE PROGRAM NAME.                               *
***********************************************************************
*   EXPAND THE DFHEISTG FOR THE REQUIRED USER FIELDS
DFHEISTG DSECT
***********************************************************************
*    THE TERMINAL PRESENTATION OUTPUT AREA BEGINS HERE
***********************************************************************
OUT      DS    ØCL18ØØ
************************  HEADER LINE
HDRMSG   DS    XL79
NL       DS    XL1
FILLØ    DS    XL62
NLØ      DS    XL1
************************  ENVIRONMENT LINE
CICSVER  DS    CL8
CICS     DS    CL7
SYSIDENT DS    CL8
SYSID    DS    CL4
TRANID   DS    CL1Ø
TRANSID  DS    CL4
PROGNA   DS    CL1Ø
PROGRAM  DS    CL8
NL1      DS    XL1
************************  STARTER STARS
FILL1    DS    XL62
NL2      DS    XL1
************************  1ST LINE OF DATA OUTPUT
FILLER1  DS    CL14
EXPLAIN  DS    CL7Ø
NEWLINE  DS    XL1
************************  2ND LINE OF DATA OUTPUT
GLUE     DS    CL1Ø
```

```
FILL2    DS    CL4
PROGNAM1 DS    CL8
FILL3    DS    CL2
ENTRY1   DS    CL8
FILL4    DS    CL2
PROGNAM2 DS    CL8
FILL5    DS    CL2
ENTRY2   DS    CL8
FILL6    DS    CL2
PROGNAM3 DS    CL8
FILL7    DS    CL2
ENTRY3   DS    CL8
NL3      DS    XL1
LIST     DS    CL15ØØ
*************************************************************************
*    THE OUTPUT AREA ENDS HERE
*************************************************************************
         EJECT
         COPY DFHCSADS
         DFHUEXIT TYPE=DSECT
*************************************************************************
*     HERE BEGINS THE CSECT ITSELF                                      *
*************************************************************************
FEDEXITS CSECT
FEDEXITS AMODE 31
FEDEXITS RMODE ANY
         B     START
PROGNAME DC    CL8'FEDEXITS'        SET TO 8 CHARACTERS
         DC    CL8'&SYSDATE'        EYECATCHER
         DC    CL8'&SYSTIME'        INFORMATION
*
BEGIN    DS    ØH
*********************************************************************
*   CICS BASIC CODING                                               *
*   BASEREGISTER IS R3                                              *
*   CSA          IS R2                                              *
*********************************************************************
         USING DFHCSADS,2
         L     5,CSAOPFLA          CSA OPTIONAL FEATURES
         USING CSAOPFL,5
         L     5,CSAUETBA          ADDRESS OF USER EXIT TABLE
         USING DFHUETH,5
*********************************************************************
CICSLVL  DS    ØH
         EXEC  CICS INQUIRE SYSTEM RELEASE(CICS)
         L     4,UETHLEA           ADDESS OF LAST UETE.
         LA    4,DFHUETE__LEN(,4) POINT PAST LAST UETE.
         LR    8,4                 SAVE FOR COMPARING NUMBER OF ENTRIES
*********************************************************************
*        GO OVER THE UET HEADER TO THE 1ST UETE                      *
```

```
*        THERE IS 1 UETE FOR EACH EXIT POINT                    *
*****************************************************************
         LA    1Ø,UETHEND         POINT TO 1ST UETE
         USING DFHUETE,1Ø
         LA    5,UEXIDTAB         POINT TO EXIT ID TABLE
         LA    2,LIST             POINT TO OUTPUT AREA
*****************************************************************
SCANLOOP DS    ØH
         LA    7,UETEPL           POINT TO UEPL WITHIN UETE
         USING DFHEPL,7
         L     6,EPLEPBA          POINT TO UEPB WITHIN UEPL
         USING DFHEPB,6
         LTR   6,6                IS ANY PROG ENABLED AT THIS EXIT?
         BZ    NEXTUETE           NO, CHECK NEXT UETE
         MVI   Ø(2),X'15'         SET NL CHARACTER
         LA    2,X'1'(,2)         POINT TO NEXT BUFFER LOCATION
         MVC   Ø(8,2),Ø(5)        MOVE EXIT POINT NAME TO AREA
         LA    2,X'E'(,2)         POINT TO THE 1ST PROGNAME OUTFIELD
         MVC   Ø(8,2),EPBEMN      MOVE 1ST EXITPGM NAME TO OUTPUT AREA
         LA    2,X'A'(,2)         POINT TO THE 1ST ENTRYNAME OUTFIELD
         MVC   Ø(8,2),EPBEPN      MOVE 1ST ENTRY NAME TO OUTPUT AREA
         LA    2,X'A'(,2)         POINT TO THE NEXT PROGNAME OUTFIELD
*****************************************************************
* COUNT FOR 2 ADDITIONAL ENTRIES PER LINE, NO MORE POSSIBLE      *
*****************************************************************
         LA    9,X'2'
*****************************************************************
NEXTUEPL DS    ØH
         L     7,EPLNEPL          POINT TO NEXT UEPL WITHIN UETE
         USING DFHEPL,7
         LTR   7,7                IS THERE A NEXT PROGRAM ENABLED ?
         BZ    NEXTUETE           NO, CHECK NEXT UETE
         L     6,EPLEPBA          POINT TO UEPB WITHIN UEPL
         MVC   Ø(8,2),EPBEMN      MOVE 1ST EXITPGM NAME TO OUTPUT AREA
         LA    2,X'A'(,2)         POINT TO NEXT ENTRYNAME OUTFIELD
         MVC   Ø(8,2),EPBEPN      MOVE 1ST ENTRY NAME TO OUTPUT AREA
         LA    2,X'A'(,2)         POINT TO NEXT PROGNAME OUTFIELD
         BCT   9,NEXTUEPL         CHECK NEXT UEPL
*                                 MORE THAN 3 ENTRIES PER LINE ?
         L     7,EPLNEPL          POINT TO NEXT UEPL WITHIN UETE
         USING DFHEPL,7
         LTR   7,7                IS THERE A NEXT PROGRAM ENABLED ?
         BZ    NEXTUETE           NO, CHECK NEXT UETE
         MVC   Ø(5,2),=C'*MORE'   LET THEM KNOW THERE ARE MORE ENTRIES.
         LA    2,X'5'(,2)         POINT PAST THIS FIELD.
*****************************************************************
NEXTUETE DS    ØH
         LA    5,X'8'(,5)         POINT TO THE NEXT EXITTAB FIELD
         LA    1Ø,DFHUETE__LEN(,1Ø)      POINT TO NEXT UETE
         CR    1Ø,4               END OF UETE REACHED ?
```

```
         BE    SENDLIST           YES, SEND OUTPUT LIST
         B     SCANLOOP           CHECK NEXT UETE
*****************************************************************
SENDLIST DS    ØH
         MVI   17(2),X'15'       MOVE FINAL NL TO BUFFER
         EXEC CICS ASSIGN SYSID(SYSID) PROGRAM(PROGRAM)
         MVC   TRANSID,EIBTRNID
         EXEC CICS SEND TEXT FROM(OUT) ERASE FREEKB PAGING
*****************************************************************
RETURN   DS    ØH
         EXEC CICS RETURN
*****************************************************************
START    DS   ØH
*****************************************************************
*     LOAD DFHEISTG USER FIELDS                                 *
*****************************************************************
         MVC   CICSVER,=C'CICSVER='
         MVC   SYSIDENT,=C'  SYSID= '
         MVC   TRANID,=C'  TRANSID='
         MVC   PROGNA,=C'  PROGRAM='
         MVC   GLUE,=C'EXIT-POINT'
         MVC   PROGNAM1,=C'PROGRAM1'
         MVC   ENTRY1,=C'ENTRY1  '
         MVC   PROGNAM2,=C'PROGRAM2'
         MVC   ENTRY2,=C'ENTRY2  '
         MVC   PROGNAM3,=C'PROGRAM3'
         MVC   ENTRY3,=C'ENTRY3  '
         MVC   EXPLAIN,EXTEXT
         MVI   NL,X'15'
         MVI   NL1,X'15'
         MVI   NL2,X'15'
         MVI   NL3,X'15'
         MVI   NEWLINE,X'15'
         MVI   FILLØ,C'-'
         MVC   FILLØ+1,FILLØ
         MVI   NLØ,X'15'
         MVC   FILL1,FILLØ
         MVC   HDRMSG,MSGDAT
         B     BEGIN
*****************************************************************
*        CONSTANTS                                              *
*****************************************************************
MSGDAT   DC    CL79'CCC2          EXITPOINT TO EXITPROGRAM X-REFERENCE'
EXTEXT   DC    CL7Ø'PROCESSING SEQUENCE  ===>  ENABLING SEQUENCE ( MAX ,
               3 )'
*****************************************************************
         EJECT
         DFHUEXIT TYPE=EXITIDS
         LTORG
```

Bruce H Bentley
Senior Systems Programmer
Federated Mutual Insurance (Canada)

CICS VSAM Copy

On CICS 35th birthday, IBM unveiled a new tool, CICS VSAM Copy for z/OS V1.1 (CICS VC), which allows you to create copies of VSAM datasets while they are still under the control of CICS. With this tool, the point-in-time back-up copies of VSAM files can be created concurrently with business-critical online processing.

FEATURES OF CICS VC

The most important feature of CICS VC is its ability to produce a consistent point-in-time copy of a VSAM dataset while it is being updated via CICS transactions. The online CICS user can concurrently update the dataset being copied without even being aware of the fact that a copy is taking place. According to IBM, there is no discernible effect on the transaction response time in typical user situations.

The resulting copy does not contain any in-flight units of work and is similar to a copy taken while the update activity to the dataset is quiesced and identical to the one created by IDCAMS REPRO. Hence the copy created can be used immediately without requiring any further processing.

Though designed to produce online copies of CICS files, this utility can also produce offline copies, without interaction with CICS.

CICS VC can create consistent copies for all types of VSAM dataset – namely KSDS, ESDS, RRDS, and VRRDS.

In the case of a KSDS, copying a VSAM dataset with alternate index path(s) through the base cluster is fully supported. CICS VC also supports update activity through both the base and the paths while a copy through the base is in progress.

CICS VC can copy multiple VSAM datasets to the same 'point in time', which is useful when copying all the datasets belonging to the same application. All these datasets must be owned by the same CICS region, though a single copy request can consist of a combination of online and offline datasets.

CICS VC also provides features for making partial copies of VSAM datasets by specifying a RID or key range. This is useful for creating subsets of datasets, especially for testing purposes.

To simplify copy request processing, CICS VC keeps an internal registry of all the VSAM datasets owned by CICS regions that have CICS VSAM Copy started. This allows the user to copy or back up a dataset without knowing which CICS region owns the dataset.

In effect, CICS VC is a unique tool that copies the VSAM file with full data integrity and consistency, without having the disadvantage of application downtime for batch processing. Also, the copy process is more efficient than the IDCAMS REPRO.

MODES OF COPY REQUEST

CICS VC supports two modes of copy request:

1 Online mode – the dataset is open for update in a CICS region and can be updated during the copy process.

2 Offline mode – the copy utility attempts to allocate and open the dataset according to the parameters specified. The copy will be processed if the open is successful.

How the copy requests are processed depends on the mode of the request:

- In the case of online copy requests, the data is read from the dataset by subtasks running in the CICS region's address space, and is subsequently written to the output dataset by subtasks running in the copy utility's address space.
- In the case of offline copy requests, there is no interaction with CICS and the request is processed entirely in the copy utility's address space.

In addition to the above two modes, there is a dynamic mode that allows you to copy a dataset without knowing whether a CICS region currently has update access to it. In this mode, CICS VC first attempts to locate the dataset within a CICS region and satisfy the request in online mode. If a region that owns the dataset cannot be found, the request is processed in offline mode.

CICS VC ARCHITECTURAL OVERVIEW

Copy server task

The copy server is responsible for creating and maintaining control blocks and loading programs that are common to other CICS VC components. It typically runs as an MVS started task in its own MVS address space.

You can start up to 256 copy servers on a single MVS image. A single copy server can support multiple CICS regions and multiple simultaneous copy requests. A copy server can be either public or private, as determined when you start the server.

The copy request or a CICS region can be associated with a specific private copy server by specifying the CCSID parameter. If not specified, the public copy server is used by default and hence only one active public copy server is allowed.

CICS copy request server

The CICS copy request server coordinates CICS VC activity

for online dataset copy requests within the CICS address space. It also registers the CICS datasets eligible for online copy requests with its associated copy server.

The CICS copy request server's main task controls processing for a single copy request within the CICS region. It starts an instance of the CICS copy server subtask for each online dataset in the copy request. It also monitors on-going CICS transactions, noting when all in-flight units of work have completed to determine when the actual copy process can begin. It notifies the copy utility and the CICS copy request server subtasks when they can begin the copy process. On completion of the copy request, it performs the final clean up.

CICS copy request server subtask

There is one copy request server subtask for each dataset being copied by a copy request. It runs as a subtask within the CICS address space and exists for the time that it takes to process the dataset for which it was started.

It is responsible for sequentially reading the VSAM dataset and determining whether each record should be reflected on the output dataset, based on information provided by the frozen view engine (described later). Record images that should be included in the output dataset are placed in a buffer for processing by the copy utility.

Frozen view engine

The CICS VC frozen view engine, using various CICS exits and VSAM intercept points, keeps track of all update activity that is performed by CICS tasks after the copy request begins. If it detects an update and the copy has not yet processed the record, the updated information is saved such that the copy request server subtask can process the record correctly. Note that the updates made after the record has been processed are ignored.

Copy utility

The copy utility, which runs as an MVS batch job, initiates copy requests and oversees the resulting copy process.

For all online dataset requests, it drives the appropriate CICS copy request server to start the copy request in the appropriate CICS region. For online copy requests, it contains one copy subtask for each online dataset that receives data from the CICS copy request server subtask and writes it to the output dataset.

For each offline dataset copy request, it contains a subtask that reads the input dataset and writes the data to the appropriate output dataset.

The job step task also validates input parameters, locates the datasets to be copied, and dynamically allocates offline datasets.

CICS VC ONLINE COPY PROCESS

Online copy processing is more complex than an offline copy process because multiple operations must be coordinated between the address space of the CICS region that owns the dataset and the address space of the batch copy utility. Multiple components, given above, collaborate to achieve the consistent copying.

Initialization

A copy request is initiated through a standard MVS job step. After performing various initialization tasks it initiates a monitor task in the CICS region that owns the file. The monitor task runs as a standard CICS application task.

Phase 1 processing

At this point, the request enters what is referred to as ‘Synchronization Phase’ or ‘Phase 1’. The CICS copy request server task waits for all the CICS tasks that were active at the

start of Phase 1 to complete or issue a syncpoint.

During this wait time, the frozen view engine records the before-images of updates to the dataset being copied.

If Phase 1 does not end before a user-specified amount of time has elapsed, the CICS copy request server main task will fail the copy request and notify the copy utility that it should terminate immediately.

The long-running tasks or conversational transactions may extend this wait, hampering the copy process. To avoid this, CICS VC allows you to specify transactions that can be ignored during this process in the transaction exclude list (TXL).

When all tasks that were active in the system before Phase 1 have completed and issued a syncpoint to start a new Unit Of Work (UOW), or backed out, the copy request server task signals the copy utility and notifies all the copy server subtasks (start of Phase 2).

Phase 2 processing

During Phase 2, each CICS copy request server subtask reads sequentially through its associated dataset until it encounters the end of the dataset or reads a key that exceeds the supplied upper key range value.

The records read from the dataset by the copy request server subtask are merged with the before-image information and placed in a transfer buffer. As a result, the copy reflects only update activity that either happened before Phase 1 began or was part of a task or unit of work that was in progress when Phase 1 started. All activity from tasks started after the start of Phase 1 is excluded from the output dataset.

The copy utility subtask for each dataset pulls records from the transfer buffer and writes them to the associated output dataset.

Phase 2 of the copy request ends when all required records

for all datasets being copied have been written to the associated output datasets.

Clean-up processing

On completion of Phase 2, the copy request serversubtask(s) and main task are terminated and the copy utility generates a report and terminates.

RESTRICTIONS

If a consistent point-in-time copy is required for an offline copy, you must make sure the dataset cannot be updated while the offline copy is in progress.

CICS VC should not be used to copy VSAM datasets that are capable of being updated from more than one CICS region (eg share option 3/4 datasets or only exploiting record-level sharing). When a copy request is received for one of these datasets, a warning message is displayed on the system console, and the copy utility continues processing. Because CICS VC is unable to detect updates to the dataset from other address spaces/regions, it is up to the user to ensure that the dataset is being updated from only the CICS region in which CICS VC is processing. This restriction does not apply to datasets being updated in other CICS regions via CICS MRO/ISC function shipping, because all of these updates are actually shipped to the owning region.

In the case of online copy requests, the following restrictions also apply:

- The dataset must be a type supported by CICS.
- VSAM KSDS datasets with alternate index paths cannot be copied through a path component. An attempt to do this will be detected and the copy request will be terminated.
- VSAM ESDSs with alternate index paths are not supported, nor are they detected. It is the responsibility of the user to recognize this restriction and not attempt to copy a dataset in this scenario.

In the case of copying multiple datasets to the same point-in-time, the only restriction is that all the datasets must be owned by the same CICS region (except for datasets being copied in offline mode). If they are not, CICS VC will detect this and the copy request will be terminated.

CONCLUSION

Clearly CICS VC is quite a useful tool and fits with IBM's on-demand initiative. It allows the users to get the copy of VSAM files on demand without having to wait for the CICS application to be brought down. CICS VC is faster than IDCAMS and hence can deliver significant savings. This also provides enterprises with additional options for achieving high availability for CICS-VSAM applications.

Sasirekha Cota
System Software Group
Tata Consultancy Services (India)

CICS Web Services Assistant in CICS TS V3.1

CICS Transaction Server Version 3.1 ships two utility programs as part of the support for Web services in CICS. These utility programs are collectively known as the CICS Web Services Assistant. The objective of this article is to introduce the reader to these utility programs and to explain how they can be used to assist in implementing Web services in CICS.

BACKGROUND

The support for Web services in CICS TS V3.1 builds on the support that was shipped as the SOAP for CICS feature in Versions 2.2 and 2.3. This was originally shipped as a support pack and was later available as a free orderable feature with CICS TS V2.3.

Whilst the SOAP for CICS feature was a useful introduction to using SOAP in a CICS environment, it was still necessary for the end user of the feature to implement a lot of functions in order to Web service enable a CICS application. The writing of message adapters was required so as to handle SOAP headers and to convert the SOAP bodies into COMMAREAs and *vice versa*. With the introduction of CICS TS V3.1, support for Web services has been greatly enhanced. There are two main aspects to this support – a batch environment for preparing either WSDL (Web Services Description Language) or CICS application language structures for use in a Web service environment, and the CICS run-time changes that utilize some of the output from the batch time preparations. This article deals with the batch environment tooling support.

THE PROBLEM

Typically, the problem to be addressed takes one of two forms. Either it is desired to expose an existing CICS application as a Web service or it is desired to implement a Web service in CICS that is specified by a WSDL definition obtained from elsewhere. Starting from a CICS application and building a Web service is typically referred to as 'bottom up' development, while starting from WSDL and implementing it is referred to as 'top down' development. CICS TS V3.1 may be either a provider of service or a requester of service. Figure 1 shows how the various terms are used in this article.

Let us first consider the problems associated with bottom up development. Here, an existing CICS application is to be exposed as a Web service. This is probably the starting point for most people in this area. The difficulty faced in this scenario is that usually the only information to hand is the layout of the COMMAREAs, which the application program is expecting to receive and return. Also, the people who are familiar with the application may not be familiar with WSDL. In order to expose the application program as a Web service, it is necessary to produce a WSDL definition of the service. So

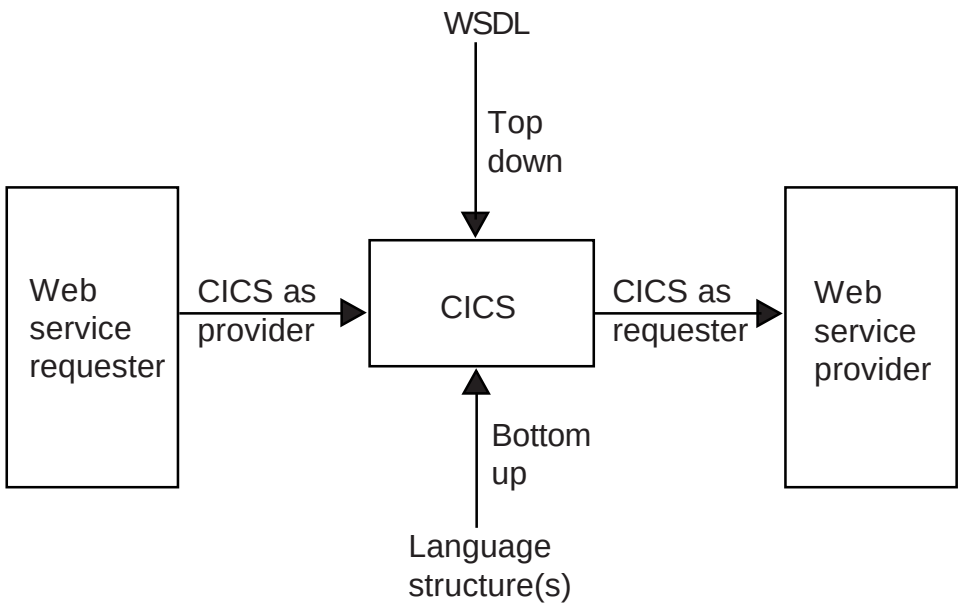

Figure 1: The terms used in this article

the problem is essentially one of converting a COMMAREA mapping into a WSDL document.

The problem in the top down development scenario is quite different. Here the typical starting point is a WSDL definition of a Web service. The requirement is to produce an application program that can implement the service defined by the WSDL. An important aspect of this is to map the data definitions, typically described in XML schema language, into a language structure suitable for use in an application program. The problem again is that the application programmer who must write the CICS application may not be familiar with XML schema language.

A SOLUTION

The CICS Web Services Assistant can be used to help to address both of the problem scenarios outlined above. The

utility programs that constitute the Web services assistant are DFHLS2WS and DFHWS2LS. The naming of these utilities may seem odd at first reading, but when it is explained that the jobs deal with converting language structures to WSDL, and WSDL to language structures, the names become more intuitive.

First, let us consider the situation where an existing COBOL application is to be exposed as a Web service. The DFHLS2WS batch job can handle COBOL, PL/I, C, and C++ structures but for this article COBOL will be used. Let us assume that we have an application concerning the number and types of pets owned by various people. The data structure mapping for the COMMAREA looks like the example shown below. The structure gives details of the name and age of the pet owner and of the pets owned by that person:

```
   10 NAME                   PIC X(16).
   10 AGE                    PIC S9(2) DISPLAY.
   10 PETS OCCURS 3.
        20 TYPE              PIC X(8).
        20 COLOUR            PIC X(8).
```

In order to produce WSDL for this structure, it must be placed where the batch job can locate it, for example in a PDS library MYHLQ.COPYLIB. Assume that it is called MYCOPYBK. If there is an output structure, this must also be made available to the batch job. The batch job must also specify parameters so that the WSDL can be produced properly. An example of the sort of things that need to be specified is shown below:

```
//LS2WS1   JOB (MYSYS,AUSER),MSGCLASS=H,
//             CLASS=A,NOTIFY=&SYSUID,REGION=0M
//*
//LS2WS     EXEC DFHLS2WS,
//INPUT.SYSUT1 DD *
LOGFILE=/u/webservices/wsbind/myservice1.log
PDSLIB=//MYHLQ.COPYLIB
REQMEM=MYCOPYBK
RESPMEM=MYCPYBK2
LANG=COBOL
PGMNAME=PETS
URI=the/url/to/access/service
PGMINT=COMMAREA
```

```
WSBIND=/u/webservices/wsbind/myservice1.wsbind
WSDL=/u/webservices/wsdl/myservice1.wsdl
*/
```

The LOGFILE parameter specifies where a log recording the progress of the batch job is to be written. The logfile is not used by anyone except CICS service. It should be retained since service will ask for this file should problems be found with the Web service. Some parameters are used to specify where the input and output structures are and some parameters are used in the generated WSDL. They provide information that is necessary for WSDL but which cannot be deduced from the COBOL copybook. When the job is run, various files are produced. The logfile, as mentioned above, might be needed by service. Also produced are the WSDL for the Web service and a file known as a WSBind file. The WSBind file is used by CICS in order to define the Web service for run-time support. The WSBind file should not be edited at all; there is no need to edit it and attempting to do so will probably result in its being unusable by CICS. If something is wrong with the file, then the batch job should be rerun in order to produce a new file. The WSDL produced conforms to both the WSDL 1.1 and the WS-I Basic Profile 1.0a specifications and looks as shown below, given the inputs indicated above. The generated WSDL is of a type known as wrapped document literal. Full details of the parameters for the batch utilities are documented in the CICS TS 3.1 infocenter.

```
<?xml version="1.0" ?>
<definitions  targetNamespace="http://www.PETS.MYCOPYBK.com"
     xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:reqns="http://www.PETS.MYCOPYBK.Request.com"
     xmlns:resns="http://www.PETS.MYCOPYBK.Response.com"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http:/
/www.PETS.MYCOPYBK.com">
    <types>
        <xsd:schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://www.PETS.MYCOPYBK.Request.com"
            xmlns:tns="http://www.PETS.MYCOPYBK.Request.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:complexType abstract="false" block="#all" final="#all"
                mixed="false" name="ProgramInterface">
```

```
                <xsd:annotation>
                    <xsd:documentation source="http://www.ibm.com/CICS/
Comment">This schema was generated for use with CICS.</
xsd:documentation>
                </xsd:annotation>
                <xsd:sequence>
                    <xsd:element name="name" nillable="false">
                        <xsd:simpleType>
                            <xsd:annotation>
                                <xsd:appinfo source="http://www.ibm.com/
CICS/Comment">#Thu Jan 20 14:23:04 GMT 2005
com.ibm.cics.wsdl.properties.synchronized=false
</xsd:appinfo>
                            </xsd:annotation>
                            <xsd:restriction base="xsd:string">
                                <xsd:maxLength value="16"/>
                                <xsd:whiteSpace value="preserve"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="age" nillable="false">
                        <xsd:simpleType>
                            <xsd:annotation>
                                <xsd:appinfo source="http://www.ibm.com/
CICS/Comment">#Thu Jan 20 14:23:04 GMT 2005
com.ibm.cics.wsdl.properties.synchronized=false
</xsd:appinfo>
                            </xsd:annotation>
                            <xsd:restriction base="xsd:short">
                                <xsd:maxInclusive value="99"/>
                                <xsd:minInclusive value="-99"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                    <xsd:element maxOccurs="3" minOccurs="3" name="pets"
nillable="false">
                        <xsd:complexType mixed="false">
                            <xsd:sequence>
                                <xsd:element name="type"
nillable="false">
                                    <xsd:simpleType>
                                    <xsd:annotation>
                                    <xsd:appinfo source="http://
www.ibm.com/CICS/Comment">#Thu Jan 20 14:23:04 GMT 2005
com.ibm.cics.wsdl.properties.synchronized=false
</xsd:appinfo>
                                    </xsd:annotation>
                                    <xsd:restriction base="xsd:string">
                                    <xsd:maxLength value="8"/>
                                    <xsd:whiteSpace value="preserve"/>
```

```
                                    </xsd:restriction>
                                    </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="colour"
nillable="false">
                                    <xsd:simpleType>
                                    <xsd:annotation>
                                    <xsd:appinfo source="http://
www.ibm.com/CICS/Comment">#Thu Jan 20 14:23:05 GMT 2005
com.ibm.cics.wsdl.properties.synchronized=false
</xsd:appinfo>
                                    </xsd:annotation>
                                    <xsd:restriction base="xsd:string">
                                    <xsd:maxLength value="8"/>
                                    <xsd:whiteSpace value="preserve"/>
                                    </xsd:restriction>
                                    </xsd:simpleType>
                                </xsd:element>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:element name="PETSOperation" nillable="false"
type="tns:ProgramInterface"/>
        </xsd:schema>
        <xsd:schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://www.PETS.MYCOPYBK.Response.com"
            xmlns:tns="http://www.PETS.MYCOPYBK.Response.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:complexType abstract="false" block="#all" final="#all"
                mixed="false" name="ProgramInterface">
                <xsd:annotation>
                    <xsd:documentation source="http://www.ibm.com/CICS/
Comment">This schema was generated for use with CICS.</
xsd:documentation>
                </xsd:annotation>
                <xsd:sequence/>
            </xsd:complexType>
            <xsd:element name="PETSOperationResponse" nillable="false"
type="tns:ProgramInterface"/>
        </xsd:schema>
    </types>
    <message name="PETSOperationResponse">
        <part element="resns:PETSOperationResponse" name="ResponsePart"/
>
    </message>
    <message name="PETSOperationRequest">
        <part element="reqns:PETSOperation" name="RequestPart"/>
```

```
    </message>
    <portType name="PETSPort">
        <operation name="PETSOperation">
            <input message="tns:PETSOperationRequest"
name="PETSOperationRequest"/>
            <output message="tns:PETSOperationResponse"
name="PETSOperationResponse"/>
        </operation>
    </portType>
    <binding name="PETSWMQSoapBinding" type="tns:PETSPort">
        <soap:binding style="document" transport="http://
schemas.xmlsoap.org/soap/http"/>
        <operation name="PETSOperation">
            <soap:operation soapAction="" style="document"/>
            <input name="PETSOperationRequest">
                <soap:body parts="RequestPart" use="literal"/>
            </input>
            <output name="PETSOperationResponse">
                <soap:body parts="ResponsePart" use="literal"/>
            </output>
        </operation>
    </binding>
    <binding name="PETSHTTPSoapBinding" type="tns:PETSPort">
        <soap:binding style="document" transport="http://
schemas.xmlsoap.org/soap/http"/>
        <operation name="PETSOperation">
            <soap:operation soapAction="" style="document"/>
            <input name="PETSOperationRequest">
                <soap:body parts="RequestPart" use="literal"/>
            </input>
            <output name="PETSOperationResponse">
                <soap:body parts="ResponsePart" use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="PETSService">
        <port binding="tns:PETSHTTPSoapBinding" name="PETSPort">
            <soap:address location="http://my-server:my-port/the/url/to/
access/service"/>
        </port>
    </service>
</definitions>
```

The only part of this WSDL that needs to be edited is the port binding, at the bottom. This is where the server and port number need to be specified. This will clearly depend on which CICS region and via which TCP/IP port the Web service is to be implemented. All the other element names in the WSDL are generated using names derived from the name of the program

as specified in the PGMNAME parameter.

The situation where a top down development is required can be simplified by the use of the DFHWS2LS batch utility. This is basically the reverse process to that described above. A supplied WSDL is to be used as a definition of the interface to a CICS application program. The WSDL to be input to the utility must be valid WSDL and may be document literal, wrapped document literal, or rpc literal WSDL. The utility program can take the WSDL, which maps the data by using an XML schema specification, and convert it to language structures in COBOL, PL/I, C, or C++ as required. The resulting language structures may then be used by application programmers in CICS application programs without their needing to read and understand the WSDL at all.

The batch utility programs can be run independently of CICS – that is, CICS does not need to be running on the system when the batch jobs are executed. It is required only that the WSBind file produced by the batch jobs be available to the CICS region on which the corresponding Web service is being implemented.

CONCLUSIONS

At first sight, the world of Web services may seem like a confusing array of specifications and abbreviations – XML, SOAP, WSDL, etc. However, by using the utility programs provided as the CICS Web Services Assistant, the need to know much about these specifications is minimized. The batch jobs can consume language structures from existing programs and generate valid WSDL, or they can consume valid WSDL and generate language structures that can be used in CICS application programs. A systems programmer could take responsibility for running the batch jobs, and the application programmers need be concerned only with the language structures.

Darren Beard (darren_beard@uk.ibm.com)
CICS Development
IBM(UK)

Using QMF services for printing in CICS

This article shows through a few examples how to print reports in CICS using QMF services, in order to satisfy end users' needs to print data retrieved from a database. Before enabling users to print reports, some planning for QMF under CICS and tailoring of QMF for CICS have to be done.

Since processing of a QMF transaction lasts longer than the average CICS transaction, a special CICS region dedicated to QMF transactions might be used, with the intention of routing QMF transactions from the current CICS region to it for processing. Also, because the maximum number of QMF users in a CICS region is restricted by the storage available below 16MB, for a large number of QMF users the need for multiple CICS regions may exist.

This article deals with printing by means of QMF rather than GDDM services, so the steps shown below are sufficient to tailor QMF for that purpose. All jobs mentioned below are located in the QMF sample library QMF.SDSQSAPE. Also dataset names, tablespace and table names, the node name, as well as the local CICS printer name given in the article are installation-dependent and should be changed according to internal standards.

1 Since QMF uses the CICS command-level application interface when running under CICS, jobs DSQ1ELNK and DSQ1EGLK must be tailored according to the internal standards, and should be submitted. Job DSQ1ELNK link-edits QMF with the CICS interface modules DFHIAI and DFHEAI0. The function of DSQ1EGLK is to translate, assemble, and link-edit the QMF-supplied governor with the same modules.

2 QMF trace dataset DSQDEBUG is allocated in job DSQ1BFRM.

3 DSQ1CDCS and DSQ1CDCT describe the QMF trace dataset to CICS.

4 Since the QMF panel dataset requires a VSAM CI size of 32KB, if the LSRPOOL in your installation is smaller than 32KB, specify an LSRPOOL that supports a VSAM CI size of 32KB through the DFHCSDUP utility.

5 Job DSQ1ECSD defines QMF programs, transaction (QMFE), and panel file to CICS using the DFHCSDUP offline utility.

6 In order to use QMF from CICS, update the CICS start-up procedure:

- Add to the STEPLIB concatenation the DB2 exit and load libraries SYS1.DSN710.SDSNEXIT and DSN710.SDSNLOAD.
- Place the load libraries containing the QMF and DB2 modules in the CICS module load library list DFHRPL (after CICS libraries):

```
DFHRPL DD ...
       QMF.SDSQLOAD
       SYS1.DSN710.SDSNEXIT
       DSN710.SDSNLOAD
```

- Add DD names for the QMF panel and trace datasets:

```
//*         QMF DATA SETS
//DSQPNLE   DD DSN=QMF710.DSQPNLE,DISP=SHR
//DSQDEBUG  DD DSN=QMF710.DSQDEBUG,DISP=SHR
```

If you choose to use exclusively QMF services to handle printing, two options exist: to define a transient data queue or to use a temporary storage queue through which reports are routed to the printer. In the case of printing to temporary storage, a program should be written to send the temporary storage queue to the printer. Temporary storage queues cannot handle more than 32,767 rows of data. So, if you need to print larger output or if you need to handle routing automatically (rather than writing a program to route output), you should use transient data queues.

CICS transient data queues are limited only by the amount of storage associated with the CICS Destination Control Table

(DCT). With transient data queues data could be printed to a dataset or SYSOUT class. Transient data queue may be defined as an intrapartition or extrapartition data queue. Some intrapartition data queues are also limited to 32,767 rows. Routing output to queues is accomplished by means of the QMF PRINT command, where you can specify both the name of the queue and the type of storage defined for that queue.

This article contains sample programs for using both queues. CICS JES programming interface commands are used to create a spool file that contains an appropriate report. Before you execute these samples, you should create and customize some QMF objects (ie the command synonym table, function key tables, QMF procedures). After customization of the example, all CICS users except SYSADM will use a temporary storage queue for printing, and only SYSADM will use a transient data queue.

The first step is to start QMF, by issuing CICS transaction QMFE, and generate a report from QMF.

In the case of printing to a temporary storage queue (QMFREPT in the program), when the Print function key is pressed, the PRT_QMF procedure is run, CICS transaction PR01 (program PRINTTS) is started, and the report will be printed without exiting QMF. Since any subsequent print command from the same terminal would use the same queue name, the program solves the problem of not appending the new report to the previous one.

A scenario for printing to the transient data queue (DPRI) follows. When the Print function key (F4) is pressed, the PRT_QMFTD procedure starts CICS transaction PR03 and program PRINTTD1 is run. PR03 then calls transaction PR02 (program PRINTTD), which thereafter starts transaction PR04 (program PRINTTD2). The user gets a CICS map, from where he can choose the print destination. The program is tailored for our environment, so when you enter '*' the spool file goes to class 'I', which belongs to the BETA 92 product. If you enter the name of a printer, the report will be printed on that printer.

If you enter '!', the report will be diverted to the other transient data queue (DPR1), which has a trigger defined to call another transaction (PR04) that performs printing, after 30 rows are written. In all cases, the program exits from QMF, and after printing the report, calls QMF again.

To print using temporary storage, the first step is to create a QMF procedure called in our case PRT_QMF. This sends the object to temporary storage, and then starts a transaction that prints the object:

```
PRINT REPORT (QUEUENAME=QMFREPT, QUEUETYPE=TS
CICS PRØ1 (FROM='QMFREPT')
```

Similarly, for printing using a transient data queue, QMF procedure PRT_QMFTD is used:

```
PRINT REPORT (QUEUENAME=DPRI, QUEUETYPE=TD
CICS PRØ3 (FROM='DPRI')
EXIT
```

In both cases QMF command synonyms are created for printing (PRTQMF and PRTQMFTD, respectively). Also the function key F4 on the report panel is customized to use the appropriate command synonym.

This example assumes that the user's profile for the CICS environment has the PFKEYS column value set to the appropriate name of the function key customization table: 'Q.MY_PFKEYS' for SYSADM, and 'Q.MY_PFKEYSTD' for all other users (creator SYSTEM).

So in this example table Q.CCOMMAND_SYNONYMS is created in tablespace DSQDBDEF.DSQTSDEF using table Q.COMMAND_SYNONYMS as a model, although it could be created in some other available tablespace:

```
CREATE TABLE Q.CCOMMAND_SYNONYMS
(VERB               CHAR(18)       NOT NULL,
 OBJECT             VARCHAR(31),
 SYNONYM_DEFINITION VARCHAR(254)   NOT NULL,
 REMARKS            VARCHAR(254))
 IN DSQDBDEF.DSQTSDEF;

CREATE UNIQUE INDEX Q.CCOMMAND_SYNONYMSX ON Q.CCOMMAND_SYNONYMS
```

```
(VERB ASC, OBJECT ASC)
USING STOGROUP DSQSGDEF
CLOSE NO;
```

After creating a command synonym table, synonyms are entered into the table by using SQL statements:

```
INSERT INTO Q.CCOMMAND_SYNONYMS (VERB, OBJECT, SYNONYM_DEFINITION)
VALUES ('PRTQMF', RUN Q.QMF_PRT, 'PRINT QMF REPORT')
INSERT INTO Q.CCOMMAND_SYNONYMS (VERB, OBJECT, SYNONYM_DEFINITION)
VALUES ('PRTQMFTD', RUN Q.QMF_PRTTD, 'PRINT QMF REPORT')
```

So when a user starts a QMF session, QMF loads a command synonym table whose name you specify in the synonyms field of the user's profile. Users who don't have unique profile rows will use the SYSTEM row. When a command is entered, QMF first checks the synonym table for a match and, if there is no match, QMF assumes the command is a base QMF command. When you enter the letters QMF in front of any command, QMF automatically assumes that the command is a base QMF command and does not check the synonym table for a match. To activate the command synonym table for users, it is necessary to update the SYNONYMS field of the user's profile with the proper command synonym table name:

```
UPDATE Q.PROFILES
SET SYNONYMS='Q.CCOMMAND_SYNONYMS'
WHERE CREATOR='SYSADM'
AND TRANSLATION='ENGLISH'
AND ENVIRONMENT='CICS';

UPDATE Q.PROFILES
SET SYNONYMS='Q.CCOMMAND_SYNONYMS'
WHERE CREATOR='SYSTEM'
AND TRANSLATION='ENGLISH'
AND ENVIRONMENT='CICS';
```

Then grant privileges so that assigned users can access the synonyms:

```
GRANT SELECT ON Q.CCOMMAND_SYNONYMS TO PUBLIC;
```

The default settings and labels for function keys on each QMF panel describe a common set of QMF tasks that end users are likely to perform. Because every site's needs are unique, however, QMF provides a way for you to customize both the

label that displays on the screen and the command that QMF executes when a user presses the key. Here, in order to simplify manipulation for end users when printing reports, two function key tables are created – Q.MY_PFKEYS and Q.MY_PFKEYSTD. They link customized function key definitions (F4) with the Report panel:

```
CREATE TABLE Q.MY_PFKEYS
   (PANEL                CHAR(18)       NOT NULL,
    ENTRY_TYPE           CHAR(1)        NOT NULL,
    NUMBER               SMALLINT       NOT NULL,
    PF_SETTING           VARCHAR(254),
    REMARKS              VARCHAR(254))
 IN DSQDBDEF.DSQTSDEF;
 COMMENT ON TABLE Q.MY_PFKEYS IS 'PF KEYS FOR CICS';
 CREATE UNIQUE INDEX Q.MY_PFKEYSX
           ON Q.MY_PFKEYS (PANEL, ENTRY_TYPE, NUMBER);

CREATE TABLE Q.MY_PFKEYSTD
   (PANEL                CHAR(18)       NOT NULL,
    ENTRY_TYPE           CHAR(1)        NOT NULL,
    NUMBER               SMALLINT       NOT NULL,
    PF_SETTING           VARCHAR(254),
    REMARKS              VARCHAR(254))
 IN DSQDBDEF.DSQTSDEF;
 COMMENT ON TABLE Q.MY_PFKEYSTD IS 'PF KEYS FOR CICS';
 CREATE UNIQUE INDEX Q.MY_PFKEYSTDX
           ON Q.MY_PFKEYSTD (PANEL, ENTRY_TYPE, NUMBER);
```

In order to customize commands (synonyms) executed when a user presses F4 on the Report panel, the following SQL statements are used:

```
INSERT INTO MY_PFKEYS (PANEL, ENTRY_TYPE, NUMBER, F_SETTING)
 VALUES('REPORT', 'K', 4, 'PRTQMF');

 INSERT INTO MY_PFKEYSTD (PANEL, ENTRY_TYPE, NUMBER, F_SETTING)
 VALUES('REPORT', 'K', 4, 'PRTQMFTD');


 GRANT SELECT ON Q.MY_PFKEYS TO PUBLIC;
```

For labels, the default 4=Print is kept.

In order to activate new function key definitions and enable users to use them, the PFKEYS field of the user's profile should be updated with the name of the proper function key

definitions table. Always include a value for the TRANSLATION and ENVIRONMENT columns in a query that updates the Q.PROFILES table:

```
UPDATE Q.PROFILES
 SET PFKEYS='Q.MY_PFKEYS'
 WHERE CREATOR='SYSADM'
 AND TRANSLATION='ENGLISH'
 AND ENVIRONMENT='CICS';

UPDATE Q.PROFILES
 SET PFKEYS='Q.MY_PFKEYSTD'
 WHERE CREATOR='SYSTEM'
 AND TRANSLATION='ENGLISH'
 AND ENVIRONMENT='CICS';
```

Program and map compilation as well as linking are done using standard procedures, so the corresponding jobs are not attached.

LPRINT

```
         PRINT ON,NOGEN
LPRINT   DFHMSD TYPE=MAP,MODE=INOUT,STORAGE=AUTO,SUFFIX=,LANG=PLI
         TITLE 'Local Printer Selection         '
LPRINT1  DFHMDI SIZE=(24,80),CTRL=(FREEKB,ALARM),MAPATTS=(SOSI),       *
               DSATTS=(SOSI),COLUMN=1,LINE=1,DATA=FIELD,               *
               TIOAPFX=YES,OBFMT=NO
         DFHMDF POS=(5,31),LENGTH=20,INITIAL='QMF Report Printing ',   *
               ATTRB=(PROT,NORM)
         DFHMDF POS=(7,31),LENGTH=9,INITIAL='Printer :',               *
               ATTRB=(PROT,NORM,ASKIP)
PRT      DFHMDF POS=(7,42),LENGTH=4,ATTRB=(NORM,                       *
               FSET,IC),INITIAL='*'
         DFHMDF POS=(7,47),LENGTH=8,                                   *
               ATTRB=(PROT,NORM,ASKIP)
         DFHMDF POS=(9,20),LENGTH=50,INITIAL='Instead of * you can ente*
               r JES printer name or  ',                               *
               ATTRB=(PROT,NORM,ASKIP)
         DFHMDF POS=(10,20),LENGTH=50,INITIAL='          ! for local CI*
               CS printer P001  ',                                     *
               ATTRB=(PROT,NORM,ASKIP)
         DFHMDF POS=(11,32),LENGTH=20,INITIAL='(trigger on 30 rows)',  *
               ATTRB=(PROT,NORM,ASKIP)
MES      DFHMDF POS=(19,22),LENGTH=43,ATTRB=(NORM,                     *
               PROT,ASKIP)
         DFHMDF POS=(23,31),LENGTH=20,INITIAL='PF3 - return to QMF ',  *
               ATTRB=(PROT,NORM,ASKIP)
```

```
         DFHMSD TYPE=FINAL
         END
```

PRINTTS

```
PRINTTS: PROC OPTIONS(MAIN);
/* ************************************************ */
/*    PROGRAM IS CALLED FROM QMF (SYNONYM: PRTQMF)    */
/*    TRANS: PRØ1                                     */
/* ************************************************ */
DCL (LOW,VERIFY,SUBSTR,ADDR,DATE,STG,CSTG) BUILTIN;
DCL S BIN FIXED(31);
DCL TEMP CHAR(8);
DCL LROW CHAR(133);
DCL CNT BIN FIXED(15) INIT(1);
DCL KMSG CHAR(8Ø);
DCL RESP     BIN FIXED(31),
    RESP2    BIN FIXED(31),
    TOKEN    CHAR(8),
    OUTLEN   BIN FIXED(31) INIT(133);
/* ***************** PROGRAM ********************** */
TOKEN = LOW(8);
TEMP = 'QMFREPT';
EXEC CICS READQ TS QUEUE(TEMP) INTO(LROW) ITEM(CNT) RESP(S);
IF S ¬= DFHRESP(NORMAL)
THEN DO;
   KMSG = 'NO DATA FOR PRINTING';
   EXEC CICS SEND FROM(KMSG) ERASE;
   EXEC CICS RETURN;
END;
EXEC CICS SPOOLOPEN OUTPUT TOKEN(TOKEN)
                    NODE('*') USERID('*')
                    CLASS('A') RECORDLENGTH(133)
                    ASA PRINT
                    NOHANDLE;
IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
  KMSG = 'ERROR AT SPOOL OPENING';
  EXEC CICS SEND FROM(KMSG) ERASE;
  EXEC CICS RETURN;
END;
DO WHILE (S = DFHRESP(NORMAL));
  EXEC CICS SPOOLWRITE TOKEN(TOKEN)
                       FROM(LROW) FLENGTH(OUTLEN)
                       LINE
                       NOHANDLE;
  IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
    KMSG = 'ERROR AT SPOOL WRITING';
    EXEC CICS SEND FROM(KMSG) ERASE;
    EXEC CICS RETURN;
```

```
  END;
  CNT = CNT + 1;
  EXEC CICS READQ TS QUEUE(TEMP) INTO(LROW) ITEM(CNT) RESP(S);
END;
EXEC CICS SPOOLCLOSE TOKEN(TOKEN) NOHANDLE;
IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
  KMSG = 'ERROR AT SPOOL CLOSING';
  EXEC CICS SEND FROM(KMSG) ERASE;
  EXEC CICS RETURN;
END;
EXEC CICS DELETEQ TS QUEUE(TEMP);
KMSG = 'PRINTING COMPLETE';
EXEC CICS SEND FROM(KMSG) ERASE;
EXEC CICS RETURN;
END PRINTTS;
```

PRINTTD

```
PRINTTD: PROC OPTIONS(MAIN);
/* ************************************************ */
/*    PROGRAM READS TD DPRI (CREATED IN QMF)        */
/*    - DEPENDING ON THE SELECTED OPTION FORMS      */
/*      PRINTOUT (TO JES SPOOL OR LOCAL CICS        */
/*                PRINTER, EG PØØ1)                 */
/*          - * (BETA92)                            */
/*          - ! (PØØ1)                              */
/*          - XXXX (JES PRINTER NAME)               */
/*      - CALLS QMF AT THE END                      */
/*    NOTE: CHANGE NODE NAME BELLOW                 */
/*    TRANS: PRØ2                                   */
/* ************************************************ */
DCL (LOW,VERIFY,SUBSTR,ADDR,DATE,STG,CSTG) BUILTIN;
DCL S BIN FIXED(31);
DCL TEMP CHAR(8);
DCL ZONA CHAR(8);
DCL TERM CHAR(4);
DCL RED CHAR(133);
DCL KMSG CHAR(8Ø);
DCL DUZ      BIN FIXED(15) INIT(133);
DCL RESP     BIN FIXED(31),
    RESP2    BIN FIXED(31),
    TOKEN    CHAR(8),
    OUTLEN   BIN FIXED(31) INIT(133);
/* ************************************************ */
%INCLUDE LPRINT;  /* MAP                            */
%INCLUDE DFHAID;
%INCLUDE DFHBMSCA;
/* ***************** PROGRAM ********************** */
TOKEN = LOW(8);
```

```
TEMP = 'DPRI';
EXEC CICS RECEIVE MAP('LPRINT1') MAPSET('LPRINT') RESP(S);
IF S = DFHRESP(MAPFAIL) THEN DO;
  EXEC CICS SEND MAP('LPRINT1') MAPSET('LPRINT') ERASE;
  EXEC CICS RETURN TRANSID('PRØ2');
END;
IF EIBAID = DFHPF3 THEN DO;
  EXEC CICS START TRANSID('QMFE') TERMID(EIBTRMID) NOHANDLE;
  IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
    MESO = 'QMF NOT AVAILABLE - CALL DB2 ADMINISTRATOR';
    EXEC CICS SEND MAP('LPRINT1') MAPSET('LPRINT') DATAONLY;
    EXEC CICS RETURN TRANSID('PRØ2');
  END;
  ELSE EXEC CICS RETURN;
END;
/* ************************************************ */
EXEC CICS READQ TD QUEUE(TEMP) INTO(RED) RESP(S);
IF S ¬= DFHRESP(NORMAL)
THEN DO;
  MESO = 'NO DATA FOR PRINTING';
  EXEC CICS SEND MAP('LPRINT1') MAPSET('LPRINT') DATAONLY;
  EXEC CICS RETURN TRANSID('PRØ2');
END;
IF SUBSTR(PRTI, 1, 1) ¬= '!' THEN DO;
  IF SUBSTR(PRTI, 1, 1) = '*' THEN DO;
    EXEC CICS SPOOLOPEN OUTPUT TOKEN(TOKEN)
                        NODE('*') USERID('*')
                        CLASS('I') RECORDLENGTH(133)
                        ASA PRINT
                        NOHANDLE;
  END;
  ELSE DO;
    EXEC CICS SPOOLOPEN OUTPUT TOKEN(TOKEN)
                        NODE('PSJESØØ1') USERID(PRTI)
                        CLASS('A') RECORDLENGTH(133)
                        ASA PRINT
                        NOHANDLE;
  END;
  IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
    MESO = 'ERROR AT SPOOL OPENING';
    EXEC CICS SEND MAP('LPRINT1') MAPSET('LPRINT') DATAONLY;
    EXEC CICS RETURN TRANSID('PRØ2');
  END;
END;
DO WHILE (S = DFHRESP(NORMAL));
  IF SUBSTR(PRTI, 1, 1) ¬= '!' THEN DO;
    EXEC CICS SPOOLWRITE TOKEN(TOKEN)
                         FROM(RED) FLENGTH(OUTLEN)
                         LINE
                         NOHANDLE;
```

```
    IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
      MESO = 'ERROR AT SPOOL WRITING';
      EXEC CICS SEND MAP('LPRINT1') MAPSET('LPRINT') DATAONLY;
      EXEC CICS RETURN TRANSID('PRØ2');
    END;
  END;
  ELSE DO;
    EXEC CICS WRITEQ TD QUEUE('DPR1') FROM(RED) LENGTH(DUZ) RESP(S);
    IF S ¬= DFHRESP(NORMAL)
    THEN DO;
      MESO = 'ERROR AT WRITING TO DPR1';
      EXEC CICS SEND MAP('LPRINT1') MAPSET('LPRINT') DATAONLY;
      EXEC CICS RETURN TRANSID('PRØ2');
    END;
  END;
  EXEC CICS READQ TD QUEUE(TEMP) INTO(RED) RESP(S);
END;
IF SUBSTR(PRTI, 1, 1) ¬= '!' THEN DO;
  EXEC CICS SPOOLCLOSE TOKEN(TOKEN) NOHANDLE;
  IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
    MESO = 'ERROR AT SPOOL CLOSING';
    EXEC CICS SEND MAP('LPRINT1') MAPSET('LPRINT') DATAONLY;
    EXEC CICS RETURN TRANSID('PRØ2');
  END;
END;
MESO = 'PRINTING COMPLETE';
EXEC CICS SEND MAP('LPRINT1') MAPSET('LPRINT') DATAONLY;
EXEC CICS RETURN TRANSID('PRØ2');
END PRINTTD;
```

PRINTTD1

```
PRINTTD1: PROC OPTIONS(MAIN);
/* ************************************************ */
/*    PROGRAM IS CALLED FROM QMF (SYNONYM: PRTQMFTD)  */
/*    - CHECKS USERID AND TERMID FROM WHERE IT'S      */
/*       CALLED AND CALLS TRANSACTION PRØ2 (PRINTTD)  */
/*    - WRITES THE LIST OF ACTIVE TERMINALS TO TS     */
/*       PRØ3NNNN (BROWSE WITH CEBR)                  */
/*    TRANS: PRØ3                                     */
/* ************************************************ */
DCL (LOW,VERIFY,SUBSTR,ADDR,DATE,STG,CSTG) BUILTIN;
DCL KMSG CHAR(8Ø);
DCL (TERM, TRAN) CHAR(4);
DCL (USER1, USER2) CHAR(8);
DCL COUNTER1 BIN FIXED(31) INIT(Ø);
DCL IDRED CHAR(8) INIT('PRØ3NNNN');
DCL 1 TSSL,
    2 TSTERM          CHAR(4),
```

```
    2 TSTRAN          CHARACTER(4),
    2 TSUSER          CHARACTER(8);
/* ************************************************ */
%INCLUDE DFHAID;
%INCLUDE DFHBMSCA;
/* ***************** PROGRAM ********************** */
EXEC CICS IGNORE CONDITION QIDERR;
EXEC CICS ASSIGN USERID(USER1);
EXEC CICS DELETEQ TS QUEUE(IDRED);
EXEC CICS INQUIRE TERMINAL START;
DO WHILE (EIBRESP ¬= DFHRESP(END));
  EXEC CICS INQUIRE TERMINAL(TERM) NEXT TRANSACTION(TRAN)
                                        USERID(USER2);
  COUNTER1 = COUNTER1 + 1;
  TSTERM  = TERM;
  TSTRAN  = TRAN;
  TSUSER  = USER2;
  EXEC CICS WRITEQ TS QUEUE(IDRED)
                      FROM(TSSL)
                      ITEM(COUNTER1) AUXILIARY;
/* ************************************************ */
/*    RESTRICTION: 1 USER = 1 SESSION               */
/* ************************************************ */
  IF USER1 = USER2 THEN DO;
    EXEC CICS INQUIRE TERMINAL END;
    EXEC CICS START TRANSID('PRØ2') TERMID(TERM) NOHANDLE;
    IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
      KMSG = 'ERROR CALLING PRØ2';
      EXEC CICS SEND FROM(KMSG) ERASE;
    END;
    EXEC CICS RETURN;
  END;
END;
EXEC CICS INQUIRE TERMINAL END;
KMSG = 'INACTIVE TRANSACTION';
EXEC CICS SEND FROM(KMSG) ERASE;
EXEC CICS RETURN;
END PRINTTD1;
```

PRINTTD2

```
PRINTTD2: PROC OPTIONS(MAIN);
/* ************************************************ */
/*    PROGRAM IS CALLED WHEN TD DPR1 IS FILLED WITH */
/*    3Ø ROWS                                       */
/*      - TRIGGER IN TD DPR1 DEFINITION             */
/*    TRANS: PRØ4                                   */
/* ************************************************ */
DCL (LOW,VERIFY,SUBSTR,ADDR,DATE,STG,CSTG) BUILTIN;
```

```
DCL S BIN FIXED(31);
DCL LROW CHAR(133);
DCL QID  CHAR(8);
DCL L        BIN FIXED(15) INIT(133);
DCL RESP     BIN FIXED(31),
    RESP2    BIN FIXED(31),
    TOKEN    CHAR(8),
    OUTLEN   BIN FIXED(31) INIT(133);
/* ***************** PROGRAM ********************** */
EXEC CICS ASSIGN QNAME(QID) RESP(S);
IF S ¬= DFHRESP(NORMAL)
THEN DO;
  LROW = 'ERROR IN ASSIGN QNAME';
  EXEC CICS SEND FROM(LROW) LENGTH(L) ERASE;
  EXEC CICS RETURN;
END;
EXEC CICS READQ TD QUEUE(QID) INTO(LROW) LENGTH(L) RESP(S);
IF S ¬= DFHRESP(NORMAL)
THEN DO;
  LROW = 'NO DATA FOR PRINTING';
  EXEC CICS SEND FROM(LROW) LENGTH(L) ERASE;
  EXEC CICS RETURN;
END;
TOKEN = LOW(8);
EXEC CICS SPOOLOPEN OUTPUT TOKEN(TOKEN)
                    NODE('*') USERID('*')
                    CLASS('A') RECORDLENGTH(133)
                    ASA PRINT
                    NOHANDLE;
IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
  LROW = 'ERROR AT SPOOL OPENING';
  EXEC CICS SEND FROM(LROW) LENGTH(L) ERASE;
  EXEC CICS RETURN;
END;
DO WHILE (S ¬= DFHRESP(QZERO));
  EXEC CICS SPOOLWRITE TOKEN(TOKEN)
                       FROM(LROW) FLENGTH(OUTLEN)
                       LINE
                       NOHANDLE;
  IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
    LROW = 'ERROR AT SPOOL WRITING';
    EXEC CICS SEND FROM(LROW) LENGTH(L) ERASE;
    EXEC CICS RETURN;
  END;
  EXEC CICS READQ TD QUEUE(QID) INTO(LROW) LENGTH(L) RESP(S);
END;
EXEC CICS SPOOLCLOSE TOKEN(TOKEN) NOHANDLE;
IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
  LROW = 'ERROR AT SPOOL CLOSING';
  EXEC CICS SEND FROM(LROW) LENGTH(L) ERASE;
```

```
END;
EXEC CICS RETURN;
END PRINTTD2;
```

DDLDML

Run from SPUFI.

```
CREATE TABLE Q.CCOMMAND_SYNONYMS
  (VERB               CHAR(18)       NOT NULL,
   OBJECT             VARCHAR(31),
   SYNONYM_DEFINITION VARCHAR(254)   NOT NULL,
   REMARKS            VARCHAR(254))
   IN DSQDBDEF.DSQTSDEF;
COMMENT ON TABLE Q.CCOMMAND_SYNONYMS IS
  'QMF COMMAND SYNONYM TABLE';
COMMENT ON COLUMN Q.CCOMMAND_SYNONYMS.VERB IS
  'NAME OF THE VERB';
COMMENT ON COLUMN Q.CCOMMAND_SYNONYMS.OBJECT IS
  'NAME OF THE OBJECT';
COMMENT ON COLUMN Q.CCOMMAND_SYNONYMS.SYNONYM_DEFINITION IS
  'DEFINITION OF SYNONYM';
COMMENT ON COLUMN Q.CCOMMAND_SYNONYMS.REMARKS IS
  'COMMENTS ABOUT SYNONYM';
CREATE UNIQUE INDEX Q.CCOMMAND_SYNONYMSX ON Q.CCOMMAND_SYNONYMS
  (VERB ASC , OBJECT ASC)
  USING STOGROUP DSQSGDEF
  CLOSE NO;
GRANT SELECT ON Q.CCOMMAND_SYNONYMS TO PUBLIC;
CREATE TABLE Q.MY_PFKEYS
   (PANEL               CHAR(18)       NOT NULL,
    ENTRY_TYPE          CHAR(1)        NOT NULL,
    NUMBER              SMALLINT       NOT NULL,
    PF_SETTING          VARCHAR(254),
    REMARKS             VARCHAR(254))
 IN DSQDBDEF.DSQTSDEF;
 COMMENT ON TABLE Q.MY_PFKEYS IS 'PF KEYS FOR CICS';
 CREATE UNIQUE INDEX Q.MY_PFKEYSX
           ON Q.MY_PFKEYS (PANEL, ENTRY_TYPE, NUMBER);
 INSERT INTO Q.MY_PFKEYS (PANEL, ENTRY_TYPE, NUMBER, PF_SETTING)
 VALUES('REPORT', 'K', 4, 'PRTQMF');
 UPDATE Q.PROFILES
   SET PFKEYS = 'Q.MY_PFKEYS'
   WHERE CREATOR='SYSTEM'
   AND TRANSLATION = 'ENGLISH'
   AND ENVIRONMENT = 'CICS';
 GRANT SELECT ON Q.MY_PFKEYS TO PUBLIC;
 CREATE TABLE Q.MY_PFKEYSTD
   (PANEL               CHAR(18)       NOT NULL,
```

```
    ENTRY_TYPE          CHAR(1)        NOT NULL,
    NUMBER              SMALLINT       NOT NULL,
    PF_SETTING          VARCHAR(254),
    REMARKS             VARCHAR(254))
  IN DSQDBDEF.DSQTSDEF;
  COMMENT ON TABLE Q.MY_PFKEYSTD IS 'PF KEYS FOR CICS';
  CREATE UNIQUE INDEX Q.MY_PFKEYSTDX
           ON Q.MY_PFKEYSTD (PANEL, ENTRY_TYPE, NUMBER);
  INSERT INTO Q.MY_PFKEYSTD (PANEL, ENTRY_TYPE, NUMBER, PF_SETTING)
  VALUES('REPORT', 'K', 4, 'PRTQMFTD');
  UPDATE Q.PROFILES
    SET PFKEYS = 'Q.MY_PFKEYSTD'
    WHERE CREATOR='SYSADM'
    AND TRANSLATION = 'ENGLISH'
    AND ENVIRONMENT = 'CICS';
  GRANT SELECT ON Q.MY_PFKEYSTD TO PUBLIC;
--INSERT INTO Q.PROFILES
--VALUES('SYSADM', 'UPPER', 'COMMA', 'YES', '132', '60', 'SQL',
--'DSQDBDEF.DSQTSDEF', 'NONE', NULL, 'ENGLISH', 'Q.MY_PFKEYSTD',
--'Q.CCOMMAND_SYNONYMS', 'SYSTEM', NULL, 'CICS')
```

JCLCICS

```
//PROG0011 JOB (ACCT),NAME,NOTIFY=&SYSUID,
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//STEP0001  EXEC PGM=IEFBR14
// F PSTEST29,'CEMT SET FILE(DFHCSD) CLO DIS'
/*
//*************************************************************
//* DEFINE QMF PROGRAMS, TRANSACTION, AND PANEL FILE TO CICS. *
//*************************************************************
//STEP0002  EXEC PGM=DFHCSDUP,
//          PARM='CSD(READWRITE),NOCOMPAT',COND=(4,LT)
//STEPLIB   DD DSN=CICSTS22.CICS.SDFHLOAD,DISP=SHR
//DFHCSD    DD DSN=CICSTS22.CICS.PSTEST29.DFHCSD,DISP=SHR
//SYSUT1    DD UNIT=SYSDA,SPACE=(1024,(100,100))
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
   DEF MAPSET(LPRINT) GROUP(QMF)
   DEF PROGRAM(PRINTTD) GROUP(QMF) L(LE370) DA(A)
   DEF PROGRAM(PRINTTD1) GROUP(QMF) L(LE370) DA(A)
   DEF PROGRAM(PRINTTD2) GROUP(QMF) L(LE370) DA(A)
   DEF PROGRAM(PRINTTS) GROUP(QMF) L(LE370) DA(A)
   DEFINE TRANSACTION(PR01) PROGRAM(PRINTTS)
         GROUP(QMF) TASKDATAL(ANY)
   DEFINE TRANSACTION(PR02) PROGRAM(PRINTTD)
         GROUP(QMF) TASKDATAL(ANY)
   DEFINE TRANSACTION(PR03) PROGRAM(PRINTTD1)
         GROUP(QMF) TASKDATAL(ANY)
```

```
   DEFINE TRANSACTION(PRØ4) PROGRAM(PRINTTD2)
         GROUP(QMF) TASKDATAL(ANY)
   DEFINE TDQUEUE(DPRI) GROUP(QMF)
          TYPE(INTRA) ATIFACILITY(TERMINAL) RECOVSTATUS(LOGICAL)
          FACILITYID(DPRI) TRIGGERLEVEL(1)
   DEFINE TDQUEUE(DPR1) GROUP(QMF)
          TYPE(INTRA) ATIFACILITY(FILE) RECOVSTATUS(LOGICAL)
          TRANSID(PRØ4) TRIGGERLEVEL(3Ø)
/*
//STEPØØØ3  EXEC PGM=IEFBR14,COND=(4,LT)
// F PSTEST29,'CEMT SET FILE(DFHCSD) OPE ENA'
/*
//
```

Nikola Lazovic, Gordana Kozovic
DB2 System Administrators
Postal Savings Bank (Serbia and Montenegro)

CICS transaction list

We have been asked to publish the list of transactions and the IBM documents that refer to them. This list was compiled by Doc Farmer for his article 'CICS transaction segregation and region creation', published in *CICS Update* in March and April this year.

No.	Trans	*RACF Security Guide*	*Resource Definition Guide*	*CICS Supplied Trans*
1	AADD	-	X	-
2	ABRW	-	X	-
3	AC01	-	X	-
4	AC02	-	X	-
5	AC03	-	X	-
6	AC05	-	X	-
7	AC06	-	X	-
8	AC20	-	X	-
9	AC21	-	X	-
10	AC22	-	X	-
11	AC23	-	X	-
12	AC24	-	X	-
13	AC25	-	X	-

14	AC26	-	X	-
15	AC27	-	X	-
16	AC28	-	X	-
17	AC2A	-	X	-
18	AC2C	-	X	-
19	AC2D	-	X	-
20	AC2E	-	X	-
21	AC2F	-	X	-
22	ACCT	-	X	-
23	ACEL	-	X	-
24	ACLG	-	X	-
25	ADDS	-	X	-
26	ADYN	-	X	-
27	AINQ	-	X	-
28	AMNU	-	X	-
29	AORD	-	X	-
30	AORQ	-	X	-
31	AREP	-	X	-
32	ASMC	-	X	-
33	ASME	-	X	-
34	AUPD	-	X	-
35	BRWS	-	X	-
36	CAFB	X	X	X
37	CAFF	X	X	X
38	CAQP	-	-	X
39	CATA	X	X	X
40	CATD	X	X	X
41	CATP	-	-	X
42	CATR	X	X	X
43	CATS	-	X	X
44	CAUT	-	X	X
45	CBAM	X	X	X
46	CBRA	-	-	X
47	CBRC	-	X	-
48	CCIN	X	X	X
49	CCMF	-	X	X
50	CCMS	-	-	X
51	CCU2	-	-	X
52	CDBC	X	X	X
53	CDBD	X	X	X
54	CDBF	X	X	X
55	CDBI	X	X	X
56	CDBM	X	-	X
57	CDBN	-	X	X
58	CDBO	X	X	X
59	CDBQ	X	X	X
60	CDBT	X	X	X
61	CDFS	X	X	X
62	CDST	-	-	X

63	CDTS	X	X	X
64	CEBR	X	X	X
65	CEBT	-	-	X
66	CECI	X	X	X
67	CECS	X	X	X
68	CEDA	X	X	X
69	CEDB	X	X	X
70	CEDC	X	X	X
71	CEDF	X	X	X
72	CEDX	X	-	X
73	CEGN	X	X	X
74	CEHP	X	X	X
75	CEHS	X	X	X
76	CEID	-	-	X
77	CEJR	X	X	X
78	CEKL	-	-	X
79	CEMS	-	-	X
80	CEMT	X	X	X
81	CEOS	-	-	X
82	CEOT	X	X	X
83	CEPC	-	-	X
84	CEPE	-	-	X
85	CEPQ	-	-	X
86	CEPW	-	-	X
87	CESC	X	X	X
88	CESD	X	X	X
89	CESF	X	X	X
90	CESN	X	X	X
91	CEST	X	X	X
92	CETR	X	X	X
93	CEX2	X	X	X
94	CFCL	X	-	X
95	CFOR	X	-	X
96	CFQR	X	-	-
97	CFQS	X	-	X
98	CFTI	-	-	X
99	CFTL	X	X	X
100	CFTM	-	-	X
101	CFTS	X	X	X
102	CFUP	-	-	X
103	CGRP	X	-	X
104	CHLP	-	-	X
105	CIEP	X	X	X
106	CIND	X	X	X
107	CINS	-	-	X
108	CIOD	-	-	X
109	CIOF	-	-	X
110	CIOR	-	-	X
111	CIRB	-	-	X
112	CIRD	-	-	X

113	CIRP	X	X	X
114	CIRR	X	X	X
115	CITS	X	-	X
116	CJTR	X	-	X
117	CLEP	-	-	X
118	CLER	-	-	X
119	CLQ2	X	X	X
120	CLR1	X	X	X
121	CLR2	X	X	X
122	CLS1	X	X	X
123	CLS2	X	X	X
124	CLS3	X	X	X
125	CLS4	X	X	X
126	CMAC	X	X	X
127	CMPX	X	X	X
128	CMSG	X	X	X
129	CMTS	X	X	X
130	COBC	-	X	-
131	COBE	-	X	-
132	COVR	X	-	X
133	CPLT	X	-	X
134	CPMI	X	X	X
135	CPSA	-	-	X
136	CPSB	-	-	X
137	CPSC	-	-	X
138	CPSD	-	-	X
139	CPSS	X	-	-
140	CQPI	X	X	X
141	CQPO	X	X	X
142	CQRY	X	X	X
143	CRDR	-	-	X
144	CREA	X	-	X
145	CREC	X	-	X
146	CRMD	X	-	X
147	CRMF	X	-	X
148	CRPA	X	X	X
149	CRPC	X	X	X
150	CRPM	X	X	X
151	CRSQ	X	X	X
152	CRSR	X	X	X
153	CRSY	X	X	X
154	CRTE	X	X	X
155	CRTX	X	X	X
156	CSAC	X	X	X
157	CSCY	X	X	X
158	CSFE	X	X	X
159	CSFR	X	-	X
160	CSFU	X	X	X
161	CSGM	X	X	X
162	CSGX	-	X	-

163	CSHA	X	-	X
164	CSHQ	X	-	X
165	CSHR	X	X	X
166	CSIR	-	X	X
167	CSJC	-	X	X
168	CSKD	-	-	X
169	CSKE	-	-	X
170	CSKL	-	-	X
171	CSKP	X	X	X
172	CSLG	-	X	-
173	CSM1	X	X	X
174	CSM2	X	X	X
175	CSM3	X	X	X
176	CSM5	X	X	X
177	CSMI	X	X	X
178	CSMT	-	X	X
179	CSNC	X	X	X
180	CSNE	X	X	X
181	CSOL	X	-	X
182	CSOT	-	X	X
183	CSPG	X	X	X
184	CSPK	X	X	X
185	CSPP	X	X	X
186	CSPQ	X	X	X
187	CSPS	X	X	X
188	CSQC	X	X	X
189	CSRK	X	X	X
190	CSRS	X	X	X
191	CSSC	-	X	X
192	CSSF	X	X	X
193	CSSN	-	X	X
194	CSST	-	X	X
195	CSSX	-	X	-
196	CSSY	X	-	X
197	CSTA	-	X	X
198	CSTB	-	-	X
199	CSTE	X	X	X
200	CSTP	X	-	X
201	CSTT	-	X	X
202	CSZI	X	X	X
203	CTIN	X	X	X
204	CTSD	X	-	X
205	CVMI	X	X	X
206	CVST	-	X	X
207	CWBA	X	X	X
208	CWBC	-	X	X
209	CWBG	X	X	X
210	CWBM	-	X	X
211	CWTO	X	X	X
212	CWTR	-	-	X
213	CWXN	X	X	X

214	CXCU	X	X	X
215	CXRE	X	X	X
216	CXRT	X	X	X
217	DADD	-	X	-
218	DBRW	-	X	-
219	DELQ	-	X	-
220	DINQ	-	X	-
221	DMNU	-	X	-
222	DORD	-	X	-
223	DORQ	-	X	-
224	DREP	-	X	-
225	DSNC	X	X	X
226	DUPD	-	X	-
227	EXCI	-	X	-
228	HPJC	-	X	-
229	ICIC	-	X	-
230	IFBL	-	X	-
231	IFBR	-	X	-
232	IMSN	-	X	-
233	IMSO	-	X	-
234	INQY	-	X	-
235	IQRD	-	X	-
236	IQRL	-	X	-
237	IQRR	-	X	-
238	IQXL	-	X	-
239	IQXR	-	X	-
240	MENU	-	X	-
241	OREN	-	X	-
242	OREQ	-	X	-
243	PADD	-	X	-
244	PBRW	-	X	-
245	PINQ	-	X	-
246	PLIC	-	X	-
247	PLIE	-	X	-
248	PMNU	-	X	-
249	PORD	-	X	-
250	PORQ	-	X	-
251	PPKO	-	X	-
252	PPLA	-	X	-
253	PREP	-	X	-
254	PUPD	-	X	-
255	REPT	-	X	-
256	TDWT	-	X	-
257	UPDT	-	X	-
258	XPKO	-	X	-
259	XPLA	-	X	-

IBM has announced Version 1.1 of its CICS Configuration Manager for z/OS. The product makes administering and maintaining CICS resource definitions easier. The resource definitions can be managed across multiple CICS regions, including regions that are controlled by CICSPlex SM. The resource definitions can be edited, compared, or migrated from one CSD file or CICSPlex SM data repository (DREP) to another. The program's change packaging techniques allow administrators to group sets of CICS resource definitions into a single logical unit, and then to promote or backout changes associated with the group.

An XML SOAP-based programmable interface is provided, allowing definition management to be included in automated processes.

Facilities in this release include options for defining approval requirements before changes are implemented, while maintaining the integrity of the change package. This means that controlled access can be given to chosen users, such as an application development department.

An audit trail of changed resources, dates, and times is provided, and a range of reports can be produced. A comparison facility allows administrators to view differences between resource definitions within various CICS environments. A CICS configuration can be scanned for definitions matching search criteria or particular values.

For further information contact your local IBM representative.
URL: www-306.ibm.com/software/htp/cics/cm.

***

California Software has announced INFINITE zSERIES, which it claims will migrate mainframe applications to Windows, Unix, or Linux.

INFINITE zSERIES includes tools for migrating COBOL, CICS, JCL, and tools for recreating MVS/VSE, OS/390, and z/OS calls. INFINITE zSERIES is bundled with INFINITE NET, which creates a graphical UI and Web-enables the newly-migrated application, automatically.

Additionally, INFINITE zSERIES includes INFINITE OLAP, a data warehouse and multi-dimensional data analysis tool.

For further information contact:
California Software, 1241 Puerta Del Sol, San Clemente, CA 92673, USA.
Tel: (949) 498 9300.
URL: www.calsw.com/calsw/infinitezseries.asp.

***

IBM has announced a range of Tivoli OMEGAMON XE monitoring solutions. This includes products to monitor and manage CICS, as well as z/OS Unix System Services and Parallel Sysplex, z/VM, DB2, IMS, zSeries networks, storage, WebSphere Application Server (WAS), WebSphere Integration Brokers, and WebSphere MQ.

The suite of products incorporates IBM's acquisition from Candle.

For further information contact your local IBM representative.
URL: www.ibm.com/ondemand.

***