181

CICS update

December 2000

In this issue

3 Java hot-pooling support

14 Verification of VSAM forward recovery

28 Expanding COBOL COPYs before compilation

32 CICS on the VSE/ESA platform: e-business support

48 CICS news

CICS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *CICS Update*, comprising twelve monthly issues, costs £175.00 in the UK; $270.00 in the USA and Canada; £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 January 1994 issue, are available separately to subscribers for £16.00 ($23.50) each including postage.

CICS Update on-line

Code from *CICS Update* can be downloaded from our Web site at http://www.xephon.com/cicsupdate.html; you will need the user-id shown on your address label.

Editor

Trevor Eddolls

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Contributions

Articles published in *CICS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Printed in England.

Java hot-pooling support

INTRODUCTION

IBM has recently shipped a number of changes to CICS Transaction Server for OS/390 Release 1.3 (CICS TS 1.3) that provide support for Java 'hot-pooling'. This article describes the background to hot-pooling within CICS, why the changes have been made, and what benefits they bring for CICS Java applications. The article also describes the service requirements necessary to use hot-pooling, the documentation changes to the CICS manuals, and the diagnostic enhancements to CICS (such as the new messages and dump codes) that are associated with hot-pooling support.

BACKGROUND TO JAVA SUPPORT IN CICS TS 1.3

Java was developed by Sun Microsystems as an object-oriented language. It initially came to prominence for use in coding applets to run on World Wide Web pages. Java's popularity has increased enormously in the past few years, to the extent that it is becoming pervasive in many areas of the computing industry, appearing on workstation, mid-range, and mainframe platforms.

CICS TS 1.3 supports two programming models for Java application development and deployment. 'Traditional' Java programs are passed through a compiler that generates platform-neutral bytecodes from their application source code. These Java bytecode files are known as class files. The bytecodes are interpreted by platform-specific Java Virtual Machines (JVMs) that convert the portable Java class files into executable machine code running on a particular computer operating system. This interpretation occurs at run-time.

An example of a Java source and class file relationship would be source file BankAccount.java being compiled into bytecodes, producing an interpretable class file called BankAccount.class.

CICS TS 1.3 provides support for a JVM for the MVS operating system environment, running under CICS' control, and interpreting Java applications that have been written and then compiled into such bytecodes. CICS TS 1.3 also supports another Java execution environment, however. Using the Enterprise Toolkit for OS/390 (ET/390), as supplied with IBM's VisualAge for Java product, Java class files in bytecode format can be further compiled into S/390 machine code and bound into target PDSE libraries as program objects. These can then be executed under CICS in a similar manner to applications written in other compiled languages such as COBOL and C++. The compiler component of ET/390 is known as the High Performance Java compiler (also referred to as HPJ).

As with all things in life, there is a trade-off to be made here between the two types of execution environment that are available for Java application programs executing in CICS TS 1.3. The binding of class files into S/390 machine code by the ET/390 HPJ compiler means that such compiled Java programs are no longer platform independent. Of course, the originating class files can also be interpreted under a platform-independent JVM environment if so desired.

Readers interested in further background details to this area of CICS are referred to my previous article *Java Support in CICS Transaction Server 1.3*, which appeared in the January 2000 edition of *CICS Update*.

WHAT IS HOT-POOLING?

HPJ-compiled Java programs are referred to as Java program objects. When such a program object is run in the CICS environment, it executes in a Language Environment (LE) 'run-unit'. A more generic term that is used to represent a run-unit is an LE 'enclave'. The word enclave has tended to be associated with the use of batch-LE rather than CICS-LE environments.

CICS TS 1.3 was shipped with support for such Java program objects, as described above. Such applications executed as CICS-LE programs. In this original support model, the run-unit or enclave for a Java program object was constructed and initialized for each invocation of the program. When the program terminated, the run-unit and environment were terminated too.

In order to improve the response time and throughput of Java program objects executing under CICS TS 1.3, and to reduce the performance overhead in loading and initializing the environment for those programs, a number of enhancements have been made. Specifically, function has been added to CICS TS 1.3 to provide support for a preinitialized enclave, which may persist and therefore get reused by multiple invocations of Java program objects. This support is referred to as hot-pooling.

By reusing preinitialized enclaves for a number of executions of Java program objects, the CPU cost and instruction-pathlength required to execute the programs can be spread across the invocations. This is particularly beneficial when the pathlength to load and initialize the run-time support for compiled Java, and to load the application and CICS Java classes, can be shared between invocations of Java program objects. This cost is reduced asymptotically – the more often an enclave can be reused, the lower the average cost per program reusing the enclave when preparing the environment for execution.

HOT-POOLING AND THE OPEN TRANSACTION ENVIRONMENT

Hot-pooling exploits the LE preinitialization services (PIPI) to construct the preinitialized and reusable enclave for use by Java program objects running under CICS TS 1.3. This enclave utilizes batch-LE services. It also exploits a feature of CICS TS 1.3 known as the Open Transaction Environment (OTE) to manage the hot-pooled program objects under a new type of Task Control Block (TCB) running within CICS.

OTE has already been utilized by CICS TS 1.3 to provide the TCB environment for JVMs to execute under. Such OTE-managed TCBs are referred to as open TCBs, and run independently of the CICS quasi-reentrant (QR) TCB that is used for traditional CICS application programs to execute on. Open TCBs for JVMs are known as J8 TCBs. Similarly, open TCBs for hot-pooling of Java program objects are known as H8 TCBs. (The letter is self-explanatory; the number denotes the storage protection key of a program using such a TCB). Each two-character open TCB name is known as an OTE mode. The different types of mode have their own specific purposes; OTE handles them in different ways. The type of mode utilized for a given sort of program (eg JVM, hot-pooled program object) is determined by CICS, not by the application program itself.

By providing a unique TCB for a program, documented CICS restrictions on the use of certain MVS services that can suspend a TCB are lifted. Such suspension of the QR TCB would have a detrimental effect upon system throughput and performance – by limiting the effect of a suspension (known as the 'blocking' of a TCB) to a particular application running under its own TCB, the impact does not affect other users of the CICS system.

CICS controls the total number of OTE open TCBs (for all types of mode) by means of the MAXOPENTCBS system initialization parameter. There may be many open TCBs allocated concurrently (up to the limiting value of MAXOPENTCBS), provided there is sufficient virtual storage available for CICS to satisfy them. A given CICS task can have only one TCB of each OTE mode. The TCB is retained by that task from the time that it is allocated until the end of the task, at which time it may be allocated to another task or destroyed. All such TCB management is controlled by CICS OTE logic: again, the application program has no control over this activity.

H8 TCB ALLOCATION RULES

There are specific rules followed by CICS when allocating an H8 TCB for use by a hot-pooled Java program object. The preference order is outlined below.

If the transaction has already been allocated an H8 TCB then this is used for the hot-pooled Java program object. If not, CICS will look for a free H8 TCB. Free H8 TCBs are those not currently allocated to tasks. However, CICS will initially utilize only one of the free H8 TCBs that has an association with the particular transaction identifier and program name of the transaction/Java program object that is to be executed. In other words, the free TCB's enclave has been previously used by the same transaction/program combination. Such a selection best optimizes the reuse of a preinitialized LE enclave environment for that transaction/program. If no such free H8 TCB exists, then CICS will create a new one, subject to the limitation of MAXOPENTCBS. If MAXOPENTCBS has been reached, CICS will try to use a free H8 TCB that has a matching program name. Failing that, it will try to use a free H8 TCB with neither matching program name nor matching transaction identifier. If no such free H8 TCB exists, then CICS will steal a free open TCB of another OTE mode (eg J8) if one exists; this results in the other open TCB being destroyed and a new H8 TCB instantiated. Finally, if no such free open TCBs exist at all, then CICS has to suspend the task until one becomes available.

By maintaining an association with an H8 TCB and its previously-used program name and transaction identifier, it is possible for there to be a number of free H8 TCBs that match the same combination of program and transid. Such a pool of free H8 TCBs is beneficial when there is a considerable throughput of the same CICS transaction, since it optimizes the reuse of the correct LE enclave for that program.

HOT-POOLING DEFINITION AND USAGE CONSIDERATIONS

Java program objects can be defined to CICS to execute in either a hot-pooled manner or in a traditional execution environment (that is, either within a reusable enclave under an H8 TCB, or in a newly created and initialized LE run-unit executing under the QR TCB). The CICS Resource Definition Online (RDO) option Program provides a new attribute – Hotpool. If this is set to Yes then the Java program object is hot-pooled. Note: the default value is No. This means that existing definitions for Java program objects will not automatically become eligible for hot-pooling when the PTF that enables hot-pooling support in CICS TS 1.3 is applied to the system. It requires a deliberate act by the CICS systems programmer to make such a definition, as would be expected since any such change should only be implemented after careful planning and testing activity has been completed.

It should be noted that Java program objects that are to be defined as

Hotpool(Yes) must also be defined as Concurrency(Threadsafe). This means that such Java programs must utilize appropriate serialization techniques when accessing any resources that may be shared with other programs executing concurrently on other TCBs under CICS. Threadsafety will result in hot-pooled applications executing under an H8 open TCB, and any CICS requests issued by the applications that are threadsafe also being executed under that H8 TCB. CICS will switch to the QR TCB to process any non-threadsafe EXEC CICS requests, then switch back to the H8 environment once more.

An important consideration when executing a Java program object in a hot-pooling environment is the way in which static storage is managed and used by the program. Java static storage is initialized at class load time. Since the hot-pooled enclave may be reused a number of times by different program invocations, values in changeable class data fields are not guaranteed to be their 'initial' values when a hot-pooled Java program object makes reference to them. The requirement is for a Java program object to explicitly reinitialize its static storage if it does have a dependency upon the contents of modifiable class data fields.

A good guideline for Java programming in such a hot-pooled environment is to define class fields with the Java keywords of 'private' and 'final'. Private fields are accessible only from within their own class; final fields cannot be subclassed.

Subject to the limiting factor as specified by MAXOPENTCBS, varying numbers of hot-pooled enclaves can be used under H8 open TCBs. However, it should be noted that each active enclave will utilize system resources. For example, the default LE run-time options as supplied with hot-pooling support will result in each enclave requiring 20KB of below-the-line storage. Modification of the default LE run-time options is discussed later in this article.

The number of open TCBs that can be utilized in a particular CICS system will be governed in part by the amount of virtual storage that is available below the 16MB line. As noted above, the default LE run-time options for the H8 open TCB enclaves result in 20KB of below-the-line virtual storage being required per TCB. The amount of virtual storage available to these TCBs can be increased if the amount of below-the-line DSA that is used by CICS is enlarged. The *CICS Performance Guide* provides further guidance on CICS DSA usage and management.

HOT-POOLING LIFETIME CONSIDERATIONS

Hot-pooled programs execute within reusable, preinitialized LE enclaves under H8 open TCBs until CICS terminates the enclave or TCB for some reason. The 'normal' reason for an enclave termination is when the LE heap storage size has grown because of enclave reuse, up to a heap size limiting value. This value is predefined at the PIPI preinitialization phase of the LE enclave. When this threshold is reached for an enclave, CICS terminates the environment upon program completion. (Note that heap storage size increases in this manner because Java garbage collection is set off for hot-pooled Java program objects. Enclave termination and creation avoids the need for the pathlength-expense of garbage collection.) The heap size value is tuneable, via the LE RUNOPTS options used during the creation of an enclave. Part of the hot-pooling deployment includes a new CICS User Replaceable Module (URM) called DFHAPH8O. DFHAPH8O is loaded during PIPI preinitialization for an LE enclave. It allows the default LE run-time options to be modified at this point. The version of DFHAPH8O as supplied with CICS sets some of these options, and the comments within the module document the options in further detail.

In addition to DFHAPH8O, hot-pooling support also provides another URM for customers to utilize. This is called DFHJHPAT. It is optional, and may be used for user-defined requirements and purposes (eg for tracing events). It must be written in C. If present on the CICS system, it is called before each hot-pooled program is invoked.

OTE H8 TCBs are terminated if CICS encounters a Short-On-Storage (SOS) situation, if a SET PROGRAM command to modify the program attributes or version is issued (eg PHASEIN), or if the program that is executing under the H8 TCB abends for some reason.

SERVICE CONSIDERATIONS FOR HOT-POOLING

Support for Java hot-pooling within CICS TS 1.3 has been implemented by a series of development enhancements to the product's various FMIDs, shipped via the APAR service process as PTFs.

CICS APAR PQ31328 / PTF UQ44003 ships support to the base CICS TS 1.3 FMID to provide the changes necessary to implement and deploy hot-pooling support within CICS TS 1.3. This APAR has a number of prerequisite PTF fixes from earlier APARs that SMP/E requires to be applied before applying UQ44003.

Of the prerequisite service required for PQ31328, the following APARs/PTFs warrant documenting here:

APAR PQ33485/PTF UQ39405

APAR PQ33485 ships a new module (DFH3QSS), required for hot-pooling support. DFH3QSS provides the means of determining whether its caller is running in a CICS environment or not, and whether the CICS API may be used.

APAR PQ31642/PTFs UQ38953, UQ38967, and UQ38968

APAR PQ31642 ships changes to the Java components of CICS TS 1.3 that provide support for HPJ-compiled Java application program objects executing in a hot-pooling environment. Such hot-pooled programs can reuse their execution environment within a preinitialized persistent enclave, and so require the CICS supplied Java classes to be statically reusable.

APAR PQ34434/PTF UQ39652

APAR PQ34434 ships changes to various modules within CICS that issue MVS getmain and freemain requests. The subpool specified upon the requests needs to be non TCB-related to make the storage viable in a Java hot-pooling environment.

The above three APARs, although prerequisite to the application of hot-pooling support, are themselves self-contained and independent of hot-pooling function.

CROSS-PRODUCT DEPENDENCIES

VisualAge for Java ET/390 PTFs UQ90022 and UQ90023 are the functional changes to enable hot-pooling support in that product. You should note that it is possible to run with this ET/390 hot-pooling enablement service applied, and yet not have the PTF service from PQ31642 applied to CICS TS 1.3. HPJ-compiled applications (of which parts of CICS TS 1.3 may be considered as a large example) do not require recompiling against the ET/390 enabling PTF service unless the hot-pooling function is intended to be used. However, the converse to this is not true. Having applied the PTF service from PQ31642, you should also have VisualAge for Java ET/390 PTFs UQ90022 and UQ90023 applied in order to provide the run-time support for such hot-pooling-enabled compiled Java code.

In addition to this, VisualAge for Java ET/390 APAR PQ34637 (PTF UQ40998) and APAR PQ34639 (UQ41004) should also be applied and Java applications compiled at this level of HPJ, if hot-pooling is to be used.

The relevant PTF(s) from the LE C library APAR PQ27831 should also be applied.

On a more general point, customers wishing to exploit the Java support in CICS TS 1.3 need to follow the instructions for installation and requirements for various cross-product dependencies that are documented in the CICS manuals. In particular, the *CICS Program Directory* (GI10-2506-01) and *CICS Installation Guide* (GC33-1681-03) give guidance on aspects of CICS' Java support installation and run-time requirements.

DOCUMENTATION CHANGES FOR HOT-POOLING

The CICS TS 1.3 manuals have been updated to document Java hot-pooling support. The various changes are summarized below:

- *The CICS Release Guide* (GC34-5352-00) documents the background to hot-pooling support.
- *The CICS Application Programming Guide* (SC33-1687-02) now describes the requirements for using hot-pooling, and what is needed to allocate, define, and manage a hot-pooled environment for applications.
- *The CICS Application Programming Reference* (SC33-1688-02) has been amended to reflect the enhancements to the CICS Application Programming Interface (API) for hot-pooling support.
- *The CICS Resource Definition Guide* (SC33-1684-02) now documents the additions to the program Resource Definition Online (RDO) attributes to support hot-pooled applications.
- *The CICS System Programming Reference* (SC33-1689-02) documents the additions to the CICS System Programming Interface (SPI) to provide support for hot-pooled applications via the EXEC CICS INQUIRE, SET, and CREATE commands.
- *The CICS Supplied Transactions* manual (SC33-1686-02) documents changes to the CEMT INQUIRE and SET PROGRAM commands for hot-pooling support.
- *The CICS Customization Guide* (SC33-1683-02) describes the two new CICS User Replaceable Modules (URMs) supplied for hot-pooling support. These are DFHAPH8O and DFHJHPAT. It also describes changes to the CICS Exit Programming Interface (XPI) for the INQUIRE_PROGRAM and SET_PROGRAM calls.
- *The CICS Messages and Codes* manual (GC33-1694-02) documents the new CICS messages and abend codes provided for hot-pooling support. These are DFHAP1219 through to DFHAP1225, and AJHA through to AJHF, AJH0 through AJH5 and AJH8 to AJH9, respectively.
- *The CICS Data Areas and Supplementary Data Areas* manuals (LY33-6090-02 and LY33-6089-02 respectively) describe changes to CICS control blocks for hot-pooling support.

- *The CICS Trace Entries* manual (SC34-5446-00) details the new trace entries that can be issued during hot-pooling activity within CICS.
- *The CICS Diagnosis Reference* (LY33-6088-02) describes changes to the CICS components that support hot-pooling.
- *The CICS Performance Guid*e (SC33-1699-03) details the performance issues relating to hot-pooling usage.

For further information on the CICS documentation changes relating to the introduction of hot-pooling support, please refer to the *Transaction Processing and Data (TP&D) Collection Kit*, order number SK2T-0730. The edition containing the updates for hot-pooling (amongst other enhancements) was issued in March 2000, and has the suffix of -25. It has an identifier of SK2T-0730-25. This edition replaces the previous -24 suffixed one (the December 1999 edition).

For further information on LE PIPI services, please refer to the *OS/390 Language Environment Programming Guide* (SC28-1939).

SUMMARY

I hope that this article has helped explain the background to CICS Java hot-pooling support within CICS TS 1.3, and the potential that exploiting this new facility within CICS could bring to Java applications.

CICS is a registered trademark of International Business Machines Corporation. Java is a registered trademark of Sun Microsystems Inc.

Readers wishing to discuss the material in this article further are welcome to contact me via e-mail at andy_wright@uk.ibm.com.

Andy Wright
CICS Change Team Programmer
IBM (UK)

Verification of VSAM forward recovery

We have a critical CICS/VSAM application that requires forward recovery, both as back-up and during disaster recovery. We were concerned at the scope for error in this process, particularly selection of the correct logs, and particularly in a DR situation when the most recent log archives would not be catalogued in the MVS catalog recovered at the DR site.

In order to provide some comfort factor after a recovery, we wrote a simple application which performs VSAM updates that can be verified easily after forward recovery. The idea is that if these updates are OK, then our real data will be OK as well.

All of the files requiring forward recovery are written to the same journal, so the same log concatenation is used for all recoveries.

The programs also do a timed switch of the CICS forward recovery journal. In our case this forces an archive to an offsite ATL.

HOW IT WORKS

A file, CICSFR*xx*, is allocated to a VSAM KSDS (*xx* is the CICS journal number used for forward recovery logging).

A transaction is scheduled every five minutes, which writes a record to the log. This record contains:

- A sequence number
- A timestamp.

Another program verifies the file after forward recovery. This program does the following:

- Verifies there are no gaps in the sequence numbers.
- Reports the latest timestamp found (approximate forward recovery time).

If any sequence gaps are found, the missing time range is reported.

INITIALIZING THE FILE

The file should be defined as a KSDS with an LRECL of 80 and keylength of 8. An initial record with a keyvalue of C'00000000' should be REPROd in.

LCIFR1

This is a PLT program that starts the forward recovery monitor. This program can also be run manually by typing the related transaction code. This will reinitialize the monitor, cancelling any pending reqids.

```
LCIFR1   AMODE  31
LCIFR1   RMODE  ANY
 TITLE 'LCIFR1, CICS forward recovery monitor program'
*
* This program runs from PLT or transaction. It reinitializes the
* temporary storage used to track each user journal. The TS
* is name CICSFRxx - Forward recovery info
*         CICSJRxx - Journal switch info
*
*
LCIFR1   DFHEIENT CODEREG=(3),DATAREG=(4),EIBREG=(8)
         REGS
         CLC   EIBTASKN+1(3),=C'III'   PLT ?
         BNE   INIT                    NO
         EXEC CICS START                                               X
               TRANSID('FR1')                                          X
               INTERVAL(000100)
         B     EXIT
INIT     EQU   *                       LOOP THROUGH FILES FOR OURS
         EXEC  CICS WRITE OPERATOR                                     X
               TEXT(MSG1)                                              X
               TEXTLENGTH(=A(MSG1L))
         EXEC  CICS INQUIRE FILE START
*
         EXEC  CICS HANDLE CONDITION END(FILEEND)
*
FILESRCH EQU   *
*
         EXEC  CICS INQUIRE FILE(FILENAME) NEXT
         CLC   FILENAME(6),=C'CICSFR' ONE OF OURS ?
         BH    FILEEND                NO PAST ALL OUR FILES
         BL    FILESRCH               NO
         MVC   QUENAME2(6),=C'CICSJR'
         MVC   QUENAME2+6(2),FILENAME+6  REMEMBER SUFFIX
*
*  READ TSQ1 IF IT EXISTS. IF FOUND WE WILL CANCEL PENDING REQID
*  THIS ALLOWS FOR SYSTEM RESTART OTHER THAN FROM PLT
*
         EXEC CICS HANDLE CONDITION                                    X
               QIDERR(CANCEL1)
         MVC   ITEM,=H'1'            USE ITEM 1
         EXEC CICS READQ  TS                                           X
               QUEUE(QUENAME1)                                         X
               INTO(QUE1AREA)                                          X
               ITEM(ITEM)                                              X
               LENGTH(QUE1LNH)
*
*  DELETE TEMP STORAGE BEFORE CREATING IT, AS IT WILL EXIST UNLESS
*  WE ARE CALLED FROM PLT
*
         EXEC CICS DELETEQ TS                                          X
         QUEUE(QUENAME1)
*
*
*  ISSUE CANCELS FOR PENDING REQIDS, IN CASE THIS PROGRAM IS NOT BEING
*  RUN FROM PLT
*
         EXEC CICS HANDLE CONDITION                                    X
               NOTFND(CANCEL1)
         EXEC  CICS CANCEL                                             X
               REQID(QUE1REQ)
*
*
CANCEL1  EQU   *
*
*  READ TSQ2 IF IT EXISTS. IF FOUND WE WILL CANCEL PENDING REQID
*  THIS ALLOWS FOR SYSTEM RESTART OTHER THAN FROM PLT
*
         EXEC CICS HANDLE CONDITION                                    X
               QIDERR(CANCEL2)
         MVC   ITEM,=H'1'            USE ITEM 1
         EXEC CICS READQ  TS                                           X
               QUEUE(QUENAME2)                                         X
               INTO(QUE2AREA)                                          X
               ITEM(ITEM)                                              X
               LENGTH(QUE2LNH)
*
         EXEC CICS DELETEQ TS                                          X
         QUEUE(QUENAME2)
*
         EXEC CICS HANDLE CONDITION                                    X
               NOTFND(CANCEL2)
         EXEC  CICS CANCEL                                             X
               REQID(QUE2REQ)
*
CANCEL2  EQU   *
```

```
         EXEC CICS HANDLE CONDITION                                    X
               NOTFND() QIDERR()
*
*
*
         EXEC  CICS ASKTIME                                            X
               ABSTIME(QUE1ABST)
         EXEC CICS FORMATTIME                                          X
              ABSTIME(QUE1ABST)                                        X
              TIMESEP(':')                                             X
              DATESEP('/')                                             X
              DDMMYY(QUE1DATE)                                         X
              TIME(QUE1TIME)
*
*
         EXEC CICS START                                               X
               TRANSID('FR2')                                          X
               FROM(FILENAME)                                          X
               INTERVAL(001500)
         MVC   QUE1REQ,EIBREQID      SAVE REQID
*
         MVC   ITEM,=H'1'            USE ITEM 1
         EXEC CICS WRITEQ TS                                           X
         QUEUE(QUENAME1)                                               X
         FROM(QUE1AREA)                                                X
         ITEM(ITEM)                                                    X
         LENGTH(QUE1LNH)                                               X
         MAIN
*
*   Now issue MSG Journal monitoring in place for this journal
*
*
         MVC   MSG3WS(L'MSG3),MSG3    MOVE CONSTANTS TO WS
         LA    R2,MSG3WS              ADDRESS WS
         USING MSG3,R2
         MVC   MSG3JNUM,FILENAME+6    COMPLETE VARIABLE FIELD
         DROP  R2
         EXEC  CICS WRITE OPERATOR                                     X
               TEXT(MSG3WS)                                            X
               TEXTLENGTH(=A(MSG3L))
*
*   Now read file to check the highest key so far. This is saved in
*   TSQ2
*
         MVC   RECKEY,=C'00000000'
         EXEC  CICS STARTBR                                            X
               FILE(FILENAME)                                          X
               RIDFLD(RECKEY)                                          X
               GTEQ
*
```

```
         EXEC  CICS HANDLE CONDITION ENDFILE(KEYEND)
*
KEYLOOP  EQU   *                      READ TO THE HIGHEST KEY
         EXEC  CICS READNEXT                                           X
               FILE(FILENAME)                                          X
               RIDFLD(RECKEY)                                          X
               INTO(RECORD)
         B     KEYLOOP                LOOP TO END OF FILE
KEYEND   EQU   *
         MVC   QUE2KEY,RECKEY         SAVE HIGHEST KEY IN THIS FILE
         EXEC  CICS ENDBR                                              X
               FILE(FILENAME)
*
*  Send MSG to Operate File Update beginning
*
*
*
         MVC   MSG2WS(L'MSG2),MSG2    MOVE CONSTANTS TO WS
         LA    R2,MSG2WS              ADDRESS WS
         USING MSG2,R2
         MVC   MSG2FILE,FILENAME      COMPLETE VARIABLE FIELD
         DROP  R2
         EXEC  CICS WRITE OPERATOR                                     X
               TEXT(MSG2WS)                                            X
               TEXTLENGTH(=A(MSG2L))
*
*   Issue start for FR3
*   This transaction will do regular updates to the file
*   CICSFRxx
*
         EXEC CICS START                                               X
               TRANSID('FR3')                                          X
               FROM(FILENAME)                                          X
               INTERVAL(ØØØ5ØØ)
         MVC   QUE2REQ,EIBREQID      SAVE REQID
*
*
         MVC   ITEM,=H'1'            USE ITEM 1
         EXEC CICS WRITEQ TS                                           X
               QUEUE(QUENAME2)                                         X
               FROM(QUE2AREA)                                          X
               ITEM(ITEM)                                              X
               LENGTH(QUE2LNH)                                         X
               MAIN
*
         B     FILESRCH               PROCESS NEXT FILE
FILEEND  EQU   *
         EXEC  CICS INQUIRE FILE END
         EXEC  CICS WRITE OPERATOR                                     X
               TEXT(MSG4)                                              X
               TEXTLENGTH(=A(MSG4L))
EXIT     EXEC  CICS RETURN
MSG1     DC    C'LCIFRØ1I Forward Recovery Monitor initializing'
MSG1L    EQU   *-MSG1
MSG2     DC    C'LCIFRØ2I File update beginning for '
MSG2FILE DS    CL8
MSG2L    EQU   *-MSG2
MSG3     DC    C'LCIFRØ3I Journal monitoring beginning for DFHJ'
MSG3JNUM DS    CL2
MSG3L    EQU   *-MSG3
MSG4     DC    C'LCIFRØ4I Forward Recovery Monitor initialized'
MSG4L    EQU   *-MSG4
         COPY  LCIFRCPY                COMMON COPY CODE
MSG2WS   DS    (MSG2L)C
MSG3WS   DS    (MSG3L)C
         END
```

LCIFR2

This is the file update program. It is scheduled every five minutes.

```
LCIFR2   AMODE  31
LCIFR2   RMODE  ANY
 TITLE 'LCIFR2, CICS forward recovery Journal monitor'
*
* This program is started with a passed filename, CICSFRxx.
* A temporary storage queue, CICSFRxx, contains the last time the
* journal xx was switched.
* Logic of this program:
*
* INQUIRE JOURNAL xx
* If (ARCHIVE STILL OUTSTANDING)
*   Issue warning
*   Restart in 5 minutes
*   Exit
* Else
*   If (last switchtime >= 15 minutes)
*      Request journal switch
*      Restart in 5 minutes (To check archived)
*   Else
*      Reschedule at last switch time + 15 minutes
*   Endif
* Endif
LCIFR2   DFHEIENT CODEREG=(3),DATAREG=(4),EIBREG=(8)
         REGS
INIT     EQU   *                       LOOP THROUGH FILES FOR OURS
         EXEC CICS ASKTIME                                      X
               ABSTIME(TIMENOW)
*        EXEC CICS RETRIEVE                                     X
```

```
               INTO(FILENAME)
         MVC   FILENAME,=C'CICSFRØ2'
*
         MVC   ITEM,=H'1'              USE ITEM 1
         EXEC CICS READQ TS                                            X
               QUEUE(QUENAME1)                                         X
               INTO(QUE1AREA)                                          X
               LENGTH(QUE1LNH)                                         X
               ITEM(ITEM)
         PACK  PACKWORK,FILENAME+6(2)  Get file suffix
         CVB   R1,PACKWORK
         STH   R1,JNUM                 Journal num as binary halfword
         EXEC CICS INQUIRE                                             X
               JOURNALNUM(JNUM)                                        X
               DISKASTATUS(DISKA)                                      X
               DISKBSTATUS(DISKB)
         CLC   DISKA,DFHVALUE(NOTREADY)
         BE    ARCHWARN
         CLC   DISKB,DFHVALUE(NOTREADY)
         BNE   TESTSWIT
ARCHWARN EQU   *
*
         MVC   MSG1WS(L'MSG1),MSG1    MOVE CONSTANTS TO WS
         LA    R2,MSG1WS              ADDRESS WS
         USING MSG1,R2
         MVC   MSG1JNUM,FILENAME+6    COMPLETE VARIABLE FIELD
         DROP  R2
         EXEC  CICS WRITE OPERATOR                                     X
               TEXT(MSG1WS)                                            X
               TEXTLENGTH(=A(MSG1L))
         ZAP   TIMEWORK,TIMENOW      NOW
         AP    TIMEWORK,TIME5        + 5 minutes
         B     RESCHED
TESTSWIT EQU   *
         ZAP   TIMEWORK,QUE1ABST       TIME LAST SWITCH
         AP    TIMEWORK,TIME15         + 15 Minutes
         CP    TIMEWORK,TIMENOW
         BH    RESCHED                NOT 15 MINUTES SINCE SWITCH
         MVC   MSG2WS(L'MSG2),MSG2    MOVE CONSTANTS TO WS
         LA    R2,MSG2WS              ADDRESS WS
         USING MSG2,R2
         MVC   MSG2JNUM,FILENAME+6    COMPLETE VARIABLE FIELD
         DROP  R2
         EXEC  CICS WRITE OPERATOR                                     X
               TEXT(MSG2WS)                                            X
               TEXTLENGTH(=A(MSG2L))
         EXEC CICS SET                                                 X
               JOURNALNUM(JNUM)                                        X
               ADVANCE
         ZAP   QUE1ABST,TIMENOW        UPDATE SWITCH TIME
         EXEC CICS FORMATTIME                                          X
              ABSTIME(QUE1ABST)                                        X
              TIMESEP(':')                                             X
              DATESEP('/')                                             X
              DDMMYY(QUE1DATE)                                         X
              TIME(QUE1TIME)
         ZAP  TIMEWORK,TIMENOW
         AP   TIMEWORK,TIME5           CHECK ARCHIVE IN 5 MINUTES
RESCHED  EQU   *                       RESCHEDULE AT TIMEWORK TIME
*
*  The following code converts the abstime in timework to
*  a pl4 HHMMSS time
*
         EXEC CICS FORMATTIME                                          X
              ABSTIME(TIMEWORK)                                        X
              TIMESEP(':')                                             X
              TIME(TIMEHMS)
         PACK PACKWORK,TIMEHH
         MP   PACKWORK,=P'1ØØØØ'          HHØØØØ
         ZAP  PACK4,PACKWORK
         PACK PACKWORK,TIMEMM
         MP   PACKWORK,=P'1ØØ'            MMØØ
         AP   PACK4,PACKWORK              HHMMØØ
         PACK PACKWORK,TIMESS
         AP   PACK4,PACKWORK              HHMMSS
*
*  Start next FR2 at appropriate interval
*
*
*
         EXEC CICS START                                               X
               TRANSID('FR2')                                          X
               FROM(FILENAME)                                          X
               TIME(PACK4)
         MVC  QUE1REQ,EIBREQID         SAVE REQID IN TS
*
         EXEC CICS WRITEQ TS                                           X
               QUEUE(QUENAME1)                                         X
               FROM(QUE1AREA)                                          X
               LENGTH(QUE1LNH)                                         X
               ITEM(ITEM)                                              X
               REWRITE MAIN
*
*
EXIT     EXEC  CICS RETURN
MSG1     dc    c'LCIFR21W CICS archive still outstanding for DFHJ'
MSG1JNUM DS    CL2
MSG1L    EQU   *-MSG1
MSG2     dc    c'LCIFR22I Switching journal DFHJ'
MSG2JNUM DS    CL2
```

```
MSG2L    EQU   *-MSG2
TIME15   DC    PL8'9ØØØØØ'             15 MINUTES
TIME5    DC    PL8'3ØØØØØ'              5 MINUTES
         COPY LCIFRCPY
MSG1WS   DS    (MSG1L)C
MSG2WS   DS    (MSG2L)C
DISKA    DS    F
DISKB    DS    F
TIMENOW  DS    PL8
TIMEWORK DS    PL8
PACK4    DS    PL4
TIMEHMS  DS    CL8
         ORG   TIMEHMS
TIMEHH   DS    CL2
         DS    CL1
TIMEMM   DS    CL2
         DS    CL1
TIMESS   DS    CL2
         ORG
JNUM     DS    H
         END
```

LCIFR3

This is the journal switch program. Journals are switched every 15 minutes to provide offsite recovery. We also ‘stack’ CICS and DB2 archives on the same cartridges to prevent excessive cart requirements.

```
LCIFR3   AMODE  31
LCIFR3   RMODE  ANY
 TITLE 'LCIFR3, CICS forward recovery monitor  program'
*
* This program is started with a passed filename, CICSFRxx.
* A temporary storage queue CICSJRxx contains the highest key in the
* file. We add 1 to the number then write a new record to the file.
* We then start tran FR3 with an interval of 5 minutes.
*
*
LCIFR3   DFHEIENT CODEREG=(3),DATAREG=(4),EIBREG=(8)
         REGS
INIT     EQU   *                        LOOP THROUGH FILES FOR OURS
         EXEC CICS RETRIEVE                                            X
               INTO(FILENAME)
*        MVC   FILENAME,=C'CICSFRØ2'
         MVC  QUENAME2(6),=C'CICSJR'
         MVC  QUENAME2+6(2),FILENAME+6  BUILD TSQNAME FROM FILENAME
*
         MVC   ITEM,=H'1'               USE ITEM 1
         EXEC CICS READQ TS                                            X
               QUEUE(QUENAME2)                                         X
               INTO(QUE2AREA)                                          X
               LENGTH(QUE2LNH)                                         X
               ITEM(ITEM)
         PACK  PACKWORK,QUE2KEY        GET HIGHEST KEY
         AP    PACKWORK,=P'1'          ADD 1
         UNPK  QUE2KEY,PACKWORK
         OI    QUE2KEY+7,X'FØ'
         MVC   RECKEY,QUE2KEY
*
*
         EXEC  CICS ASKTIME                                            X
               ABSTIME(QUE1ABST)
         EXEC CICS FORMATTIME                                          X
              ABSTIME(QUE1ABST)                                        X
              TIMESEP(':')                                             X
              DATESEP('/')                                             X
              DDMMYY(RECDATE)                                          X
              TIME(RECTIME)
*
*
         EXEC CICS WRITE                                               X
               FILE(FILENAME)                                          X
               FROM(RECORD)                                            X
               RIDFLD(RECKEY)
*
*
*
         EXEC CICS START                                               X
               TRANSID('FR3')                                          X
               INTERVAL(ØØØ5ØØ)                                        X
               FROM(FILENAME)
         MVC   QUE2REQ,EIBREQID        SAVE REQID
         MVC   ITEM,=H'1'              USE ITEM 1
*
*
         EXEC CICS WRITEQ TS                                           X
               QUEUE(QUENAME2)                                         X
               FROM(QUE2AREA)                                          X
               LENGTH(QUE2LNH)                                         X
               ITEM(ITEM)                                              X
               REWRITE MAIN
*
*
EXIT     EXEC  CICS RETURN
         COPY LCIFRCPY
         END
```

LCIFR4

This program verifies a recovered file. It is started with a parameter indicating the journal number, eg ‘FR4 02’ verifies file CICSFR02.

```
LCIFR4   AMODE  31
LCIFR4   RMODE  ANY
 TITLE 'LCIFR4, CICS forward recovery check  program'
*
* This program checks the recovery file CICSFRxx for journal
* problems, etc.
* INPUT : trancode followed by journal num eg 'FR4 2'
*         CICSJRxx - Journal switch info
*
*
LCIFR4   DFHEIENT CODEREG=(3),DATAREG=(4),EIBREG=(8)
         REGS
INIT     EQU   *
         EXEC CICS HANDLE CONDITION                                    X
              ERROR(RECERROR)
         EXEC CICS RECEIVE                                             X
              INTO(INPMSG)                                             X
              LENGTH(INPMSGL)                                          X
              MAXLENGTH(=H'6')
         EXEC CICS HANDLE CONDITION                                    X
              ERROR()
         TRT  INPJNUM,NUMTAB              TEST NUMERICS
         BNZ  RECERROR                    NON-NUMERIC JNUM
         MVC  FILENAME(6),=C'CICSFR'      BUILD FILE NAME
         MVC  FILENAME+6(2),INPJNUM
*
*   Now read file to check the highest key so far. This is saved in
*   TSQ2
*
         MVC   RECKEY,=C'ØØØØØØØØ'
         MVI   FR4FLAG,FR4OK               Assume no errors
         EXEC  CICS STARTBR                                            X
               FILE(FILENAME)                                          X
               RIDFLD(RECKEY)                                          X
               GTEQ
*
         EXEC  CICS HANDLE CONDITION ENDFILE(KEYEND)
*
READ1ST  EQU   *                      READ TO THE HIGHEST KEY
         EXEC  CICS READNEXT                                           X
               FILE(FILENAME)                                          X
               RIDFLD(RECKEY)                                          X
               INTO(RECORD)
*
KEYLOOP  EQU   *                      READ TO THE HIGHEST KEY
         MVC   SAVREC,RECORD          SAVE PREVIOUS RECORD
         EXEC  CICS READNEXT                                           X
               FILE(FILENAME)                                          X
               RIDFLD(RECKEY)                                          X
               INTO(RECORD)
         PACK  PACKWORK,SAVKEY        SAVE OLD KEY
         AP    PACKWORK,=P'1'         ADD 1
         PACK  PACKWOR2,RECKEY        CURRENT KEY
         CP    PACKWOR2,PACKWORK      COMPARE KEYS
         BE    KEYLOOP                NO GAP
*
*  Send MSG to Operator key range incorrect
*
*
*
         MVC   MSG2WS(MSG2L),MSG2     MOVE CONSTANTS TO WS
         LA    R2,MSG2WS              ADDRESS WS
         USING MSG2,R2
         MVC   MSG2KEY1,SAVKEY        COMPLETE VARIABLE FIELD
         MVC   MSG2KEY2,RECKEY        COMPLETE VARIABLE FIELD
         DROP  R2
         EXEC  CICS WRITE OPERATOR                                     X
               TEXT(MSG2WS)                                            X
               TEXTLENGTH(=A(MSG2L))
*
*
         MVC   MSG3WS(MSG3L),MSG3     MOVE CONSTANTS TO WS
         LA    R2,MSG3WS              ADDRESS WS
         USING MSG3,R2
         MVC   MSG3DAT1,SAVDATE       COMPLETE VARIABLE FIELD
         MVC   MSG3TIM1,SAVTIME       COMPLETE VARIABLE FIELD
         MVC   MSG3DAT2,RECDATE       COMPLETE VARIABLE FIELD
         MVC   MSG3TIM2,RECTIME       COMPLETE VARIABLE FIELD
         DROP  R2
         EXEC  CICS WRITE OPERATOR                                     X
               TEXT(MSG3WS)                                            X
               TEXTLENGTH(=A(MSG3L))
         MVI   FR4FLAG,FR4BAD         Not errors exist
*
         B     KEYLOOP                Process next key
KEYEND   EQU   *
         EXEC  CICS ENDBR                                              X
               FILE(FILENAME)
         CLI   FR4FLAG,FR4BAD         any errors
         BE    DOMSG5
*
*  SEND SUCCESSFUL MESSAGE TO TERMINAL
*
         MVC   MSG4WS(MSG4L),MSG4    MOVE CONSTANTS TO WS
         LA    R2,MSG4WS              ADDRESS WS
         USING MSG4,R2
         MVC   MSG4DATE,RECDATE       COMPLETE VARIABLE FIELD
         MVC   MSG4TIME,RECTIME       COMPLETE VARIABLE FIELD
         MVC   MSG4FILE,FILENAME      COMPLETE VARIABLE FIELD
         DROP  R2
         EXEC  CICS SEND                                               X
               FROM(MSG4WS)                                            X
               ERASE                                                   X
               LENGTH(=AL2(MSG4L))
         MVC   MSG4WS(MSG4L),MSG4     MOVE CONSTANTS TO WS
         LA    R2,MSG4WS              ADDRESS WS
         USING MSG4,R2
         MVC   MSG4DATE,RECDATE       COMPLETE VARIABLE FIELD
         MVC   MSG4TIME,RECTIME       COMPLETE VARIABLE FIELD
         MVC   MSG4FILE,FILENAME      COMPLETE VARIABLE FIELD
         DROP  R2
         EXEC  CICS SEND                                               X
               FROM(MSG4WS)                                            X
               LENGTH(=AL2(MSG4L))
         B     EXIT
*
RECERROR EQU   *
         EXEC  CICS SEND                                               X
               FROM(MSG1)                                              X
               ERASE                                                   X
               LENGTH(=AL2(MSG1L))
         B     EXIT
DOMSG5   EQU   *
*
*  SEND FAILURE MESSAGE TO TERMINAL
*
         EXEC  CICS SEND                                               X
               FROM(MSG5)                                              X
               ERASE                                                   X
               LENGTH(=AL2(MSG5L))
         B     EXIT

EXIT     EXEC  CICS RETURN
NUMTAB   DC    256X'FF'
         ORG   NUMTAB+X'FØ'
         DC    1ØX'ØØ'
         ORG
MSG1     DC    C'LCIFR41I Invalid input: Type "FR4 xx"'
MSG1L    EQU   *-MSG1
MSG2     DC    C'LCIFR42E Missing keys detected between '
MSG2KEY1 DS    CL8
         DC    C' and '
MSG2KEY2 DS    CL8
MSG2L    EQU   *-MSG2
MSG3     DC    C'LCIFR43I Time range is between '
MSG3DAT1 DS    CL8
         DC    C' '
MSG3TIM1 DS    CL8
```

```
         DC    C' and '
MSG3DAT2 DS    CL8
         DC    C' '
MSG3TIM2 DS    CL8
MSG3L    EQU   *-MSG3
MSG4     DC    C'LCIFR44I No errors detected on file '
MSG4FILE DS    CL8
         DC    (8Ø-*+MSG4)C' '
         DC    C'         Latest update time was '
MSG4DATE DS    CL8
         DC    C' '
MSG4TIME DS    CL8
MSG4L    EQU   *-MSG4
MSG5     DC    C'LCIFR45E Errors detected - see syslog'
MSG5L    EQU   *-MSG5
         COPY  LCIFRCPY                COMMON COPY CODE
INPMSGL  DS    H
INPMSG   DS    CL6
         ORG   INPMSG
         DS    CL4
INPJNUM  DS    CL2
         ORG
FR4FLAG  DS    C
FR4OK    EQU   Ø
FR4BAD   EQU   1
MSG2WS   DS    (MSG2L)C
MSG3WS   DS    (MSG3L)C
MSG4WS   DS    (MSG4L)C
         END
```

LCIFRCPY

This is the common copy code.

```
QUE1LNH  DC   AL2(QUE1LEN)
QUE2LNH  DC   AL2(QUE2LEN)
RECLNH   DC   AL2(L'RECORD)
         DFHEISTG
         DS   ØD
PACKWORK DS   PL8
PACKWOR2 DS   PL8
ITEM     DS   H
FILENAME DS   CL8
         ORG  FILENAME
QUENAME1 DS   CL8                     CICSFRxx
QUENAME2 DS   CL8                     CICSJRxx
RECORD   DS   CL8Ø
         ORG  RECORD
RECKEY   DS   CL8
         DS   C
```

```
RECDATE  DS   CL8
         DS   C
RECTIME  DS   CL8
         ORG
SAVREC   DS   CL8Ø
         ORG  SAVREC
SAVKEY   DS   CL8
         DS   C
SAVDATE  DS   CL8
         DS   C
SAVTIME  DS   CL8
         ORG
QUE1AREA DS   ØD                        TEMP QUEUE 1 - switch time
QUE1ABST DS   PL8                       ABSTIME SWITCH
QUE1DATE DS   CL8                       DATE SWITCH
QUE1TIME DS   CL8                       TIME SWITCH
QUE1REQ  DS   CL8                       REQID
QUE1LEN  EQU  *-QUE1AREA
QUE2AREA DS   ØD                        TEMP QUEUE 2 - file key
QUE2KEY  DS   CL8
QUE2REQ  DS   CL8                       REQID
QUE2LEN  EQU  *-QUE2AREA
```

Joe Owens
Senior Technician
Standard Life Assurance Company (UK)

Expanding COBOL COPYs before compilation

Copybooks are normally used in COBOL to hold pieces of code that can be shared among programs. Normally, copybooks are used for file descriptions, but can be used for other things as well. The COPY instruction is processed by the compiler.

There are occasions when a COBOL program needs to be pre-processed before entering the compiler. The most typical case is with EXEC CICS...END-EXEC instructions, which must be translated to COBOL calls before the source code enters the compiler.

The same is true for EXEC ADABAS, for instance. And the problem arises when there are copybooks containing EXEC...END-EXEC blocks. Since EXECs are pre-compiled and copybooks are expanded by the compiler, this means that EXECs will be passed directly to the compiler without being pre-processed (replaced by calls) and thus the compilation returns an error.

There are two possible ways to deal with this situation. One is to replace copybooks that contain EXECs by already pre-processed ones, ie where EXECs are already replaced by calls. This method has the disadvantage that if the pre-compiler changes, the pre-compiled copybooks will have to be recreated.

The second way is to expand all or part of the copybooks to a temporary file that will act as input for the pre-processor. This is exactly what this REXX program does. It expands copies, including nested ones, and, as an option, does not expand copies that contain the keyword SUPPRESS. For example, a COBOL line like COPY MYMEMBER SUPPRESS means that the copy expansion will not appear in the COBOL listing output. If you invoke COPYGET with the parameter SUPPRESS=YES, this means that copies containing the keyword SUPPRESS will not be pre-expanded to the temporary file. Without this option, all copies will be expanded.

Below is an example of the relevant part of a job to run COPYGET, in STEP01 (STEP02 runs the CICS pre-processor). The COPYLIBS DD contains the libraries where copybooks can be found, in search order. After execution, the SYSTSPRT output will contain a list of all copies: those that were expanded and from which library, those that were not found (and if this happens return code 4 is set, so your job can decide what to do, but processing continues), and those that were not expanded because the COPY instruction contained the SUPPRESS option and the ‘SUPPRESS=YES’ parameter was specified, as in the example job. If you don’t want this option, simply remove the parameter.

```
//XXXJOB JOB REGION=2Ø48K,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//STEPØ1 EXEC PGM=IKJEFTØ1
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSTSIN  DD  *
PROFILE NOPREFIX
EXEC 'PDS.FOR.EXECS(COPYGET)'  'SUPPRESS=YES'
/*
```

```
//INFILEØ  DD  DISP=SHR,DSN=my.cobol.program
//OUTFILE  DD  DISP=(NEW,PASS),DSN=&&TEMP1,
//             SPACE=(CYL,(5,5)),UNIT=SYSDA,
//             RECFM=FB,LRECL=8Ø
//COPYLIBS DD  *
pds.with.copybks.one
pds.with.copybks.two
pds.with.copybks.three
/*
//STEPØ2 EXEC PGM=DFHECP1$
//SYSIN    DD DISP=(OLD,DELETE),DSN=&&TEMP1
......
```

COPYGET SOURCE CODE

```
/* REXX TSO BATCH ==================================================*/
/*                                                                  */
/*  COPYGET - Expands COBOL copies into an output file.             */
/*  Copies can be nested. The following DDnames must be allocated: */
/*    INFILEØ - Primary COBOL input file.                           */
/*    OUTFILE - Output file with expanded copies.                   */
/*    COPYLIBS- File containing a list of PDS where copies reside. */
/*                                                                  */
/*  The optional argument 'SUPPRESS=YES' means that copies          */
/*  with the SUPPRESS keywork are not expanded to the output file. */
/*  Return codes: Ø - All OK.  4 - One or more copies not found.   */
/*==================================================================*/

arg option .
if  option = "SUPPRESS=YES" then suppress = 1
                            else suppress = Ø
do x = 1 to 1ØØ
   execio 1 diskr copylibs
   if rc <>Ø then leave x
   pull lib.x  .
end
lib.Ø = x-1

k = Ø
step_retcod = Ø
inf = infileØ
do alpha = Ø
   execio 1 diskr inf
   if rc <> Ø then do
      if k = Ø then leave alpha
      call deallocate
      iterate alpha
   end
   parse pull linha
   linha7 = substr(linha,7)
   if word(linha7,1)= "COPY" then do
      if suppress=1 & left(word(linha7,3),8)="SUPPRESS" then do
         say ">>>  Copy Suppressed: " word(linha7,2)
      end
      else do
         call allocate
         if result = 0 then linha = overlay("*",linha,7)
      end
   end
   queue linha
   execio 1 diskw outfile
end
exit step_retcod

/*==================================================================*/

allocate:
 retcod = 0
 membro = strip(word(linha7,2),,".")
 k = k+1
 inf = infile||k
 do j = 1 to lib.0
    nome = lib.j"("membro")"
    if sysdsn(nome) = "OK" then do
       "alloc da('"nome"') dd("inf") shr"
       leave j
    end
 end
 if j > lib.0 then do
    say "==== COPY NOT FOUND:  " membro
    k = k-1
    inf = infile||k
    retcod = 4
    step_retcod = 4
 end
 else do
    say ">>>  Copy expanded from " nome
 end
return retcod

deallocate:
 "execio 0 diskr" inf "(finis"
 "free dd("inf")"
 k = k-1
 inf = infile||k
return
```

Luis Paulo Ribeiro
Systems Engineer
Edinfor (Portugal)

CICS on the VSE/ESA platform: e-business support

INTRODUCTION

CICS e-business support enables end users with Web browsers or network computers to access CICS application programs and transactions using standard Internet protocols. This article provides a technical overview of the CICS technology choices available to enable this for CICS on the VSE/ESA platform.

As shown in Figure 1, there are two main architecture models for connecting from Web browsers or network computers to CICS. They are:

- The three-tier model, which has connectivity via a Web server running on an intermediate system. Tier 1 is the client Web browser, tier 2 is the Web server platform, and tier 3 is the S/390 server running CICS. IBM's Application Framework for e-business is based on the three-tier model and is implemented for CICS running on VSE/ESA via the CICS Transaction Gateway.
- The two-tier model, which has a direct connection to CICS without the need for an intermediate system. This is implemented within CICS Transaction Server for VSE/ESA via services known as CICS Web Support.

THE CICS TRANSACTION GATEWAY

Overview

The CICS Transaction Gateway provides a comprehensive set of Java-based Web server facilities for accessing CICS application programs and transactions from a Web browser in a three-tier model, and is a connector component of IBM's Application Framework for e-business. It runs on the same intermediate system as a Web server and communicates with all current CICS systems.

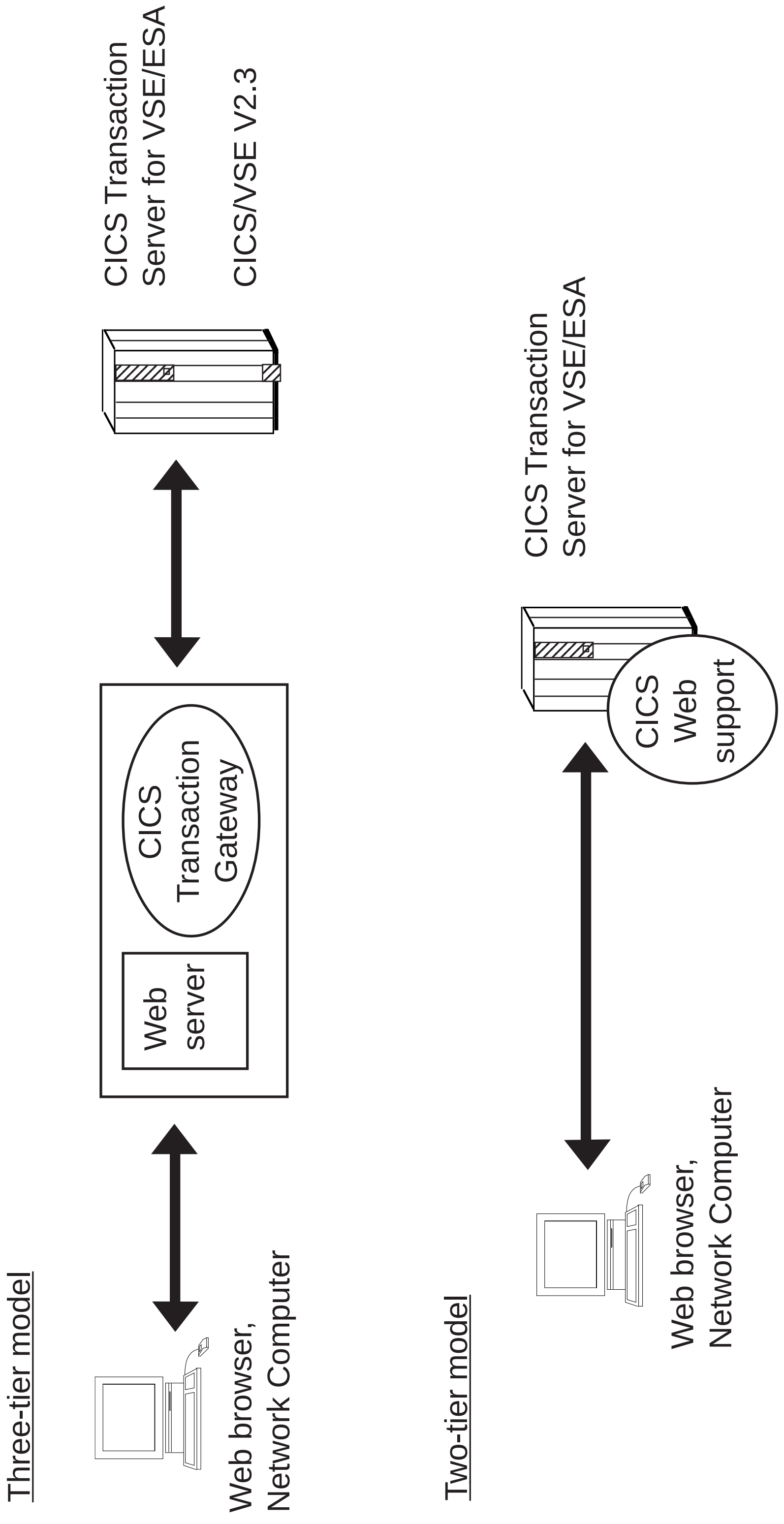
Three-tier model
Web browser, Network Computer
Web server
CICS Transaction Gateway
CICS Transaction Server for VSE/ESA
CICS/VSE V2.3
Two-tier model
Web browser, Network Computer
CICS Web support
CICS Transaction Server for VSE/ESA

Figure 1: Architecture models

Java classes and JavaBeans are supplied for use in any Java program environment – that is Java servlets (programs that run on a Web server), Java applets (programs that run on a Web browser), and Java applications that can run on any Java execution platform.

Java support

The Java support provided can be used in the following main ways:

- Writing servlets – the CICS Transaction Gateway supplies Java classes and JavaBeans that allows servlets to be written to handle browser requests requiring services from CICS. Servlets are programs that are loaded by the Web server and run in the Web server environment.
- Writing applets – the same Java classes and JavaBeans can be used to write an applet to access CICS in the same ways as a servlet, the difference being that an applet executes in a Web browser on the end user workstation after being downloaded from a Web server. It requires a Java-enabled browser on the workstation. Applets have some restrictions compared to Java servlets and Java applications – for example they may not access local resources such as local files.
- Writing applications – again, the same Java facilities are available to access CICS as for servlets and applets. The main difference is that Java applications do not run with a Web server or Web browser, but run in any other Java execution environment and have none of the limitations that an applet has. A Java application can run on the same system as the CICS Transaction Gateway, or remotely on another system or workstation.
- Turnkey access to 3270 transactions – the CICS Transaction Gateway supplies one pre-written servlet, called the Terminal Servlet. This provides turnkey Web browser access to CICS 3270 based transactions. The Terminal Servlet translates Web browser requests to CICS interactions on a one-for-one basis, so that the user sees the browser equivalent of the ‘green’ screens that would appear on a real 3270 terminal.
- Customized access to 3270 transactions – the CICS Transaction

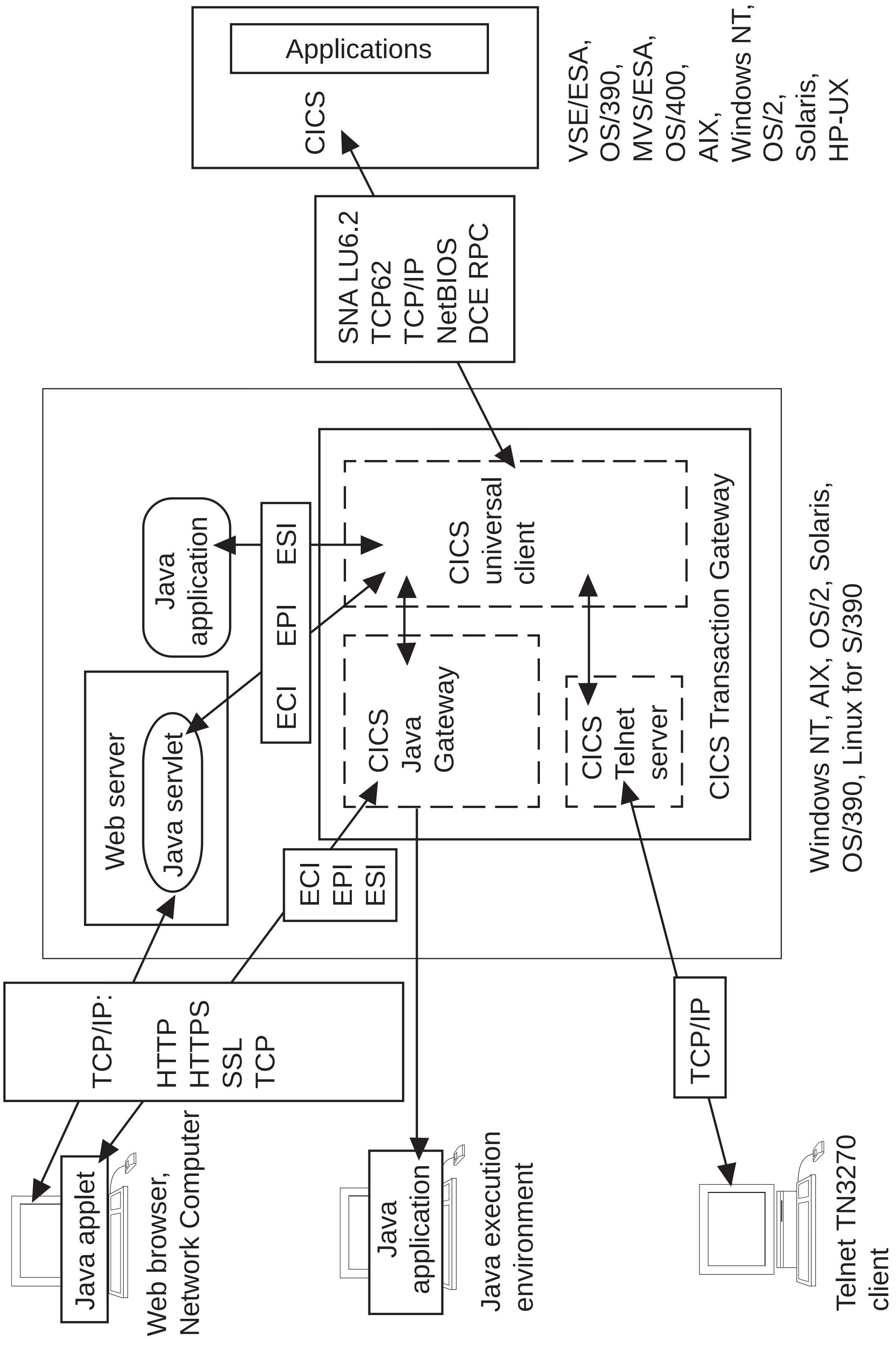

Figure 2: CICS TS structure and interface

Gateway also provides facilities for customizing (without programming) the default screen appearance on a Web browser produced by the Terminal Servlet. For example, it is possible to map specific screens to specific HTML pages and to insert screen output fields into HTML pages via variable substitution and HTML templates.

Major components

The CICS Transaction Gateway structure and interfaces are illustrated in Figure 2. The following are the major Gateway components:

- The Java Gateway application – on non-OS/390 platforms, it communicates with CICS applications through facilities provided by the CICS Universal Client component. On OS/390 systems, it communicates with CICS applications via the CICS External Communication Interface (EXCI).
- A CICS Universal Client that provides the basic interfaces and connectivity to host CICS systems, as well as terminal emulation function.
- A CICS Java class library that provides the Java APIs for communication between a user-written Java program and a CICS system.
- A set of EPI JavaBeans for creating Java front-ends to existing CICS 3270-based applications without the need for programming. JavaBeans are self-contained reusable software components that can be visually composed into applications, applets, or servlets via the use of visual application builders – for example via the IBM VisualAge for Java product.
- The Terminal Servlet that allows a Web browser to be used to access existing 3270-based CICS transactions.
- A set of Common Connector Framework (CCF) Java classes that are supplied in support of IBM's Common Connector Framework strategy. The objective of CCF is to provide consistent client application interaction with enterprise resources such as CICS or DB2 from any Java execution environment.

- A TN3270 server that provides support to allow multiple end-users using Telnet TN3270 clients to connect to CICS.

Application Programming Interfaces

The External Call Interface (ECI) enables a client Java application to call a CICS program and is equivalent to Distributed Program Link. It uses a COMMAREA interface, and to the target CICS application program it appears as if it is being called by another program in the same CICS system. Multiple CICS programs can be called, either in the same CICS system or in different systems, within the same invocation of the client.

The External Presentation Interface (EPI) enables a client Java application to emulate a 3270 terminal and so drive a CICS 3270-based application. Essentially the client application simulates a 3270 terminal, and there is no change to the CICS 3270 application. The basic interface is at the 3270 datastream level, but there are higher-level Java EPI classes and Java EPI beans that can be used to shield the programmer from this. The same client can be connected to multiple CICS systems or act as multiple 3270s to the same CICS system. Note that the EPI is not supported when the Gateway is running on OS/390.

The External Security Interface (ESI) enables a client application to invoke services provided by APPC Password Expiry Management (PEM) – for example to verify or change passwords. The ESI requires an External Security Manager on System/390 on which the host CICS system is running. This provides support for an APPC architected sign-on transaction that signs on users to a CICS host system and which can process requests for a password change.

Connectivity

The CICS Transaction Gateway supports connection to CICS systems using a variety of protocols. The only form of connectivity currently supported by CICS on the VSE/ESA platform is SNA LU6.2, but IBM’s direction is to also provide support for TCP/IP in the future.

From Windows NT, Windows 95/98, and OS/2, there is also support for TCP62 via the IBM Communications Server, which enables transport of SNA LU6.2 data over TCP/IP connections to CICS on MVS or OS/390.

The CICS Transaction Gateway also supports TCP/IP and DCE RPC connection to a CICS system running on AIX, HP-UX, Solaris, and Windows NT. This could also be used as an intermediate CICS system which can connect to CICS on S/390 using SNA. The NetBIOS protocol is also supported for connection to CICS for OS/2.

Network protocols supported for connectivity to Web browsers and other Java environments include TCP/IP Sockets, SSL, HTTP, and HTTP-S. Support for Secure Sockets Layer (SSL) enables data encryption and authentication for network security.

Supported environments

Access to CICS via the CICS Transaction Gateway is supported via any Java-enabled Web browser at the JDK 1.1 level or later, for example Netscape or Internet Explorer. The Terminal Servlet requires a Web server or servlet engine that provides support equivalent to the Java Servlet Development Kit (JSDK) Version 1.1 or later, for example as provided by the IBM WebSphere products.

The Gateway is supported on the following platforms:

- Windows NT, Windows 98/95 (development puposes only).
- OS/2, AIX, Solaris, OS/390.
- Linux for S/390 (availability planned for 12/00).

CICS systems supported:

- CICS Transaction Server for OS/390.
- CICS for MVSE/ESA Version 4.1.
- CICS Transaction Server for VSE/ESA.
- CICS/VSE Version 2.3.
- CICS Transaction Server for OS/2.
- TXSeries (now part of WebSphere Enterprise Edition), which provides CICS on AIX, Solaris, HP-UX, and Windows NT.

- CICS for OS/400.

Licensing and packaging

The CICS Transaction Gateway is licenced and delivered with:

- CICS Transaction Server for OS/390.
- CICS Transaction Server for VSE/ESA.
- TXSeries Version 4.2, for AIX, Solaris, Windows NT, HP-UX.
- CICS Transaction Server for OS/2 Warp.
- VisualAge for Java Professional or Enterprise Edition.
- WebSphere Application Server Advanced/Enterprise Edition.

The CICS Transaction Gateway may also be downloaded from the CICS Web site for use under a licence obtained via one of the products listed above. The licence permits unlimited use and unlimited right to copy. Note also that the licence delivered with CICS Transaction Server for VSE/ESA permits immediate use of the CICS Transaction Gateway with CICS/VSE Version 2.3.

CICS WEB SUPPORT

Overview

CICS Web Support is a new set of services delivered as part of CICS Transaction Server for VSE/ESA Version 1 Release 1.1.1, enabling direct access to CICS applications and transactions from Web browsers or network computers. This uses the two-tier model and therefore does not require an intermediate system running a Web server or gateway. It provides support for TCP/IP and the HTTP protocol, new APIs for ‘Web aware’ applications, and the ability to access existing applications and 3270 transactions.

One or more TCP/IP ports are assigned to CICS on which CICS listens for incoming requests and then processes them. The CICS network address and port is specified in a standard Universal Resource Locator (URL) which is sent from a Web browser. Also included in the URL is the name of the CICS application program to process the request.

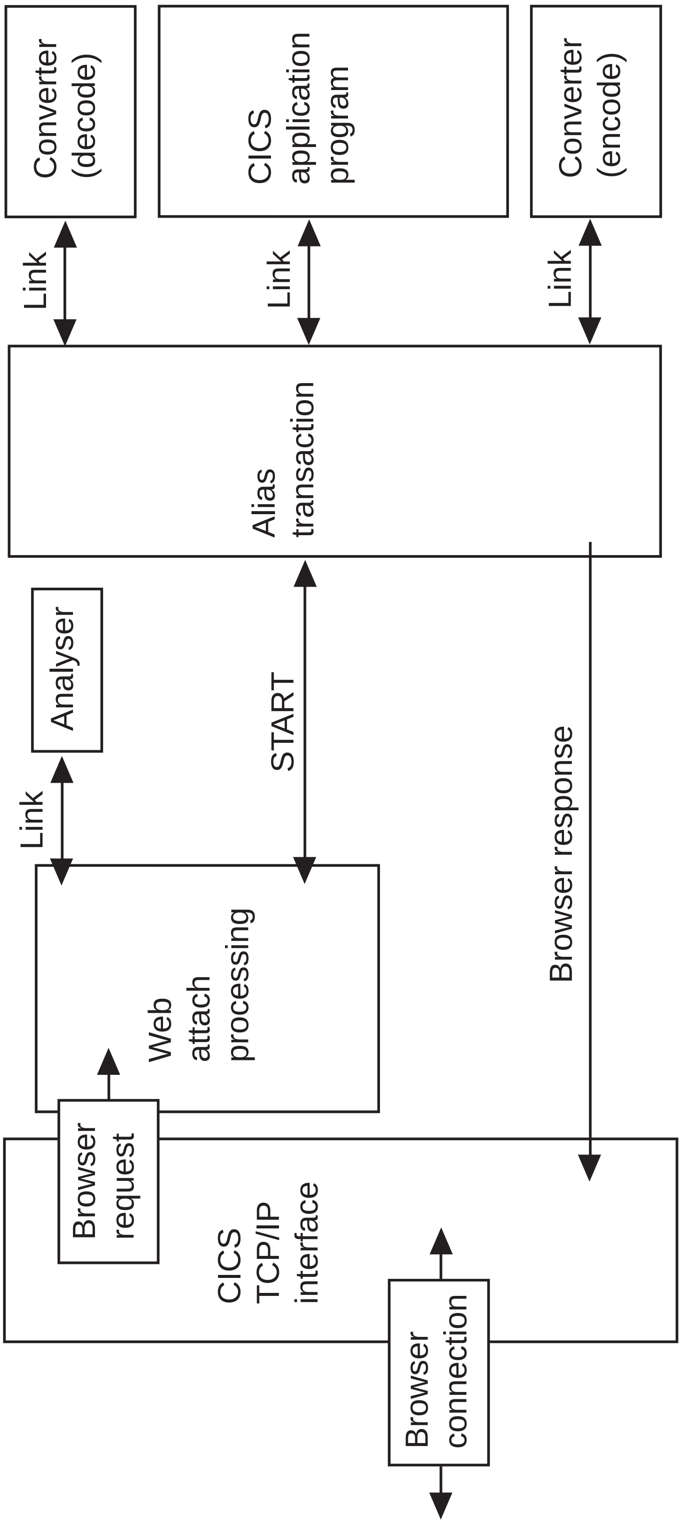

Figure 3: Architecture and flow

Architecture

A CICS application is invoked from a Web browser via a standard URL of the general form ‘http://machine.name:port/converter/alias/program?optional-token’, where:

- *machine.name* is the IP address or DNS name of the CICS region.
- *port* is the configured CICS Web Support listening port number.
- *converter* is the name of the program used for decode and encode processing.
- *alias* is the transaction id of the alias transaction.
- *program* is the name of the application program to be invoked.
- *optional-token* is optional data to be passed with the request.

For example, http://cicstest.hur.ibm.com/cics/cwba/webpgm1.

The architecture and flow is illustrated in Figure 3, and is as follows:

- A request from a Web browser comes into CICS via the CICS TCP/IP interface and the CICS sockets listener, which then attaches a transaction for Web attach processing.
- Web attach processing receives the incoming request, translates the HTTP headers to EBCDIC, and links to the analyser. The analyser interprets the request and specifies the CICS resources needed to process it. On return, if requested by the analyser, the body of the HTTP request is translated to EBCDIC. Web attach processing then starts an alias transaction to handle all further processing of the request and terminates.
- If requested, the alias first calls a converter for decode functions to allow modification of the request before it is passed to the application in the COMMAREA. The alias then calls the CICS application program, passing the COMMAREA as set up by decode.
- The CICS application program processes the request and builds an HTTP response using the new CICS Web-related API

commands, or builds an HTTP response directly in the COMMAREA.

- On return from the application, a converter is called if requested for encode functions that can build or modify the headers and body of the HTTP response. If requested, the alias translates the response to ASCII and then returns the response to the CICS TCP/IP interface and terminates.
- A call is then issued to the operating system TCP/IP services to send the complete HTTP response back to the Web browser.

The analyser

The analyser is a mandatory user-replaceable module, which interprets the incoming request and specifies the CICS resources needed to process it – for example the name of the CICS program, name of the converter, user-id or terminal-id to be associated with the alias transaction, and codepage conversion definition. There is an analyser for each TCP/IP port on which CICS listens for requests.

A replacement analyser could be provided for a number of reasons – for example to allow a different URL convention, to support multiple codepages, or to provide an audit trail. The supplied default analyser:

- Implements the general CICS Web Support URL format.
- Supports an optional token of eight bytes in length.
- Provides ISO-8859-01 codepage conversion (standard latin-1 HTML code page).

The converter

This is an optional user-replaceable module that provides decode and encode functions. Decode is invoked before the incoming request is passed to the CICS application program, and encode is invoked after the CICS application program has processed the request.

The main purpose of decode is to provide the COMMAREA with the request in the format expected by the application. If no converter decode function is specified in the URL, then the entire HTTP request is passed in the COMMAREA to the application program.

The main purpose of encode is to provide a complete HTTP response, that is the HTTP response headers together with the body of the response. It can create or modify headers and the body supplied by the application program. If no encode function is specified, then the application program must build the entire HTTP response in the COMMAREA.

New APIs

There are two new sets of APIs that are provided for the development of 'Web aware' applications, which are summarized in Figure 4. It is recommended that these new APIs be used, although it is possible to write COMMAREA-style applications for handling HTTP requests and responses.

EXEC WEB	
EXTRACT	Get information about incoming HTTP request
READ	Get the value of a specific header or HTML form field
STARTBROWSE, READNEXT, ENDBROWSE	Get header information or HTML form fields sequentially
RECEIVE	Get the body of the HTTP request
WRITE	Add a specific output header to the response
SEND	Select document for delivery as body of the HTTP response
RETRIEVE	Get token of document sent via earlier WEB SEND
EXEC DOCUMENT	
CREATE	Create a new document
INSERT	Insert data into an existing document
SET	Add symbols and values to the document symbol table
RETRIEVE	Retrieve a document into application storage

Figure 4: New APIs

The new EXEC CICS WEB API provides an easier way of handling HTTP requests and responses compared with processing the data as one long string in a COMMAREA.

The new EXEC CICS DOCUMENT API introduces the concept of documents – a collection of formatted data areas. Its prime use is for constructing HTML pages for sending to a Web browser, but is not restricted to this and other mark-up languages could be used. Documents can be made up of both text and binary elements, and can contain templates and symbols – symbol substitution being performed at execution time. Bookmarks can be used to insert data at specific points in documents, and documents can be embedded within documents. Templates containing static data and symbols can be created offline and are new CICS resources defined via RDO with a resource type of DOCTEMPLATE, which can reside in several places – for example CICS files, Transient Data queues, Temporary Storage queues, VSE library members.

Accessing existing COMMAREA-based applications

To access an existing application that is COMMAREA driven, a converter could be used to shield the application from HTTP and HTML.

A decode function could be used to reformat the COMMAREA into the form expected by the application. And, similarly, an encode function could be used to build the HTTP headers and the HTML body of the response after the application program has processed the request and put its results in the COMMAREA.

Accessing existing 3270 transactions

Existing 3270-based transactions are accessed through use of the new 3270 Bridge function. The supplied Web bridge exit program automatically handles conversion of 3270 datastreams to and from HTML, and supports both BMS and 3270 terminal control. No user programming is required, and there are no changes required to the 3270 transaction or associated application programs.

Support is provided for customization of the standard HTML output produced, both for BMS and 3270 terminal control applications. For example it is possible to suppress a PF key or change the name of the text displayed to represent it, to supply a different HTML page background or its colour, or to insert additional text to be displayed in the HTML page.

The 3270 application is invoked with a URL that must specify DFHWBTTA as the CICS program to be invoked, followed by the 4-character ID of the required 3270 transaction to be executed. For example, http://testcics/cics/cwba/dfhwbtta/CECI.

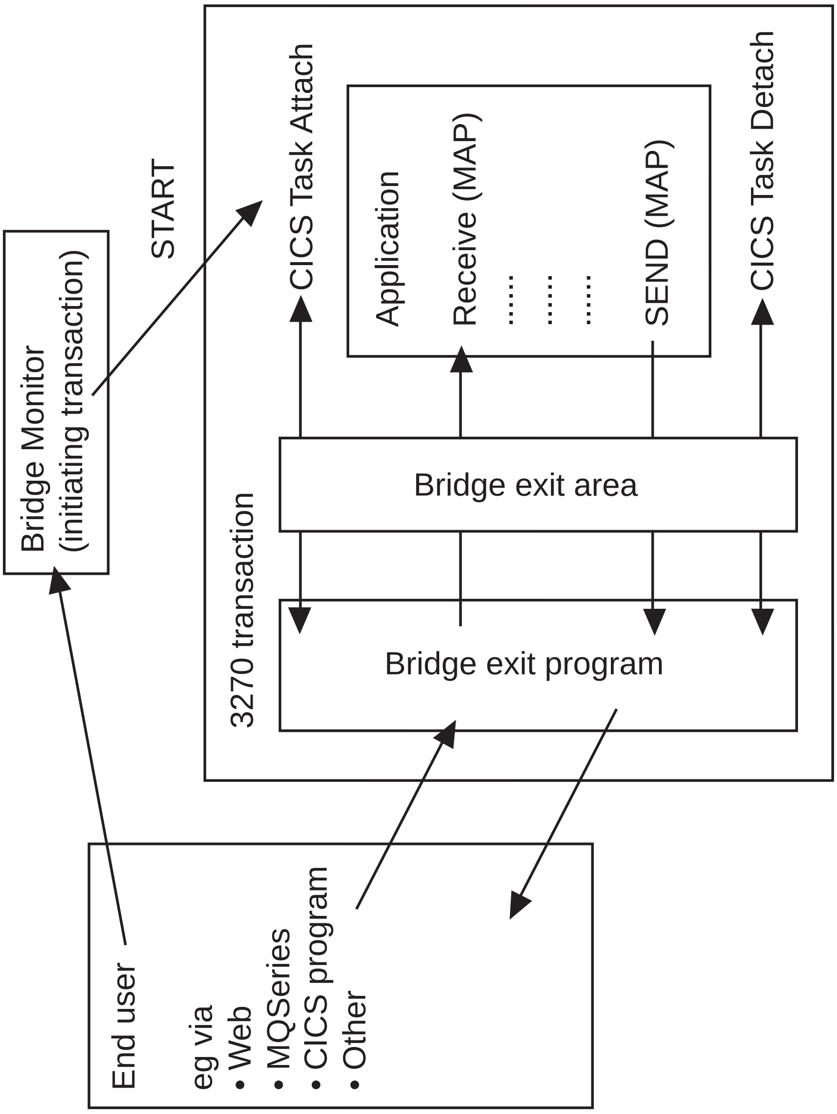

Figure 5: 3270 Bridge environment

Managing CICS Web Support

To enable CICS Web Support, CICS must be started with TCPIP=YES, a new parameter in the System Initialization Table (SIT).

A new CICS resource type, TCPIPSERVICE, defines the instance and characteristics of a TCP/IP service. Multiple TCPIPSERVICE definitions can be used to enable CICS to listen to many TCP/IP ports concurrently. TCPIPSERVICE is defined on-line via the CEDA transaction in the same way as other CICS resources.

There are also new CEMT and System Programming Interface (SPI) commands to perform INQUIRE and SET operations on TCP/IP and Web resources.

The 3270 Bridge

The 3270 Bridge is a new function within CICS Transaction Server that provides the ability to run existing 3270-based transactions without a 3270 and without changing the associated applications. It enables invocation of existing 3270 applications in new ways, for example to add new GUI front-ends and to access CICS from non-CICS environments – for example, from the Internet in conjunction with CICS Web Support, or via MQSeries.

For specified transactions, all 3270 terminal I/O requests are intercepted and handled by a Bridge Exit program, which uses a special COMMAREA called the Bridge Exit Area for communication. The Bridge Exit program is a user-replaceable module, which essentially emulates a 3270 terminal transparently to the 3270 transaction being executed. It intercepts all the 3270 input and output commands issued by the application program, and does all necessary data transformation and routing. The 3270 Bridge environment is illustrated in Figure 5.

There are two supplied Bridge Exit programs. One is a sample supplied in source form which uses Temporary Storage or Transient Data queues for input and output. The other is a program for use only with CICS Web Support and is supplied in object code form.

FURTHER INFORMATION

Further information can be found at the following Web sites:

- http://www.software.ibm.com/ts/cics/:
  – CICS Web Enablement Selection Guide (White Paper)
  – The CICS Transaction Gateway Books
  – The CICS Transaction Server for VSE/ESA books
  – The CICS Internet Guide (documents the CICS Web Support)
  – The CICS External Interfaces Guide (documents the 3270 Bridge).
- http://www.redbooks.ibm.com (ITSO Red Books):
  – Revealed! Architecting Web Access to CICS
  – Revealed! CICS Transaction Gateway with more Clients Unmasked
  – CICS Transaction Server for VSE/ESA: Web Support and 3270 Bridge.

IBM also provides a services offering, Planning Services for CICS Web Enablement, and details can be found at http://www.as.ibm.com/asww/offerings/mww73bE.html.

Chris Smith
CICS Business Unit
IBM United Kingdom Laboratories (UK)

CICS Update is looking for some short hints and tips type article for publication in future issues. If you’ve discovered something or written something that’s brief and would be useful to others working with CICS then let us know. Articles can be sent to any of the addresses shown on page 2

CICS news

IBM has announced Version 2 Release 2 of its COBOL for OS/390 and VM, supporting development and execution of COBOL applications in OS/390 Unix System Services.

It allows the use of the Debug Tool to support debugging of COBOL applications running in the OS/390 Unix environment, via the remote interface. Production debugging capability is available in the MVS/TSO, OS/390 Unix, and CICS environments.

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

* * *

Tivoli has announced its Tivoli Business Systems Manager (TBSM), replacing Tivoli Manager for OS/390, while the Distributed Edition component replaces Tivoli Global Enterprise Manager (GEM). The combination of both in one product, we're told, provides an end-to-end enterprise management system.

The new product also contains many new enhancements from the previous release of Tivoli Manager for OS/390.

It adds CICSPlex/SM as a source of discovery of CICS regions, running under OS/390, VSE, or OS/2, as well as discovering CICS files and transactions. Exception monitoring will be provided though the use of system availability monitoring and realtime analysis.

The Distributed Edition can exploit Tivoli Distributed Monitors, including the ability to map generic distributed monitors to a software component, and supports CICS and DB2 instrumentation to monitor and control Distributed Edition applications on OS/390. Functions common to both include the use of all usability function previously available in Tivoli Manager for OS/390, creation of lines of business via drag-and-drop, SQL 7.0 database for reporting, status propagation, the ability to create LOB views containing both distributed and OS/390 resources, and a pre-packaged NetView application monitoring interface supported with DB2/CICS instrumentation.

For further information contact your local IBM representative.
URL: http://www.tivoli.com/news/press/pressreleases/en/2000/1005_tbsm.html.

* * *

IBM has unveiled its Host Access Client Package, which replaces and upgrades the Personal Communications and Host On-Demand bundle.

Host On-Demand includes CICS Gateway for Java, allowing users to access any number of host sessions and still use their browser for other tasks.

It requires no client installation or middle tier server, having centralized deployment facilities. Only the Web server needs code installation. There's also centralized administration, which means users don't need to know host connectivity settings.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/network/.