215

CICS update

October 2003

In this issue

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 December 1999 issue, are available separately to subscribers for £16.00 ($24.00) 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.

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

When Xephon is given copyright, articles published in *CICS Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) 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.

SOAP support under CICS TS – a technology preview

Simple Object Access Protocol (SOAP) is becoming an emerging standard for platform-independent communication across diverse distributed systems, using HTTP and WebSphere MQ. SOAP specifies a format for messages passed between Web services, with a firewall-friendly way to make remote procedure calls over the Internet.

Microsoft and IBM officially joined the SOAP development effort in May 2000 by co-authoring the SOAP Version 1.1 specification and co-submitting it as a W3C Note, officially signalling the start of the 'Web services revolution'. Before SOAP existed, programs trying to use Web Services had to pull down Web pages and 'scrape' the HTML to look for the appropriate text. SOAP has been widely accepted, because of its ease of use and its independence of underlying protocols, programming languages, and hardware platforms.

HIGHLIGHTS OF SOAP FUNCTIONALITY

SOAP functionality highlights include:

- SOAP messages are sent in a request/response fashion.
- SOAP defines an XML structure to call a method and pass its parameters.
- SOAP defines an XML structure to return values that were requested.
- SOAP defines an XML structure to return faults, if the service provider encounters an error and cannot execute the requested method.
- SOAP provides a common interface to applications, and XML provides a common data format.
- SOAP provides the capability for seamless cross-platform

interoperability between loosely-coupled and dynamically-integrated applications.

- A SOAP specification defines the XML document structure, relying on XML schemas and XML namespaces, for sending Web service requests and responses.
- A SOAP specification does not define how a message, once received, will create an instance of an object and execute the method.
- A SOAP message contains a literal, potentially multi-node, XML document in the SOAP body. A SOAP RPC contains a request/response method invocation in the SOAP body requiring the use of encoding rules.

WHY IS SOAP SUPPORT IMPORTANT TO CICS?

IBM has created CA1M SupportPac as a technology preview, containing SOAP support for CICS Transaction Server. This offering is free. Having SOAP support under CICS is an important step for IBM to demonstrate that CICS legacy applications can participate in e-business, and customers can preserve their investment in existing CICS code, maximizing the re-use of enterprise assets.

CA1M allows access to existing COMMAREA-based applications via SOAP messages. This requires an application layer to map from an XML-based SOAP payload to the COMMAREA and back. WebSphere Studio Enterprise Developer (WSED) provides tools to generate converter routines from COBOL copybooks that can perform these mappings.

SOAP is language-neutral and works with COBOL and PL/I, as well as Java, maximizing opportunities for its usage. Developers can now rely on the expertise and existing proven code of other developers to develop new applications. It is possible now to implement new applications that are XML-aware that can be driven via a SOAP message.

Adding SOAP support in CICS is an evolutionary approach, because it is a fundamental technology using Web services that can be used to access enterprise and legacy applications and data. It allows companies to have their legacy applications reach out and communicate with e-business applications. Web services are platform-independent interfaces that allow communication with other applications using standards-based Internet technologies. Web services simplify the complexity of integration by reducing the number of APIs to one and the number of data formats to one.

SOAP support in CICS is also synergistic with WebSphere and the industry direction for Web services.

SAMPLE SOAP MESSAGES

To demonstrate the simplicity of SOAP, here is a very simple example of adding two numbers together (5+3).

Creating a request:

```
<?xml version="1.Ø" ?>
<SOAP:Envelope xmins:SOAP="urn:schemas-xmlsoap-org:soap.v1">
 <SOAP:Body>
  <AddTwoNumbers>
   <FirstNumber>5</ FirstNumber>
   <SecondNumber>3</ SecondNumber>
  </AddTwoNumbers>
 </SOAP:Body>
</SOAP:Envelope>
```

Creating a response:

Once a SOAP request is received, the object is created and the method is called, sending along the data. The message contains the name of the response and the value of the response.

```
<?xml version="1.Ø" ?>
<SOAP:Envelope xmins:SOAP="urn:schemas-xmlsoap-org:soap.v1">
 <SOAP:Body>
  <AddTwoNumbersResponse>
   <Value>8</ Value>
  </AddTwoNumbersResponse>
 </SOAP:Body>
</SOAP:Envelope>
```

CICS SUPPORTPAC CA1M : SOAP FOR CICS V1.2

CA1M SupportPac is a technology preview, created by IBM, that contains SOAP support for CICS Transaction Server. The use of this SupportPac allows the writing of CICS applications that process SOAP messages. The technology preview is suitable for prototype applications.

This technology preview provides an infrastructure to allow CICS applications to service and make SOAP message requests. It includes transports over HTTP and WebSphere MQ. The HTTP-based support builds on top of functions provided by CICS Web Support and uses that support to manage security and transaction attributes. This includes the ability to use an SSL connection via the HTTPS protocol, but no SOAP-specific security mechanisms have been included.

CICS TS V1.3 applications can accept and process SOAP 1.1 messages sent via HTTP or WebSphere MQ. HTTP access to remote SOAP servers is supported in CICS TS 2.2 only.

When a SOAP message is received, the SupportPac processes the SOAP envelope and handles basic SOAP headers. When the body of the SOAP message (typically an XML document) is reached it is passed on to a program that the user writes for processing. This user-written program could parse the XML document, construct a COMMAREA, and drive an existing business logic program, then construct a response XML document.

The original V1 support pack has been enhanced as V1.2. This version of the SupportPac includes the following new functions:

- A SOAPAction header in outbound messages over HTTP – outbound SOAP messages which use the HTTP transport will now contain the SOAPAction header. The header contains an empty string. This support allows your application to communicate with SOAP servers that require a SOAPAction header to be present in the HTTP request.

- User-written handlers – you can now write your own handler programs to provide additional processing for inbound and outbound SOAP messages.

- Data sharing between handlers – you can pass data between your handler programs using BTS containers.

SOAP FOR CICS PREQUISITES

SOAP for CICS prerequisites include:

- Inbound SOAP requires CICS Transaction Server for OS/390 V1.3 or CICS Transaction Server for z/OS V2.2.
- Outbound SOAP capability requires CICS Transaction Server for z/OS V2.2. The SupportPac includes outbound HTTP support in modules, which will form the PTF for APAR PQ72017.

The XML parser is enabled in the LE runtime with the PTFs for the following APARs:

- OS/390 V2 R10 requires PQ53267 (UQ61947) and PQ57816 (UQ62918).
- z/OS V1 R2 requires PQ57008 (UQ62894).
- z/OS V1 R3 requires PQ60925 (UQ67085).
- z/OS V1 R4 requires PQ66155 (UQ70042).

APAR PQ65085 provides PTFs that enable additional COBOL runtime messages if you use COBOL adapters generated by WSED.

Please review information in APAR II13616 for the latest updates to CICS SOAP support prerequisite maintenance. For the most current information please refer to http://www-3.ibm.com/software/htp/cics/soap/prereq.html.

CICS OUTBOUND HTTP CAPABILITY

APAR PQ72017 introduces CICS outbound HTTP capability that can be used with SOAP.

PQ72017 provides the DFHWBCLI interface, which gives CICS the outbound HTTP capability that can be used with the SOAP technology preview. Please note that the DFHWBCLI interface is not a long-term solution for providing this capability; it will be withdrawn in a future release, when it will be replaced by a different mechanism. Applications written to use the DFHWBCLI interface will need to be altered to work with the later mechanism.

The DFHWBCLI interface is linked to with a COMMAREA interface.

It will provide the following functions:

- Inquire_Proxy – inquires about the standard proxy configured to be used by this CICS region.
- Converse – sends an HTTP request to a remote HTTP server and receives a response from it. CICS creates the request from fields provided in the COMMAREA.
- Send – sends an HTTP request to a remote HTTP server. CICS creates the request from fields provided in the COMMAREA. A session token is returned to be used on a subsequent Receive or Close call.
- Receive – receives an HTTP response to a previous HTTP request. The session token for that request is used to identify the response to be received.
- Close – abandons a previous request without receiving a reply.

Also, CICS must be configured to support the Language Environment runtime. PTFs for the following LE APARs must be applied to enable the XML parser component within the LE runtime: OS/390 V2 R10 requires PQ53267 (UQ61947) and PQ57816 (UQ62918); z/OS V1 R2 requires PQ57008 (UQ62894); z/OS V1 R3 requires PQ60925 (UQ67085); z/OS V1 R4 requires PQ66155 (UQ70042).

If the APARs are not installed, ASRA and AKEA abends may be experienced, with R15 set to 0. The failures happen after leaving the application and calling the LE stub where the entry point is being determined.

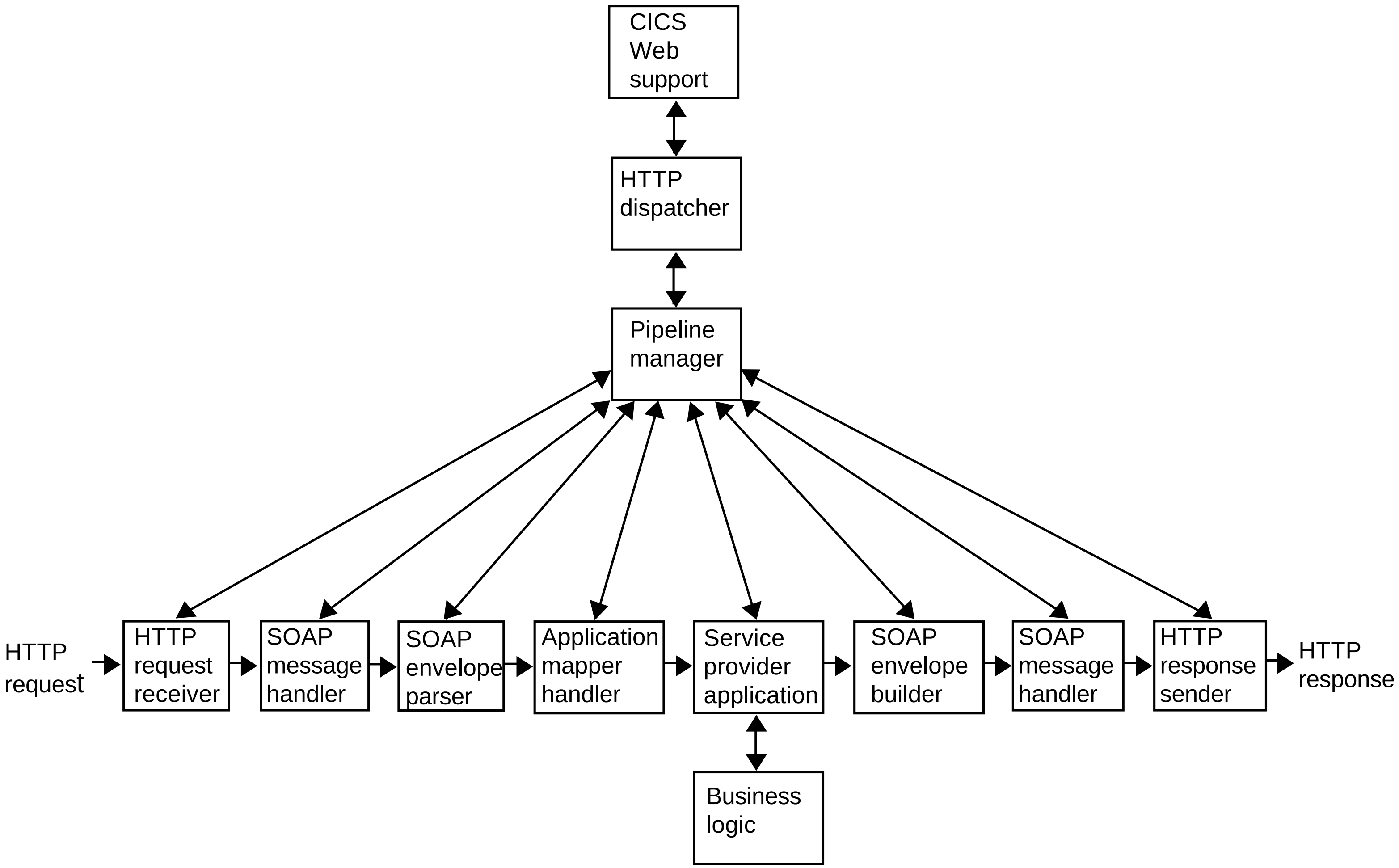

Figure 1: Inbound SOAP message flow

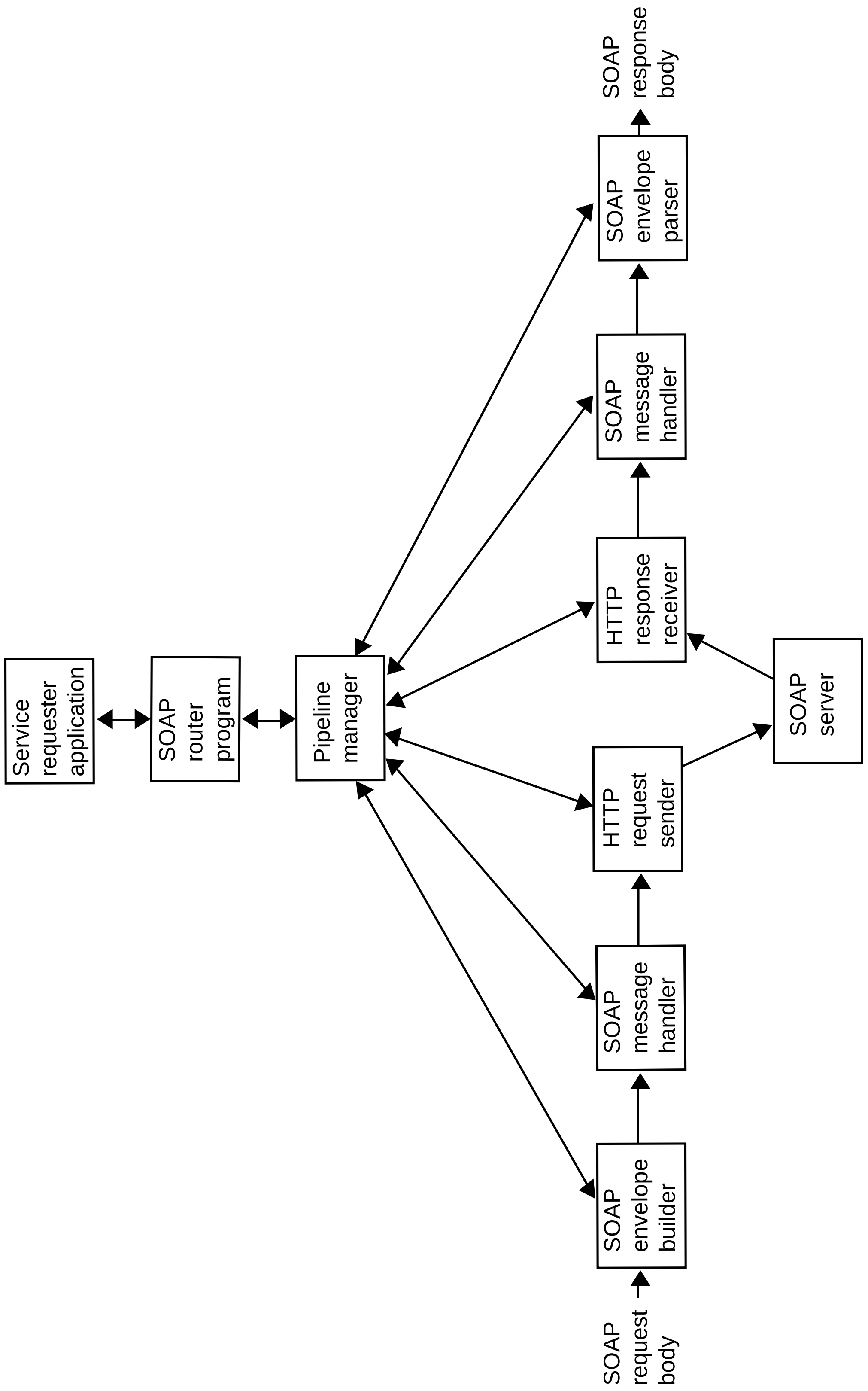

Figure 2: Outbound SOAP message flow

SOAP FOR CICS USE OF BUSINESS TRANSACTION SERVICES

The SOAP for CICS Technology Preview employs CICS Business Transaction Services (BTS).

SOAP applications use CICS BTS as the interface with the SOAP support provided by this SupportPac. For example:

- In a service provider application, the inbound SOAP request body is passed to the application in a BTS container, and the application returns the outbound SOAP response body in another container.
- In a service requester application, the application constructs the outbound SOAP request body in a BTS container, and the response is returned to the application in another BTS container.

  The way that SOAP for CICS uses BTS is very 'light'. One BTS process is created for each SOAP request. Each process completes in the single transaction in which the SOAP 'pipeline' program processes the request. This means that there are two I/Os to the repository file for each request – one at the beginning to write a 50 byte record to reserve the process name, and one at the end to delete that record. This is not an extensive BTS application. Also, since the SOAP for CICS code does not use any of the event features of BTS, it makes no use of the 'local request queue' dataset (DFHLRQ).

Inbound SOAP message handling – a SOAP pipeline for a service provider

The SOAP request contained in an HTTP request is transformed by the pipeline into a SOAP response contained in an HTTP response. The 'pivot point' of the pipeline (the point at which the processing of the request ends and the processing of the response starts) is the service provider application, which links to the business logic.

Figure 1 demonstrates the flow of inbound SOAP messages, where the HTTP dispatcher program, which is invoked by CICS Web support, invokes the pipeline manager, which manages a pipeline.

Outbound SOAP message handling – a SOAP pipeline for a service requester

The SOAP request body created by the application program is transformed by the pipeline into a SOAP request contained in an HTTP request. The 'pivot point' of the pipeline is in the server, which returns a SOAP response contained in an HTTP response. The pipeline transforms this into a SOAP response body, which is returned to the application program.

Figure 2 demonstrates the flow of outbound SOAP messages, where an application program invokes the outbound SOAP router program, which invokes the pipeline manager.

WRITING A SOAP APPLICATION PROGRAM

A SOAP application is an application program that processes SOAP messages. There are two distinct types of SOAP application:

- Service provider application – this receives an inbound SOAP message, processes the contents, and sends a response. Using this SupportPac, you can write service provider applications in CICS Transaction Server for OS/390 Version 1 Release 3, and CICS Transaction Server for z/OS Version 2 Release 2.
- Service requester application – this sends an outbound SOAP message, receives the response, and processes the contents of the response. Using this SupportPac, you can write service requester applications in CICS Transaction Server for z/OS Version 2 Release 2. SOAP applications use CICS business transaction services (BTS) as the interface with the SOAP support.

WRITING A SERVICE PROVIDER APPLICATION

Here are the necessary steps for writing a service provider application:

1 Retrieve the attach event, with the **RETRIEVE REATTACH EVENT** command. The application program is a BTS activity program, which is attached once by the pipeline manager. All BTS activity programs must deal with their re-attachment events.

2 Use the **GET CONTAINER** command to retrieve the request body from the INPUT container. For example:

```
EXEC CICS GET CONTAINER('INPUT')
SET(BODY-PTR)
FLENGTH(BODY-LEN)
```

3 Parse the request body. You can use the language statements that some compilers provide for parsing XML. You can LINK to a parsing program, or you can provide your own parsing code in the body of your program.

4 Using the relevant data from the request body, invoke the business logic. It is advisable to keep the business logic and the manipulation of the SOAP message separate. To do this, put the business logic in a separate program and LINK to it.

5 Using the response from the business logic, construct the body of the SOAP response.

6 Return the response body to the pipeline in the OUTPUT container, using the **PUT CONTAINER** command. For example:

```
EXEC CICS PUT CONTAINER('OUTPUT')
FROM(OUT-BODY)
FLENGTH(OUT-BODY-LEN)
```

WRITING A SERVICE REQUESTER APPLICATION

Here are the necessary steps for writing a service requester application:

1 Perform the business logic that is necessary in order to construct the SOAP request. It is advisable to keep the business logic and the manipulation of the SOAP message separate. For example, you could put the business logic in a separate program and LINK to it.

2 Construct the outbound SOAP request body.

3 Invoke the SOAP outbound pipeline manager.

4 Process the inbound SOAP response.

THE SAMPLE APPLICATIONS INCLUDED IN SUPPORTPAC

The CA1M SupportPac includes a number of sample application programs. Here are a few programs that you may find useful.

SOAPSAMP

SOAPSAMP is a COBOL program that illustrates how to write a service provider application. The program contains all the logic needed to receive a SOAP request body from a client, parse the request body, and return a response to the client. The business logic of the sample program performs an elementary look-up of a fictitious stock quotation. The client sends a body containing a <symbol> element that contains a stock symbol. SOAPSAMP returns a <Quote> element that contains a number obtained from the look-up, accompanied by <Applid>, <Date>, and <Time> elements.

XMLTRACE

XMLTRACE is a COBOL program illustrating how the XML PARSE statement processes an XML document. It traces the XML events and data to SYSOUT.

You can initiate the program in these ways:

- From a SOAP client
- From a Web browser
- From another CICS program.

SOAPBOX

This program is supported in CICS Transaction Server for z/OS Version 2 Release 2 only.

SOAPBOX is a COBOL program that illustrates how to write a service requester application in CICS. The program has a 3270-based user interface, allowing you to submit SOAP requests to SOAPSAMP and display the response. The SOAPBOX program communicates with the SOAPSAMP application using the outbound HTTP support.

SAMPLE SOAP CLIENT – SOAPCLIENT4XG.JAVA

There are a number of SOAP clients available for most popular programming languages. Most provide you with a class library, a COM object or its equivalent, to call from your own program. Typically, the use of these client libraries follows this pattern:

1 Your program passes the name of the remote method to invoke any necessary parameters.

2 The library assembles the appropriate XML document of a SOAP request to package this information.

3 The library passes this XML document to the SOAP server identified by a SOAP endpoint URL, much as you point a browser at a Web server address by specifying the server's URL.

4 After the SOAP server attempts to execute the method, it assembles a SOAP response XML document around the result of the execution and passes it back to the SOAP client.

5 Upon receiving the SOAP response, the client library parses the XML to get the result of the method invocation and passes this result to the program using the library.

Please go to http://www-106.ibm.com/developerworks/xml/library/x-soapcl/ for more details on general purpose SOAP clients.

SOAP FOR CICS NEWSGROUP

IBM has created a discussion newsgroup for customers to share experiences with the SOAP for CICS technology preview at http://www-3.ibm.com/software/htp/cics/soap/newsgroups.html.

You can post statements about the software or simply share good (or bad) experiences with others.

SOAP FUTURE DIRECTIONS FROM IBM'S CICS STRATEGY TEAM

IBM sees great value in adding SOAP support under CICS and it is a strategic direction to add it to the base product in the future.

Mark Cocker, the senior software engineer in IBM's CICS strategy team, stated the following: "We anticipate customers using the SupportPac to prototype solutions involving .NET client applications or intermediary servers submitting SOAP requests into CICS to drive business logic transactions. The SupportPac is free and provided as is with no formal IBM support and therefore only suitable for prototypes and not high-volume business-critical production systems. That being said, the SupportPac relies only on supported CICS application programming interfaces (APIs) (including BTS and the new HTTP outbound LINK interface).

"… Moving forward, it is very likely that the set of technologies that make up Web Services, including SOAP, are going to have a similar dramatic effect on how business-to-business services will be implemented. CICS needs to be positioned to handle SOAP clients and reach out to SOAP servers in an easy to use and evolutionary approach, and this SupportPac is the first step."

On 5 September, IBM announced SOAP for CICS as a fully supported product, incorporating SOAP support into the core CICS products. Please refer to announcement letter # 203-199, entitled *IBM SOAP for CICS feature delivers fully supported SOAP access to CICS,* for details.

The SOAP for CICS feature delivers an enhanced level of the function already available as a Technology Preview in SupportPac CA1M, as a fully supported product for use in production.

There are also technical changes in the SOAP for CICS feature, some of which provide improved error handling to assist in problem determination. Other changes enhance performance or provide extended function. Unlike the Technology Preview, SOAP for CICS is delivered as a feature of CICS Transaction Server for z/OS V2, licensed under the IBM Customer Agreement (ICA); it may be used in production, and is supported by Program Services.

The planned availability date for SOAP for CICS was 26 September 2003.

CONCLUSION

The support of SOAP in CICS is new and needs to be matured. It was initially offered as a technology preview in March 2003 and was enhanced in June 2003, but still is not ready for production usage.

IBM's direction is to incorporate SOAP support into the core CICS products. IBM plans to announce a fully-supported production-ready version of SOAP for CICS during the third quarter of 2003. Combining existing proven CICS code with the new technologies puts IBM on the leading edge, allowing customers to respond and adapt quickly to changes.

Elena Nanos
IBM Certified Solution Expert in CICS Web Enablement
Zurich NA (USA)

Integrating your CICS messages

This article shows how to integrate your CICS messages, which have already been sent to TD queues, with the organization central log or with a Help Desk application, by redirecting the messages to MQSeries. Using MQSeries helps bridge the gap between SNA, which is used in our OS/390 environment, and TCP/IP, which is used in the Help Desk/central logging system.

This facility is needed for:

- Home-written application programs that traditionally send error or alert messages to Transient data.
- Some software packages that use a special queue as an application log.
- The CICS system messages, which are sent by CICS to cs* queue, for instance cssl and csml. These CICS messages contain useful information that should be noted, including that about RDO definitions, security violations, and transaction dumps.

METHOD

The method is rather simple: the task-related user exit of the transient data is used. The queue name is checked to see whether this is a queue name that is of interest. If it is, then the queue context is checked for keywords or patterns and the record is then written to MQSeries. All this is done in the CICS task-related user exit for the transient data.

The USEREXIT is enabled during start-up from the last program called from the PLT. The command to enable the userexit is:

```
(EXEC CICS ENABLE PROGRAM('S5ØPTDEX') GALENGTH(TRDWORK
          EXIT('XTDEREQ') START
```

IMPLEMENTATION EXAMPLE

In our organization, we use Diebold ATMs (Automatic Teller Machines). The Diebold software is written in Assembler in CICS.

We get messages about software, hardware communication, or operational problems in the ATM. A typical message is about a machine that has run out of money, an ATM printer that is not working, or a credit card that has been swallowed by the machine.

There is a CICS transaction to control the ATMs, but this transaction is not used often by the operators, who are students and unfamiliar with 3270 screens.

Using the program below, each TD record written to the DIEBOLD software queue was checked by the program. The part of the text in the record containing the machine name, the error code, and the error description was sent by MQ to our Help Desk system. In the Help Desk system (in our case PeopleSoft Vantive) a case was created, based on the data sent from CICS.

Remarks:

- During compilation, the MQ stub must be at the lked.sysin DDcard of the compilation (see above).
- The overhead of the program is small because we check the queue name at the beginning of the code.
- The data we put into the MQSeries queue is duplicated so we do not lose it. Usually this data will go to a spool or a printer.
- The program is working well in CICS/TS 1.3 with OS390 V2.8 and MQSeries V1.2.
- The program is adjusted to work in a test and production environment without any changes. Since we use a different queue manager for MQ in test and production, the queue and queue manager names are determined using an Assembler macro from the CICS applid.
- MQ can be used, of course, for EBCIDIC to ASCII translation. Since MQ is sensitive to ‘garbage data’ the area in the programs to contain the record context should be clean.

```
//***********************************************************
//* COMPILE AND LINK ASM-CICS PROGRAMS. WITH DBUGGER.            *
//****************************************************************
//ASM EXEC ASMCMQ,NAME='S5ØPTDEX',LLIB='SYSO.CICSTS13.SYSLOAD',
//           SYS='P'
//TRN.SYSIN   DD *
****************************************************************
```

```
*   TRANSIENT DATA USER EXIT (XTDEREQ) INVOKED BEFORE A WRITEQ *
*   TO DO THE FOLLOWING:                                       *
*   THE QUEUE NAME IS CHECK, ONLY SELECTED QUEUE NAMES         *
*   (CICS QUEUE LIKE CSNE OR APPLICATION QUEUE NAMES           *
*   ARE CHECJKED                                               *
****************************************************************
*  COPYBOOKS AND DSECTS REQUIRED BY EXIT PROGRAM               *
****************************************************************
        PRINT  OFF
         DFHUEXIT  TYPE=EP,ID=XTDEREQ
         DFHUEXIT  TYPE=XPIENV      XPI INTERFACE
****************************************************************
        COPY DFHBMSCA            BMS ATTRIBUTE EQUATES
        COPY DFHAID              HANDLE AID EQUATES
        CMQA
****************************************************************
*        E X E C    I N T E R F A C E    S T O R A G E         *
****************************************************************
DFHEISTG DSECT
         DS    CL4
EXITWORK DSECT
         DS    7F
         PRINT ON
****************************************************************
* DCPMJNA STANDARD MESSAGE HEADER                              *
         DS    CL4
LOGTRAN  DS    CL4
         DS    CL4
LOGEXT   DS    CL52
TDREC1   DSECT
         DS    CL15
TD1TERM  DS    ØCL9
TD1TERMC DS    CL4
         DS    CL5
         DS    CL16
TD1TEXT1 DS    CL13
         DS    CL32
TD1DIEB  DS    CL6
TD1TEXT2 DS    CL25
         ORG   TDREC1
TDREC2   DS    ØH
         DS    CL9
TD2PROG  DS    CL8
         DS    CL1
TD2TERM  DS    CL4
         DS    CL1
TD2TEXT  DS    ØCL4Ø
         DS    CL23
TD2ABEND DS    CL5
         DS    CL24
```

```
*        PRINT OFF
S5ØPTDEX DFHEIENT DATAREG=(9),CODEREG=(3)
         B     LØØ1ØØ
         DC    CL8'S5ØPTDEX'
         DC    CL8'&SYSDATE'
         DC    CL8'&SYSTIME'
*       PRINT  ON
****************************************************************
*        PROGRAM CODE FOLLOWS HERE                             *
****************************************************************
LØØ1ØØ   DS    ØH
         LR    R2,R1
         USING DFHUEPAR,R2
         L     R4,UEPGAA
         USING EXITWORK,R4
         L     R5,UEPCLPS
         L     R8,8(R5)              POINT TO MESSAGE
         USING TDREC1,R8
         MVC   MQSYSID,SYSID
         L     R5,4(R5)
         CLC   Ø(4,R5),=C'DJNF'
         BE    LØØ2ØØ         IF YES - ATM QUEUE - DO FARTHER CHECKS
         CLC   Ø(4,R5),=C'SKØ3'
         BE    LØØ2ØØ         IF YES - ATM QUEUE - DO FARTHER CHECKS
         CLC   Ø(4,R5),=C'DJNA'
         BE    LØØ3ØØ SEND TO MQ IF YES - APPLICATION QUEUE
         BAL   R7,ASSIGN
         B     L99999
LØØ2ØØ   DS    ØH
         CLC   TD1TERMC,=C'TERM'     IS THIS A RECORD WE WANT
         BNE   L99999                NO -GET OUT
         CLC   TD1DIEB,=C'DIEB15'    IS THIS A DIEB15 ERROR
         BE    L99999                YES -GET OUT
         CLC   TD1DIEB,=C'DIEB16'    IS THIS A DIEB16 ERROR
         BE    L99999                YES -GET OUT
         BAL   R7,ASKTIME
         MVC   MQTIME,TIME
         MVC   MQDATE,DATE
         MVC   MQTERM,TD1TERM
         MVC   MQDIEB,TD1DIEB
         MVC   MQPROG,=C'S5ØPTDEX'
         MVC   MQSYS,ATM
         MVC   MQTEXT1,TD1TEXT1
         MVC   MQTEXT2,TD1TEXT2
         BAL   R7,MQPUT1
LØØ3ØØ   DS    ØH
         CLC   TD2PROG(4),=C'IWOM'
         BNE   L99999
         CLC   TD2ABEND,=C'ABEND'
         BNE   L99999
         BAL   R7,ASKTIME
         MVC   MQTIME,TIME
         MVC   MQDATE,DATE
         MVC   MQ2TERM,TD2TERM
         MVC   MQPROG,TD2PROG
         MVC   MQSYS,SWIFT
         MVC   MQ2TEXT,TD2TEXT
         BAL   R7,MQPUT1
L99999   DS    ØH
         LA    R15,UERCNORM
         EXEC CICS RETURN
*        DFHEIRET   RCREG=R15
****************************************************************
*  PUT MESSAGE TO VANTIVE USING MQPUT1                         *
****************************************************************
MQPUT1   DS    ØH
         ST    R7,MQPUT17
         CALL  MQPUT1,(HCONN,OBJDESC,MSGDESC,                          X
               PUTMSGOPTS,BUFFERLENGTH,                                X
               BUFFER,COMPCODE,REASON)
         MVI   BUFFER,C' '
         MVC   BUFFER+1(92),BUFFER
         L     R7,MQPUT17
         BR    R7
****************************************************************
HCONN    DC    F'Ø'
         AIF   ('&SYSPARM' EQ 'T').TESTX
         AIF   ('&SYSPARM' EQ 'X').TESTX
         AIF   ('&SYSPARM' EQ 'V').TESTV
         AIF   ('&SYSPARM' EQ 'Y').TESTV
         AIF   ('&SYSPARM' EQ 'P').PRODL
         AIF   ('&SYSPARM' EQ 'L').PRODL
.TESTX   ANOP
OBJDESC  CMQODA OBJECTNAME=VAN.EVENT_LOG_SYS1,                         X
               OBJECTQMGRNAME=MQST
         AGO   .MSG
.TESTV   ANOP
OBJDESC  CMQODA OBJECTNAME=VAN.EVENT_LOG_SYS1,                         X
               OBJECTQMGRNAME=MQSV
         AGO   .MSG
.PRODL   ANOP
OBJDESC  CMQODA OBJECTNAME=VAN.EVENT_LOG_SYS3,                         X
               OBJECTQMGRNAME=MQSC
.MSG     ANOP
**MSGDESC  CMQMDA FORMAT=MQFMT_STRING
MSGDESC  XMQMDA
PUTMSGOPTS CMQPMOA
BUFFERLENGTH  DC   F'93'
         DC    C'MQ COMPCODE AND REASON EYECATCHER'
COMPCODE DS    F
```

```
REASON   DS    F
MQPUT17  DS    F
****************************************************************
*  TEXT OF VANTIVE MESSAGE                                     *
****************************************************************
BUFFER   DS   ØCL93
VNTVMSG  DS   ØCL93
MQDATE   DS   CL8
         DC   C' '
MQTIME   DS   CL8
         DC   C' '
MQSYSID  DS   CL4
         DC   C' '
MQPROG   DS   CL8
MQSYS    DS   CL7                   ' SWIFT '
         DC   C' '
MQDIEB   DC   CL6' '
         DC   C' '
MQTERM   DS   CL9
         DC   C' '
MQTEXT1  DS   CL12
         DC   C' '
MQTEXT2  DS   CL25
         ORG  MQSYSID
         DS   CL5
         DS   CL8
         DS   CL7
MQ2TERM  DS   CL4
         DC   C' '
MQ2TEXT  DS   CL45
         DC   C'     '
****************************************************************
*  DETERMINE SYSID                                             *
****************************************************************
ASSIGN   DS    ØH
         ST    R7,ASSIGNR7
         EXEC CICS ASSIGN SYSID(SYSID)
         L     R7,ASSIGNR7
         BR    R7
ASSIGNR7 DS    F
SYSID    DS    CL4
CICL     DC    C'CICL'
CICP     DC    C'CICP'
CICT     DC    C'CICT'
CICU     DC    C'CICU'
CICV     DC    C'CICV'
CICX     DC    C'CICX'
CICY     DC    C'CICY'
*************************************************************
*       A S K T I M E                                       *
```

```
*************************************************************
ASKTIME  DS    ØH
         ST    R7,TIMER7
         EXEC CICS ASKTIME ABSTIME(PACKTIME)
    EXEC CICS FORMATTIME ABSTIME(PACKTIME) TIME(TIME)                 X
               YYYYMMDD(DATE) TIMESEP
         L     R7,TIMER7
         BR    R7
TIMER7   DS    F
PACKTIME DS    PL8
TIME     DC    CL8' '
DATE     DS    CL8
*************************************************************
*       C O N S T A N T S  &  W O R K I N G  S T O R A G E  *
*************************************************************
FULLZERO DC    F'Ø'
ATM      DC    C'  ATM  '
SWIFT    DC    C' SWIFT '
JUNK     DS    CL1ØØØ
         END
/*
//LKED.SYSLIB DD
//             DD DSN=SYSØ.CICSTS13.SYSLOAD,DISP=SHR
//LKED.SYSIN    DD DATA
   INCLUDE CSQSTUB(CSQCSTUB)
/*
//
```

Uri Cohen
CICS System Programmer (Israel)

Generating a job for DB2 tablespace/indexspace reorganization from CICS

This article shows an example of how to use a CICS PL/I application to generate a job that can be submitted from MVS. CICS JES programming interface commands are used to create a spool file containing the appropriate JCL statements. The problem of identifying tablespaces and indexespaces that require reorganization, according to the well-known rules, is solved through the CICS transaction DBRG shown below. Installation-dependent parameters given in the program are the applid of the corresponding CICS system and the related DB2 subsystem name. A prerequisite to get an accurate list of tablespaces and indexespaces is the recent execution of the STOSPACE utility, and also the RUNSTATS utility for the selected tablespace/ indexspace. For each of the listed elements, the primary, secondary, and actual space allocation are specified, so altering PRIQTY and SECQTY before reorganization can be done in order to avoid an excessive number of VSAM extents. The size of the output datasets in the generated job control is calculated to be sufficient for the REORG utility of the chosen tablespace or indexspace. Program and map compilation, as well as program and plan binding, are done using standard procedures. Catalog table declarations included in the program are generated by DCLGEN (the corresponding table name is put into comments in the same line as the INCLUDE statement).

DBREORM

```
         TITLE 'GROUP PANEL for DB2 reorg       '
         PRINT ON,NOGEN
DBREORM  DFHMSD TYPE=MAP,LANG=PLI,MODE=INOUT,STORAGE=AUTO,SUFFIX=
         TITLE 'HEADER for db2 reorg              '
DBREOR1  DFHMDI SIZE=(7,8Ø),CTRL=(FREEKB,ALARM),MAPATTS=(SOSI),        *
               DSATTS=(SOSI),COLUMN=1,LINE=1,DATA=FIELD,TIOAPFX=YES,   *
               OBFMT=NO
         DFHMDF POS=(1,1),LENGTH=16,INITIAL='DB2REORG-DBREORM',        *
               ATTRB=(PROT,NORM)
         DFHMDF POS=(1,5Ø),LENGTH=6,INITIAL='date :',ATTRB=(PROT,NORM)
* DATUM                           DATE
DATUM    DFHMDF POS=(1,57),LENGTH=8,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(1,66),LENGTH=8,INITIAL='  time :',ATTRB=(PROT,NORM*
               )
* VREME                           TIME
VREME    DFHMDF POS=(1,75),LENGTH=5,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(2,1),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(3,14),LENGTH=53,                                  *
               INITIAL='Generating JOB for REORGANIZATION of TableSpace*
               /IndexSpace',ATTRB=(PROT,NORM)
         DFHMDF POS=(5,19),LENGTH=36,                                  *
               INITIAL='STOSPACE utility run last time on : ',         *
               ATTRB=(PROT,NORM)
* STODAT                          STOSPACE DATE
STODAT   DFHMDF POS=(5,56),LENGTH=8,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(5,65),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(7,7),LENGTH=69,                                   *
               INITIAL='DB-Owner   TS-INDEX  Part   Primary(KB)  Second*
               ary(KB)     Space (KB)',ATTRB=(PROT,NORM)
         TITLE 'ROW for db2 reorg               '
DBREOR2  DFHMDI SIZE=(2,8Ø),COLUMN=SAME,LINE=NEXT,DATA=FIELD,          *
               TIOAPFX=YES,OBFMT=NO
* DB                              DATABASE
DB       DFHMDF POS=(2,7),LENGTH=8,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(2,16),LENGTH=1,ATTRB=(PROT,NORM)
* TS                              TABLESPACE
TS       DFHMDF POS=(2,18),LENGTH=8,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(2,27),LENGTH=1,ATTRB=(PROT,NORM)
* PART                            PART
PART     DFHMDF POS=(2,29),LENGTH=2,ATTRB=(PROT,NUM,NORM,FSET),        *
               PICIN='--',PICOUT='--'
         DFHMDF POS=(2,32),LENGTH=1,ATTRB=(PROT,NORM)
* PRI                             PRI
PRI      DFHMDF POS=(2,34),LENGTH=12,ATTRB=(PROT,NUM,NORM,FSET),       *
               PICIN='------------',PICOUT='------------'
         DFHMDF POS=(2,47),LENGTH=1,ATTRB=(PROT,NORM)
* SEC                             SEC
SEC      DFHMDF POS=(2,49),LENGTH=12,ATTRB=(PROT,NUM,NORM,FSET),       *
               PICIN='------------',PICOUT='------------'
         DFHMDF POS=(2,62),LENGTH=1,ATTRB=(PROT,NORM)
* SPAC                            SPACE
SPAC     DFHMDF POS=(2,64),LENGTH=12,ATTRB=(PROT,NUM,NORM,FSET),       *
               PICIN='------------',PICOUT='------------'
         DFHMDF POS=(2,77),LENGTH=1,ATTRB=(PROT,NORM)
         TITLE 'TRAILER for db2 reorg           '
DBREOR3  DFHMDI SIZE=(2,8Ø),CTRL=(FREEKB,ALARM),COLUMN=1,LINE=22,      *
               DATA=FIELD,TIOAPFX=YES,OBFMT=NO
* PORUKA                          MESSAGE
PORUKA   DFHMDF POS=(1,1),LENGTH=77,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(1,79),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(2,17),LENGTH=47,                                  *
               INITIAL='PF3: end  PF7: backward PF8: forward ENTER :job*
               ',ATTRB=(PROT,NORM)
         DFHMSD TYPE=FINAL
         END
```

DB2REORG

```
 DB2REORG: PROC(KOMPOINT) OPTIONS(MAIN);
 /* ************************************************** */
 /*    PROGRAM QUERIES DB2 CATALOG, DETERMINES THE     */
 /*    LIST OF TABLESPACES AND INDEXES THAT NEED       */
 /*    REORGANIZATION, AND GENERATES JCL FOR REORG     */
 /*    UTILITY INCLUDING SPACE PARAMETERS SUFFICIENT   */
 /*    FOR REORGANIZATION                              */
 /*    TRANS: DBRG                                     */
```

```
/* ************************************************** */
DCL (SELECT, SUBSTR, DATE, TIME, ADDR, BASED, VERIFY) BUILTIN;
DCL (STG, NULL, CSTG, INDEX, LENGTH, LOW) BUILTIN;
DCL EXITNAME CHAR(8);
DCL ENTRYNAME CHAR(8);
DCL CONN_STATUS BIN FIXED(31);
DCL CHAR_SQLCODE CHAR(14);
DCL 1 CHAR_SQLSTR BASED(ADDR(CHAR_SQLCODE)),
    2 CHAR_FLR CHAR(4),
    2 CHAR_SQLCOD CHAR(1Ø);
DCL S BIN FIXED(31);
DCL I BIN FIXED(15);
DCL APOST BIT(8) INIT('Ø11111Ø1'B);
DCL APOSTCH CHAR(1) BASED(ADDR(APOST));
DCL BROJAC BIN FIXED(15) INIT(Ø);
DCL BROJAC1 BIN FIXED(31) INIT(Ø);
DCL KMSG CHAR(8Ø);
DCL IDRED CHAR(8);
DCL EOF BIT(1) INIT('Ø'B);
DCL DATUM CHAR(6);
DCL 1 DAT,
    2 DD CHAR(2),
    2 F1 CHAR(1) INIT('/'),
    2 MM CHAR(2),
    2 F2 CHAR(1) INIT('/'),
    2 GG CHAR(2);
DCL VREME CHAR(9);
DCL 1 VRE,
    2 SAT CHAR(2),
    2 F3 CHAR(1) INIT(':'),
    2 MINUT CHAR(2);
DCL REDSCR PIC '9';
DCL (IND, INDTS) BIT(1);
DCL DSNDBT CHAR(8);
DCL SUB CHAR(3);
DCL (PR1, PR2, PR3, PR4, PR5, PR6, DBN, TSN) CHAR(15) VAR INIT('');
DCL PARTNO CHAR(2) VAR INIT('');
DCL PARTPOM PIC '99';
DCL (PR31, MAXKEY, MAXKEY1, POMMAXKEY) PIC '(15)9' INIT(Ø);
DCL (PR51, MAXTAB, MAXTAB1, POMMAXTAB) PIC '(15)9' INIT(Ø);
DCL (POMPOLJE1, POMPOLJE2, POMPOLJE3, POMPOLJE4) PIC '(15)9';
DCL (POMPOLJE5, POMPOLJE6, POMPOLJE7) PIC '(15)9';
DCL PR32 PIC '(15)Z';
DCL BRIND BIN FIXED(31);
DCL POMSTODAT PIC '999';
DCL (POMDAN, POMMES) PIC '99';
DCL PRESTUP PIC '9';
DCL POMD1 PIC '99V99';
DCL 1 POMD1A BASED(ADDR(POMD1)),
    2 POMD11 PIC '99',
    2 POMD12 PIC '99';
/* ************************************************** */
EXEC SQL INCLUDE DBREORM; /* MAP                      */
EXEC SQL INCLUDE DFHAID;
EXEC SQL INCLUDE DFHBMSCA;
EXEC SQL INCLUDE SQLCA;
                          /* DB2 CATALOG TABLES       */
EXEC SQL INCLUDE SYSIND;     /* SYSIBM.SYSINDEXES     */
EXEC SQL INCLUDE SYSINDPA;   /* SYSIBM.SYSINDEXPART   */
EXEC SQL INCLUDE SYSSTOGR;   /* SYSIBM.SYSSTOGROUP    */
EXEC SQL INCLUDE SYSTABPA;   /* SYSIBM.SYSTABLEPART   */
EXEC SQL INCLUDE SYSTAB;     /* SYSIBM.SYSTABLES      */
EXEC SQL INCLUDE SYSCOL;     /* SYSIBM.SYSCOLUMNS     */
DCL KOMPOINT PTR;         /* COMMON AREA              */
DCL 1 KZ BASED(KOMPOINT),
    2 KZIND      PIC '99999',
    2 KZIND1     PIC '9',
    2 KZSPCDATE  CHAR(8),
    2 KZTS(7),
      3 KZPARTITION  BIN FIXED(15),
      3 KZTSNAME     CHAR(8),
      3 KZDBNAME     CHAR(8),
      3 KZPQTY       BIN FIXED(31),
      3 KZSQTY       BIN FIXED(31),
      3 KZSPACE      BIN FIXED(31);
DCL 1 TSSL,
    2 PARTITION  BIN FIXED(15),
    2 TSNAME     CHAR(8),
    2 DBNAME     CHAR(8),
    2 PQTY       BIN FIXED(31),
    2 SQTY       BIN FIXED(31),
    2 SPACE      BIN FIXED(31);
EXEC SQL WHENEVER SQLERROR   GO TO SQLERR;
EXEC SQL WHENEVER SQLWARNING GO TO SQLERR;
EXEC SQL WHENEVER NOT FOUND  CONTINUE;
/* ************ MAIN PROGRAM ************************ */
EXEC CICS IGNORE CONDITION QIDERR;
EXITNAME  = 'DSN2EXT1';
ENTRYNAME = 'DSNCSQL';
EXEC CICS INQUIRE EXITPROGRAM(EXITNAME) ENTRYNAME(ENTRYNAME)
          CONNECTST(CONN_STATUS) NOHANDLE;
IF CONN_STATUS ¬= DFHVALUE(CONNECTED) THEN DO;
  KMSG = 'DB2-CICS NOT CONNECTED';
  CALL GRESKA(KMSG);
END;
IF EIBCALEN = Ø THEN ALLOCATE KZ;
CALL NULLMAP(ADDR(DBREOR1O), CSTG(DBREOR1O));
CALL NULLMAP(ADDR(DBREOR2O), CSTG(DBREOR2O));
CALL NULLMAP(ADDR(DBREOR3O), CSTG(DBREOR3O));
IDRED = EIBTRMID || 'DBRG';
```

```
DATUM = DATE;
VREME = TIME;
DD = SUBSTR(DATUM,5,2);
MM = SUBSTR(DATUM,3,2);
GG = SUBSTR(DATUM,1,2);
SAT = SUBSTR(VREME,1,2);
MINUT = SUBSTR(VREME,3,2);
DBREOR1O.DATUMO = DD || F1 || MM || F2 || GG;
DBREOR1O.VREMEO = SAT || F3 || MINUT;
DBREOR1O.STODATO = KZSPCDATE;
BROJAC = Ø;
EXEC CICS ASSIGN APPLID(DSNDBT);
/* INSTALLATION DEPENDENT      */
IF DSNDBT = 'PSTEST29' THEN DO;
  SUB = 'DBT';
END;
ELSE DO;
  SUB = 'DSN';
END;
EXEC CICS RECEIVE MAP('DBREOR1') MAPSET('DBREORM') RESP(S);
IF S = DFHRESP(MAPFAIL) THEN DO;
  DBREOR3O.PORUKAO = ' ';
  CALL KREITS;
  POMSTODAT = SUBSTR(DCLSYSSTOGROUP.SPCDATE, 3, 3);
  PRESTUP = Ø;
  IF SUBSTR(DCLSYSSTOGROUP.SPCDATE, 1, 2) = 'ØØ' THEN PRESTUP = 1;
  ELSE DO;
    POMD1 = SUBSTR(DCLSYSSTOGROUP.SPCDATE, 1, 2);
    IF POMD12 ¬= Ø THEN PRESTUP = 1;
  END;
  POMMES = 1;
  IF POMSTODAT < 32 THEN POMDAN = POMSTODAT;
  ELSE DO;
    POMMES = POMMES + 1;
    IF POMSTODAT < (6Ø + PRESTUP) THEN POMDAN = POMSTODAT - 31;
    ELSE DO;
      POMMES = POMMES + 1;
      IF POMSTODAT < (91 + PRESTUP)
        THEN POMDAN = POMSTODAT - (59 + PRESTUP);
      ELSE DO;
        POMMES = POMMES + 1;
        IF POMSTODAT < (121 + PRESTUP)
          THEN POMDAN = POMSTODAT - (9Ø + PRESTUP);
        ELSE DO;
          POMMES = POMMES + 1;
          IF POMSTODAT < (152 + PRESTUP)
            THEN POMDAN = POMSTODAT - (12Ø + PRESTUP);
          ELSE DO;
            POMMES = POMMES + 1;
            IF POMSTODAT < (182 + PRESTUP)
```

```
              THEN POMDAN = POMSTODAT - (151 + PRESTUP);
            ELSE DO;
              POMMES = POMMES + 1;
              IF POMSTODAT < (213 + PRESTUP)
                THEN POMDAN = POMSTODAT - (181 + PRESTUP);
              ELSE DO;
                POMMES = POMMES + 1;
                IF POMSTODAT < (244 + PRESTUP)
                  THEN POMDAN = POMSTODAT - (212 + PRESTUP);
                ELSE DO;
                  POMMES = POMMES + 1;
                  IF POMSTODAT < (274 + PRESTUP)
                    THEN POMDAN = POMSTODAT - (243 + PRESTUP);
                  ELSE DO;
                    POMMES = POMMES + 1;
                    IF POMSTODAT < (3Ø5 + PRESTUP)
                      THEN POMDAN = POMSTODAT - (273 + PRESTUP);
                    ELSE DO;
                      POMMES = POMMES + 1;
                      IF POMSTODAT < (335 + PRESTUP)
                        THEN POMDAN = POMSTODAT - (3Ø4 + PRESTUP);
                      ELSE DO;
                        POMMES = POMMES + 1;
                        POMDAN = POMSTODAT - (334 + PRESTUP);
                      END;
                    END;
                  END;
                END;
              END;
            END;
          END;
        END;
      END;
    END;
  END;
  DBREOR1O.STODATO = POMDAN || '.' || POMMES || '.' ||
                     SUBSTR(DCLSYSSTOGROUP.SPCDATE, 1, 2);
  KZSPCDATE = DBREOR1O.STODATO;
  EXEC CICS SEND MAP('DBREOR1') MAPSET('DBREORM') ACCUM FREEKB
       ERASE;
  CALL NAPRED;     /* SCROLL FORWARD */
  CALL POPUNI;
END;
SELECT (EIBAID);
  WHEN (DFHENTER) DO;
    CALL JES;
    DBREOR3O.PORUKAO = ' GENERATED JOB WITH JOBNAME   ' || DSNDBT ||
                       ' - SEE SDSF: SD.OA OPTION SE';
    EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
    EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
  END;
  WHEN (DFHPF3) DO;
    EXEC CICS DELETEQ TS QUEUE(IDRED);
    KMSG = '***END OF WORK***';
    EXEC CICS SEND FROM(KMSG) ERASE;
    EXEC CICS RETURN;
  END;
  WHEN (DFHPF7, DFHPF8) DO;
    EXEC CICS SEND MAP('DBREOR1') MAPSET('DBREORM') ACCUM FREEKB;
    IF EIBAID = DFHPF7 THEN CALL NAZAD;  /* SCROLL BACKWARD */
    ELSE CALL NAPRED;
  END;
  OTHERWISE DO;
    EXEC CICS SEND MAP('DBREOR1') MAPSET('DBREORM') ACCUM FREEKB;
    DBREOR3O.PORUKAO = (4Ø)' '||' ***   WRONG KEY    *** ';
    CALL POPUNI;
  END;
END;
/* *** P R O C E D U R E: *************************** */
KREITS: PROC;
  EXEC CICS DELETEQ TS QUEUE(IDRED);
  EXEC SQL SELECT MAX(SPCDATE)
           INTO :DCLSYSSTOGROUP.SPCDATE
           FROM SYSIBM.SYSSTOGROUP
           WHERE NAME LIKE 'SG%';
  EXEC SQL DECLARE C1 CURSOR FOR
    SELECT PARTITION, TSNAME, DBNAME, PQTY, SQTY, SPACE
    FROM SYSIBM.SYSTABLEPART
    WHERE (PQTY * 4 + 3 * SQTY * 4) < SPACE AND
          TSNAME LIKE 'TS%'
    UNION
    SELECT PARTITION, SUBSTR(IXNAME, 1, 8),
           IXCREATOR, PQTY, SQTY, SPACE
    FROM SYSIBM.SYSINDEXPART
    WHERE (PQTY * 4 + 3 * SQTY * 4) < SPACE AND
          IXNAME IN (SELECT NAME
                     FROM SYSIBM.SYSINDEXES
                     WHERE TBNAME LIKE 'TB%')
    ORDER BY 2,1;
  EXEC SQL OPEN C1;
  EXEC SQL FETCH C1 INTO :DCLSYSTABLEPART.PARTITION,
                         :DCLSYSTABLEPART.TSNAME,
                         :DCLSYSTABLEPART.DBNAME,
                         :DCLSYSTABLEPART.PQTY,
                         :DCLSYSTABLEPART.SQTY,
                         :DCLSYSTABLEPART.SPACE;
  IF SQLCODE = 1ØØ THEN DO;
    EXEC SQL CLOSE C1;
    DBREOR3O.PORUKAO = 'NO DATA FOR GIVEN CONDITION  ';
  END;
```

```
    DO WHILE (SQLCODE = Ø);
      BROJAC1   = BROJAC1 + 1;
      TSSL.PARTITION = DCLSYSTABLEPART.PARTITION;
      TSSL.TSNAME    = DCLSYSTABLEPART.TSNAME;
      TSSL.DBNAME    = DCLSYSTABLEPART.DBNAME;
      TSSL.PQTY      = DCLSYSTABLEPART.PQTY * 4;
      TSSL.SQTY      = DCLSYSTABLEPART.SQTY * 4;
      TSSL.SPACE     = DCLSYSTABLEPART.SPACE;
      EXEC CICS WRITEQ TS QUEUE(IDRED)
                          FROM(TSSL)
                          ITEM(BROJAC1) AUXILIARY;
      EXEC SQL FETCH C1 INTO :DCLSYSTABLEPART.PARTITION,
                             :DCLSYSTABLEPART.TSNAME,
                             :DCLSYSTABLEPART.DBNAME,
                             :DCLSYSTABLEPART.PQTY,
                             :DCLSYSTABLEPART.SQTY,
                             :DCLSYSTABLEPART.SPACE;
    END;
    EXEC SQL CLOSE C1;
  END KREITS;
  NAPRED: PROC;
    BROJAC = Ø;
    I = KZIND;
    DO WHILE (BROJAC < 7 & ¬EOF);
      I = I + 1;
      EXEC CICS READQ TS QUEUE(IDRED)
                         INTO(TSSL)
                         ITEM(I)
                         RESP(S);
      IF S = DFHRESP(NORMAL) THEN DO;
        BROJAC = BROJAC + 1;
        DBREOR2O.PARTO = TSSL.PARTITION;
        DBREOR2O.TSO   = TSSL.TSNAME;
        DBREOR2O.DBO   = TSSL.DBNAME;
        DBREOR2O.PRIO  = TSSL.PQTY;
        DBREOR2O.SECO  = TSSL.SQTY;
        DBREOR2O.SPACO = TSSL.SPACE;
        KZPARTITION(BROJAC) = TSSL.PARTITION;
        KZTSNAME(BROJAC)    = TSSL.TSNAME;
        KZDBNAME(BROJAC)    = TSSL.DBNAME;
        KZPQTY(BROJAC)      = TSSL.PQTY;
        KZSQTY(BROJAC)      = TSSL.SQTY;
        KZSPACE(BROJAC)     = TSSL.SPACE;
        EXEC CICS SEND MAP('DBREOR2') MAPSET('DBREORM') ACCUM FREEKB;
        KZIND = I;
        KZIND1 = BROJAC;
      END;
      ELSE EOF = '1'B;
    END;
    DBREOR3O.PORUKAO = 'PLACE CURSOR ON CHOSEN ROW AND <ENTER>';
    CALL POPUNI;
  END NAPRED;
  NAZAD: PROC;
    BROJAC = Ø;
    IF KZIND - (7 + KZIND1) < 1 THEN I = Ø;
    ELSE I = KZIND - (7 + KZIND1);
    DO WHILE (BROJAC < 7 & ¬EOF);
      I = I + 1;
      EXEC CICS READQ TS QUEUE(IDRED)
                         INTO(TSSL)
                         ITEM(I)
                         RESP(S);
      IF S = DFHRESP(NORMAL) THEN DO;
        BROJAC = BROJAC + 1;
        DBREOR2O.PARTO = TSSL.PARTITION;
        DBREOR2O.TSO   = TSSL.TSNAME;
        DBREOR2O.DBO   = TSSL.DBNAME;
        DBREOR2O.PRIO  = TSSL.PQTY;
        DBREOR2O.SECO  = TSSL.SQTY;
        DBREOR2O.SPACO = TSSL.SPACE;
        KZPARTITION(BROJAC) = TSSL.PARTITION;
        KZTSNAME(BROJAC)    = TSSL.TSNAME;
        KZDBNAME(BROJAC)    = TSSL.DBNAME;
        KZPQTY(BROJAC)      = TSSL.PQTY;
        KZSQTY(BROJAC)      = TSSL.SQTY;
        KZSPACE(BROJAC)     = TSSL.SPACE;
        EXEC CICS SEND MAP('DBREOR2') MAPSET('DBREORM') ACCUM FREEKB;
        KZIND = I;
        KZIND1 = BROJAC;
      END;
      ELSE EOF = '1'B;
    END;
    DBREOR3O.PORUKAO = 'PLACE CURSOR ON CHOSEN ROW AND <ENTER>';
    CALL POPUNI;
  END NAZAD;
  POPUNI: PROC;
    DO I = (BROJAC + 1) TO 7;
      DBREOR2O.DBO  = ' ';
      DBREOR2O.TSO = ' ';
      DBREOR2O.PARTO = Ø;
      DBREOR2O.PRIO = Ø;
      DBREOR2O.SECO = Ø;
      DBREOR2O.SPACO = Ø;
      KZPARTITION(I) = Ø;
      KZTSNAME(I)    = ' ';
      KZDBNAME(I)    = ' ';
      KZPQTY(I)      = Ø;
      KZSQTY(I)      = Ø;
      KZSPACE(I)     = Ø;
      EXEC CICS SEND MAP('DBREOR2') MAPSET('DBREORM') ACCUM FREEKB;
    END;
    EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM') ACCUM FREEKB;
    EXEC CICS SEND PAGE;
    EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
  END POPUNI;
  JES: PROC;
    IF ¬((EIBCPOSN >= 64Ø & EIBCPOSN <= (64Ø + 79)) |
         (EIBCPOSN >= 8ØØ & EIBCPOSN <= (8ØØ + 79)) |
         (EIBCPOSN >= 96Ø & EIBCPOSN <= (96Ø + 79)) |
         (EIBCPOSN >= 112Ø & EIBCPOSN <= (112Ø + 79)) |
         (EIBCPOSN >= 128Ø & EIBCPOSN <= (128Ø + 79)) |
         (EIBCPOSN >= 144Ø & EIBCPOSN <= (144Ø + 79)) |
         (EIBCPOSN >= 16ØØ & EIBCPOSN <= (16ØØ + 79))) THEN DO;
      DBREOR3O.PORUKAO = ' CURSOR NOT PROPERLY POSITIONED';
      EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
      EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
    END;
    IND = 'Ø'B;
    IF EIBCPOSN >= 64Ø & EIBCPOSN <= (64Ø + 79) THEN DO;
      REDSCR = 1;
      IF KZTSNAME(REDSCR) = ' ' THEN IND = '1'B;
    END;
    IF EIBCPOSN >= 8ØØ & EIBCPOSN <= (8ØØ + 79) THEN DO;
      REDSCR = 2;
      IF KZTSNAME(REDSCR) = ' ' THEN IND = '1'B;
    END;
    IF EIBCPOSN >= 96Ø & EIBCPOSN <= (96Ø + 79) THEN DO;
      REDSCR = 3;
      IF KZTSNAME(REDSCR) = ' ' THEN IND = '1'B;
    END;
    IF EIBCPOSN >= 112Ø & EIBCPOSN <= (112Ø + 79) THEN DO;
      REDSCR = 4;
      IF KZTSNAME(REDSCR) = ' ' THEN IND = '1'B;
    END;
    IF EIBCPOSN >= 128Ø & EIBCPOSN <= (128Ø + 79) THEN DO;
      REDSCR = 5;
      IF KZTSNAME(REDSCR) = ' ' THEN IND = '1'B;
    END;
    IF EIBCPOSN >= 144Ø & EIBCPOSN <= (144Ø + 79) THEN DO;
      REDSCR = 6;
      IF KZTSNAME(REDSCR) = ' ' THEN IND = '1'B;
    END;
    IF EIBCPOSN >= 16ØØ & EIBCPOSN <= (16ØØ + 79) THEN DO;
      REDSCR = 7;
      IF KZTSNAME(REDSCR) = ' ' THEN IND = '1'B;
    END;
    IF IND THEN DO;
      DBREOR3O.PORUKAO = ' CURSOR POSITIONED ON EMPTY ROW  ';
      EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
      EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
    END;
    INDTS = 'Ø'B;
    IF SUBSTR(KZDBNAME(REDSCR), 1, 2) = 'DB' THEN INDTS = '1'B;
    DBN = '';
    TSN = '';
    DO I = 1 TO 8;
      IF SUBSTR(KZDBNAME(REDSCR), I, 1) ¬= ' '
        THEN DBN = DBN || SUBSTR(KZDBNAME(REDSCR), I, 1);
      IF SUBSTR(KZTSNAME(REDSCR), I, 1) ¬= ' '
        THEN TSN = TSN || SUBSTR(KZTSNAME(REDSCR), I, 1);
    END;
    CALL OBRADA;
    CALL PISI;
  END JES;
  PISI: PROC;
  DCL RED CHAR(8Ø);
  DCL TOKEN CHAR(8);
  DCL OUTLEN BIN FIXED(31) INIT(8Ø);
    TOKEN = LOW(8);
    PARTPOM = KZPARTITION(REDSCR);
    IF PR1 = '' THEN PR1 = '1';
    IF PR2 = '' THEN PR2 = '1';
    IF PR3 = '' THEN PR3 = '1';
    IF PR4 = '' THEN PR4 = '1';
    IF PR5 = '' THEN PR5 = '1';
    IF PR6 = '' THEN PR6 = '1';
    EXEC CICS SPOOLOPEN OUTPUT TOKEN(TOKEN)
                        NODE('*') USERID('*')
                        CLASS('A') RECORDLENGTH(8Ø)
                        PRINT
                        NOHANDLE;
    IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
      DBREOR3O.PORUKAO = ' ERROR AT SPOOL OPENING    ';
      EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
      EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
    END;
    RED = '//BANKT   JOB MSGCLASS=X,TIME=144Ø,REGION=4M,' ||
          'NOTIFY=&SYSUID';
    CALL PISI1(RED, TOKEN);
    RED = '//* *** WARNING :  ****************************** *';
    CALL PISI1(RED, TOKEN);
    RED = '//* *** BEFORE EXECUTING REORG FOR TABLESPACE     *';
    CALL PISI1(RED, TOKEN);
    RED = '//* *** WITH OPTION LOG NO,IMAGECOPY IS REQUIRED!!*';
    CALL PISI1(RED, TOKEN);
    RED = '//* *** AFTER REORG TS,IMAGECOPY IS REQUIRED    !!*';
    CALL PISI1(RED, TOKEN);
    RED = '//* ********************************************* *';
    CALL PISI1(RED, TOKEN);
    RED = '//STEPØØØ1 EXEC DSNUPROC,SYSTEM=' || APOSTCH || SUB ||
          APOSTCH || ',' || 'UID=' || APOSTCH || 'BANK' || APOSTCH ||
          ',UTPROC=' || APOSTCH || APOSTCH;
    CALL PISI1(RED, TOKEN);
    RED = '//DSNUPROC.SORTWKØ1 DD DSN=BANKT.SORTWKØ1.PRIV,';
    CALL PISI1(RED, TOKEN);
    RED = '//      DISP=(MOD,DELETE,CATLG),';
    CALL PISI1(RED, TOKEN);
    RED = '//      SPACE=(16384,(' || PR1 || ',' || PR2 ||
          '),,,ROUND),';
    CALL PISI1(RED, TOKEN);
    RED = '//      UNIT=SYSDA';
    CALL PISI1(RED, TOKEN);
    RED = '//DSNUPROC.SORTWKØ2 DD DSN=BANKT.SORTWKØ2.PRIV,';
    CALL PISI1(RED, TOKEN);
    RED = '//      DISP=(MOD,DELETE,CATLG),';
    CALL PISI1(RED, TOKEN);
    RED = '//      SPACE=(16384,(' || PR1 || ',' || PR2 ||
          '),,,ROUND),';
    CALL PISI1(RED, TOKEN);
    RED = '//      UNIT=SYSDA';
    CALL PISI1(RED, TOKEN);
    RED = '//DSNUPROC.SORTWKØ3 DD DSN=BANKT.SORTWKØ3.PRIV,';
    CALL PISI1(RED, TOKEN);
    RED = '//      DISP=(MOD,DELETE,CATLG),';
    CALL PISI1(RED, TOKEN);
    RED = '//      SPACE=(16384,(' || PR1 || ',' || PR2 ||
          '),,,ROUND),';
    CALL PISI1(RED, TOKEN);
    RED = '//      UNIT=SYSDA';
    CALL PISI1(RED, TOKEN);
    RED = '//DSNUPROC.SORTWKØ4 DD DSN=BANKT.SORTWKØ4.PRIV,';
    CALL PISI1(RED, TOKEN);
    RED = '//      DISP=(MOD,DELETE,CATLG),';
    CALL PISI1(RED, TOKEN);
    RED = '//      SPACE=(16384,(' || PR1 || ',' || PR2 ||
          '),,,ROUND),';
    CALL PISI1(RED, TOKEN);
    RED = '//      UNIT=SYSDA';
    CALL PISI1(RED, TOKEN);
    RED = '//DSNUPROC.SYSUT1 DD DSN=BANKT.SYSUT1.PRIV,';
    CALL PISI1(RED, TOKEN);
    RED = '//      DISP=(MOD,DELETE,CATLG),';
    CALL PISI1(RED, TOKEN);
    RED = '//      SPACE=(16384,(' || PR3 || ',' || PR4 ||
          '),,,ROUND),';
    CALL PISI1(RED, TOKEN);
    RED = '//      UNIT=SYSDA';
    CALL PISI1(RED, TOKEN);
    RED = '//DSNUPROC.SORTOUT DD DSN=BANKT.SORTOUT.PRIV,';
    CALL PISI1(RED, TOKEN);
```

```
RED = '//     DISP=(MOD,DELETE,CATLG),';
CALL PISI1(RED, TOKEN);
RED = '//     SPACE=(16384,(' || PR3 || ',' || PR4 ||
      '),,,ROUND),';
CALL PISI1(RED, TOKEN);
RED = '//     UNIT=SYSDA';
CALL PISI1(RED, TOKEN);
IF INDTS THEN DO;
  RED = '//DSNUPROC.SYSREC DD DSN=BANKT.SREORG.PRIV,';
  CALL PISI1(RED, TOKEN);
  RED = '//     DISP=(MOD,DELETE,CATLG),';
  CALL PISI1(RED, TOKEN);
  RED = '//     SPACE=(16384,(' || PR5 || ',' || PR6 ||
        '),,,ROUND),';
  CALL PISI1(RED, TOKEN);
  RED = '//     UNIT=SYSDA';
  CALL PISI1(RED, TOKEN);
  RED = '//DSNUPROC.SYSIN    DD  *';
  CALL PISI1(RED, TOKEN);
  RED = '  REORG TABLESPACE ' || DBN || '.' || TSN;
  CALL PISI1(RED, TOKEN);
  RED = '  LOG NO';
  CALL PISI1(RED, TOKEN);
  IF PARTPOM > Ø THEN DO;
    DO I = 1 TO 2;
      IF SUBSTR(PARTPOM, I, 1) ¬= 'Ø'
        THEN PARTNO = PARTNO || SUBSTR(PARTPOM, I, 1);
    END;
    RED = '  PART ' || PARTNO;
    CALL PISI1(RED, TOKEN);
  END;
  RED = '  UNLOAD CONTINUE';
  CALL PISI1(RED, TOKEN);
END;
ELSE DO;
  RED = '//DSNUPROC.SYSIN    DD  *';
  CALL PISI1(RED, TOKEN);
  RED = '  REORG INDEX ' || DBN || '.' || TSN;
  CALL PISI1(RED, TOKEN);
  IF PARTPOM > Ø THEN DO;
    DO I = 1 TO 2;
      IF SUBSTR(PARTPOM, I, 1) ¬= 'Ø'
        THEN PARTNO = PARTNO || SUBSTR(PARTPOM, I, 1);
    END;
    RED = '  PART ' || PARTNO;
    CALL PISI1(RED, TOKEN);
  END;
  RED = '  UNLOAD CONTINUE';
  CALL PISI1(RED, TOKEN);
END;
```

```
  RED = '//';
  CALL PISI1(RED, TOKEN);
  EXEC CICS SPOOLCLOSE TOKEN(TOKEN) NOHANDLE;
  IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
    DBREOR3O.PORUKAO = ' ERROR AT SPOOL CLOSING    ';
    EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
    EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
  END;
END PISI;
PISI1: PROC(RED1, TOKEN1);
  DCL RED1 CHAR(8Ø);
  DCL TOKEN1 CHAR(8);
  DCL OUTLEN1 BIN FIXED(31) INIT(8Ø);
  EXEC CICS SPOOLWRITE TOKEN(TOKEN1)
                       FROM(RED1) FLENGTH(OUTLEN1)
                       LINE
                       NOHANDLE;
  IF EIBRESP ¬= DFHRESP(NORMAL) THEN DO;
    DBREOR3O.PORUKAO = ' ERROR AT SPOOL WRITING    ';
    EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
    EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
  END;
END PISI1;
OBRADA: PROC;
  IF INDTS THEN DO;
    DCLSYSTABLES.DBNAME = DBN;
    DCLSYSTABLES.TSNAME = TSN;
    EXEC SQL DECLARE C2 CURSOR FOR
      SELECT NAME, CARD, CREATOR
      FROM SYSIBM.SYSTABLES
      WHERE TSNAME = :DCLSYSTABLES.TSNAME AND
            DBNAME = :DCLSYSTABLES.DBNAME AND
            TYPE = 'T';
    EXEC SQL OPEN C2;
    EXEC SQL FETCH C2 INTO :DCLSYSTABLES.NAME,
                           :DCLSYSTABLES.CARD,
                           :DCLSYSTABLES.CREATOR;
    IF SQLCODE = 1ØØ THEN DO;
      EXEC SQL CLOSE C2;
      DBREOR3O.PORUKAO = ' NO DATA FOR GIVEN CONDITION  - SYSTABLES';
      EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
      EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
    END;
    DO WHILE (SQLCODE = Ø);
      DCLSYSINDEXES.TBNAME = DCLSYSTABLES.NAME;
      DCLSYSINDEXES.TBCREATOR = DCLSYSTABLES.CREATOR;
      EXEC SQL SELECT COUNT(*)
               INTO :BRIND
               FROM SYSIBM.SYSINDEXES
               WHERE TBCREATOR = :DCLSYSINDEXES.TBCREATOR AND
```

```
                   TBNAME = :DCLSYSINDEXES.TBNAME;
    IF DCLSYSTABLES.CARD < Ø THEN DO;
      EXEC SQL CLOSE C2;
      DBREOR3O.PORUKAO = ' RUNSTATS IS NEEDED FOR: ' || DBN ||
                         '.' || TSN;
      EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
      EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
    END;
    ELSE POMPOLJE1 = DCLSYSTABLES.CARD;
    POMPOLJE2 = BRIND;
    PR31 = PR31 + POMPOLJE1 * POMPOLJE2;
    MAXKEY1 = Ø;
    DCLSYSCOLUMNS.TBNAME     = DCLSYSTABLES.NAME;
    DCLSYSCOLUMNS.TBCREATOR  = DCLSYSTABLES.CREATOR;
    EXEC SQL DECLARE C3 CURSOR FOR
      SELECT LENGTH, NULLS, COLTYPE
      FROM SYSIBM.SYSCOLUMNS
      WHERE TBNAME = :DCLSYSCOLUMNS.TBNAME AND
            TBCREATOR = :DCLSYSCOLUMNS.TBCREATOR AND
            KEYSEQ > Ø;
    EXEC SQL OPEN C3;
    EXEC SQL FETCH C3 INTO :DCLSYSCOLUMNS.LENGTH,
                           :DCLSYSCOLUMNS.NULLS,
                           :DCLSYSCOLUMNS.COLTYPE;
    DO WHILE (SQLCODE = Ø);
      POMPOLJE3 = DCLSYSCOLUMNS.LENGTH;
      POMMAXKEY = POMPOLJE3;
      IF DCLSYSCOLUMNS.NULLS = 'Y' THEN POMMAXKEY = POMMAXKEY + 1;
      IF SUBSTR(DCLSYSCOLUMNS.COLTYPE, 1, 3) = 'VAR'
        THEN POMMAXKEY = POMMAXKEY + 2;
      MAXKEY1 = MAXKEY1 + POMMAXKEY;
      EXEC SQL FETCH C3 INTO :DCLSYSCOLUMNS.LENGTH,
                             :DCLSYSCOLUMNS.NULLS,
                             :DCLSYSCOLUMNS.COLTYPE;
    END;
    EXEC SQL CLOSE C3;
    IF MAXKEY1 > MAXKEY THEN MAXKEY = MAXKEY1;
    EXEC SQL FETCH C2 INTO :DCLSYSTABLES.NAME,
                           :DCLSYSTABLES.CARD,
                           :DCLSYSTABLES.CREATOR;
  END;
  EXEC SQL CLOSE C2;
  PR31 = PR31 * (12 + MAXKEY);
END;
ELSE DO;
  DCLSYSINDEXPART.IXNAME = TSN;
  DCLSYSINDEXPART.IXCREATOR = DBN;
  DCLSYSINDEXPART.PARTITION = KZPARTITION(REDSCR);
  EXEC SQL SELECT CARD
             INTO :DCLSYSINDEXPART.CARD
             FROM SYSIBM.SYSINDEXPART
             WHERE PARTITION = :DCLSYSINDEXPART.PARTITION AND
                   IXNAME = :DCLSYSINDEXPART.IXNAME AND
                   IXCREATOR = :DCLSYSINDEXPART.IXCREATOR;
  IF DCLSYSINDEXPART.CARD < Ø THEN DO;
    DBREOR3O.PORUKAO = ' RUNSTATS IS NEEDED FOR: ' || DBN ||
                       '.' || TSN;
    EXEC CICS SEND MAP('DBREOR3') MAPSET('DBREORM');
    EXEC CICS RETURN COMMAREA(KZ) TRANSID('DBRG');
  END;
  POMPOLJE4 = DCLSYSINDEXPART.CARD;
  PR31 = PR31 + POMPOLJE4;
  EXEC SQL DECLARE C4 CURSOR FOR
    SELECT A.LENGTH, A.NULLS, A.COLTYPE
    FROM SYSIBM.SYSCOLUMNS A, SYSIBM.SYSINDEXES B,
         SYSIBM.SYSINDEXPART C
    WHERE A.TBNAME = B.TBNAME AND
          A.TBCREATOR = B.TBCREATOR  AND
          B.NAME = C.IXNAME AND
          B.CREATOR = C.IXCREATOR  AND
          A.KEYSEQ > Ø AND
          C.PARTITION = :DCLSYSINDEXPART.PARTITION AND
          C.IXNAME = :DCLSYSINDEXPART.IXNAME AND
          C.IXCREATOR = :DCLSYSINDEXPART.IXCREATOR;
  EXEC SQL OPEN C4;
  EXEC SQL FETCH C4 INTO :DCLSYSCOLUMNS.LENGTH,
                         :DCLSYSCOLUMNS.NULLS,
                         :DCLSYSCOLUMNS.COLTYPE;
  DO WHILE (SQLCODE = Ø);
    POMPOLJE5 = DCLSYSCOLUMNS.LENGTH;
    POMMAXKEY = POMPOLJE5;
    IF DCLSYSCOLUMNS.NULLS = 'Y' THEN POMMAXKEY = POMMAXKEY + 1;
    IF SUBSTR(DCLSYSCOLUMNS.COLTYPE, 1, 3) = 'VAR'
      THEN POMMAXKEY = POMMAXKEY + 2;
    MAXKEY = MAXKEY + POMMAXKEY;
    EXEC SQL FETCH C4 INTO :DCLSYSCOLUMNS.LENGTH,
                           :DCLSYSCOLUMNS.NULLS,
                           :DCLSYSCOLUMNS.COLTYPE;
  END;
  EXEC SQL CLOSE C4;
  PR31 = PR31 * (12 + MAXKEY);
END;
PR31 = PR31 / 16384;
PR32 = PR31;
DO I = 1 TO 15;
  IF SUBSTR(PR32, I, 1) ¬= ' ' THEN PR3 = PR3 || SUBSTR(PR32, I, 1);
END;
PR32 = PR31 / 1Ø;
DO I = 1 TO 15;
  IF SUBSTR(PR32, I, 1) ¬= ' ' THEN PR4 = PR4 || SUBSTR(PR32, I, 1);
```

```
END;
PR32 = 2 * PR31 / 4;
DO I = 1 TO 15;
  IF SUBSTR(PR32, I, 1) ¬= ' ' THEN PR1 = PR1 || SUBSTR(PR32, I, 1);
END;
PR32 = (2 * PR31 / 4) / 1Ø;
DO I = 1 TO 15;
  IF SUBSTR(PR32, I, 1) ¬= ' ' THEN PR2 = PR2 || SUBSTR(PR32, I, 1);
END;
IF INDTS THEN DO;
  DCLSYSTABLEPART.PARTITION = KZPARTITION(REDSCR);
  DCLSYSTABLEPART.TSNAME    = TSN;
  DCLSYSTABLEPART.DBNAME    = DBN;
  EXEC SQL SELECT CARD
           INTO :DCLSYSTABLEPART.CARD
           FROM SYSIBM.SYSTABLEPART
           WHERE PARTITION = :DCLSYSTABLEPART.PARTITION AND
                 TSNAME    = :DCLSYSTABLEPART.TSNAME AND
                 DBNAME    = :DCLSYSTABLEPART.DBNAME;
  EXEC SQL DECLARE C5 CURSOR FOR
    SELECT NAME, CREATOR
    FROM SYSIBM.SYSTABLES
    WHERE TSNAME = :DCLSYSTABLES.TSNAME AND
          DBNAME = :DCLSYSTABLES.DBNAME AND
          TYPE   = 'T';
  EXEC SQL OPEN C5;
  EXEC SQL FETCH C5 INTO :DCLSYSTABLES.NAME,
                         :DCLSYSTABLES.CREATOR;
  DO WHILE (SQLCODE = Ø);
    MAXTAB1 = Ø;
    DCLSYSCOLUMNS.TBNAME     = DCLSYSTABLES.NAME;
    DCLSYSCOLUMNS.TBCREATOR  = DCLSYSTABLES.CREATOR;
    EXEC SQL DECLARE C6 CURSOR FOR
      SELECT LENGTH, NULLS, COLTYPE
      FROM SYSIBM.SYSCOLUMNS
      WHERE TBNAME = :DCLSYSCOLUMNS.TBNAME AND
            TBCREATOR = :DCLSYSCOLUMNS.TBCREATOR;
    EXEC SQL OPEN C6;
    EXEC SQL FETCH C6 INTO :DCLSYSCOLUMNS.LENGTH,
                           :DCLSYSCOLUMNS.NULLS,
                           :DCLSYSCOLUMNS.COLTYPE;
    DO WHILE (SQLCODE = Ø);
      POMPOLJE6 = DCLSYSCOLUMNS.LENGTH;
      POMMAXTAB = POMPOLJE6;
      IF DCLSYSCOLUMNS.NULLS = 'Y' THEN POMMAXTAB = POMMAXTAB + 1;
      IF SUBSTR(DCLSYSCOLUMNS.COLTYPE, 1, 3) = 'VAR'
        THEN POMMAXTAB = POMMAXTAB + 2;
      MAXTAB1 = MAXTAB1 + POMMAXTAB;
      EXEC SQL FETCH C6 INTO :DCLSYSCOLUMNS.LENGTH,
                             :DCLSYSCOLUMNS.NULLS,
                               :DCLSYSCOLUMNS.COLTYPE;
      END;
      EXEC SQL CLOSE C6;
      IF MAXTAB1 > MAXTAB THEN MAXTAB = MAXTAB1;
      EXEC SQL FETCH C5 INTO :DCLSYSTABLES.NAME,
                             :DCLSYSTABLES.CREATOR;
    END;
    EXEC SQL CLOSE C5;
    POMPOLJE7 = DCLSYSTABLEPART.CARD;
    PR51 = POMPOLJE7 * (28 + MAXTAB);
    PR51 = PR51 / 16384;
    PR32 = PR51;
    DO I = 1 TO 15;
      IF SUBSTR(PR32, I, 1) ¬= ' '
        THEN PR5 = PR5 || SUBSTR(PR32, I, 1);
    END;
    PR32 = PR51 / 1Ø;
    DO I = 1 TO 15;
      IF SUBSTR(PR32, I, 1) ¬= ' '
        THEN PR6 = PR6 || SUBSTR(PR32, I, 1);
    END;
  END;
END OBRADA;
NULLMAP: PROC(@PTR, @LEN);
  DCL @PTR PTR, @LEN FIXED BIN(15),
      @DUMM CHAR(256) BASED(@PTR);
  SUBSTR(@DUMM, 1, @LEN) = LOW(@LEN);
END NULLMAP;
GRESKA: PROC(OUTMSG);
  DCL OUTMSG CHAR(8Ø);
  EXEC CICS SEND FROM(OUTMSG) ERASE;
  EXEC CICS RETURN;
END GRESKA;
SQLERR:
  CHAR_SQLCODE = SQLCODE;
  KMSG = 'SQLERROR: ' || CHAR_SQLCODE;
  EXEC CICS SEND FROM(KMSG) ERASE;
  EXEC CICS RETURN;
END DB2REORG;
```

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

Tracing in an open transaction environment – hints and tips

INTRODUCTION

CICS dump and trace formatting options can be very useful when investigating transaction throughput and task activity on a busy CICS system. In particular, exploitation of the Open Transaction Environment (OTE), as introduced in CICS Transaction Server 1.3, can increase the need to exploit these options.

BACKGROUND TO OTE

CICS support for OTE was introduced in CICS Transaction Server 1.3. It allows the user to run certain types of program environment under their own TCB in the CICS address space. This frees up work from the CICS quasi-reentrant (QR) TCB, which is the workhorse TCB in the CICS address space and is used for subdispatching traditional CICS workloads.

The OTE-managed TCBs are known as open TCBs. They run truly independently of the CICS QR TCB. They are dispatched separately by the MVS dispatcher and may execute in parallel with the QR TCB, when being concurrently dispatched on different Central Processors (CPs) by the hardware/operating system. Programs running under their own TCB may invoke functions or operations that could lead to the use of MVS services suspending the TCB. This is known as ‘blocking’. Isolating the program in its own open TCB prevents any effect of blocking the TCB for some period from impacting on other applications running in the CICS system at that time, under either the QR TCB or their own open TCBs.

Support for OTE has been phased across a number of CICS releases. With the initial general availability of CICS Transaction Server 1.3, OTE supported only JVM programs that interpreted

Java class files of bytecodes. Subsequent to this, PTF UQ44033 enhanced OTE support in CICS Transaction Server 1.3 by providing an open TCB environment for HPJ-compiled Java programs as well (‘hot-pooling’ support). With CICS Transaction Server 2.2, OTE now also supports a new mode of open TCB, for OPENAPI TRUE programs. The DB2 RMI Attachment facility for CICS has been enhanced in CICS Transaction Server 2.2 to allow exploitation of OTE to process requests from CICS to DB2 6.1 systems (and higher). For connections to DB2 5.1 systems (and below), privately managed TCBs are still used to perform the requests via the CICS Transaction Server 2.2 Adapter.

OPENAPI TRUE programs execute under an L8 OTE open TCB mode. JVM programs execute under a J8 OTE open TCB mode. Hot-pooled programs execute under an H8 OTE open TCB mode. Note: Java hot-pooling support in CICS Transaction Server 2.2 is retained for application migration purposes from CICS Transaction Server 1.3. CICS Transaction Server 2.2 supports a reusable JVM environment, offering significant performance improvements over the original JVM support provided within CICS Transaction Server 1.3. The JVM environment is the strategic platform for Java programs running in CICS Transaction Server 2.2.

For all types of OTE open TCB, their selection, allocation, and termination is managed by CICS. The different forms of OTE open TCB mode are very specific to the type of program environment they are intended to support. As such, the particular type of open TCB mode that is allocated for a particular program environment (L8 for an OPENAPI TRUE program, J8 for a JVM, H8 for a Java hot-pooled program object) is determined by CICS. It cannot be explicitly specified or overridden by the system programmer or the application program itself. Also, it is not possible to exploit OTE for program environments other than those outlined above. This is because CICS has to provide the internal functionality to support each mode of OTE operation, and at present only these three variants of OTE environment are supported.

CONCURRENT TASK ACTIVITY WITHIN CICS

Before the advent of OTE, CICS task activity was predominantly dispatched under a single MVS TCB (QR). This meant that an analysis of the tasks within a busy CICS system would rarely show more than one task actually running at any one time. In CICS dispatching terms, a 'running' state indicates that the task is dispatched and executing code, rather than waiting for redispatch, or waiting for an event such as an ECB to be posted, for example.

For there to be more than one task running concurrently would indicate that some work was being performed under another TCB; this would typically have been the FO TCB (for CICS File Control open and close operations), the RO TCB (for resource management such as security calls) or the CO TCB (for concurrent operations such as I/O activity if SUBTASKS=YES were specified in the SIT). Other TCBs existed for use on FEPI operations, for ONC/RPC work, etc.

With the introduction of Java application programming support in CICS TS 1.3, Java classes can now be interpreted (under a JVM) or executed (in a hot-pooled environment) under their own J8 or H8 TCBs in parallel with work being driven under the QR TCB. In a multiprocessor environment, this will mean that a number of tasks can be seen to be concurrently in a running state within the CICS system. An example view of part of the KE_TASK summary from a CICS system running a workload of JVM1 and HPJ1 transactions is shown below:

```
 ===KE: Kernel Domain KE_TASK Summary

KE_NUM  KE_TASK    STATUS          TCA_ADDR   TRAN_#    TRANSID
0051    123D7080   ***Running**    11CC9080   13101     JVM1
0052    123BA080   Not Running     11CD2080   13102     HPJ1
0053    123BAB00   ***Running**    11CC8080   12804     JVM1
0054    12521780   ***Running**    11CC4080   12805     JVM1
0055    11CFB400   ***Running**    11CC7680   13103     HPJ1
0056    11CFC400   ***Running**    11CE1080   13104     HPJ1
0057    11CFB780   Not Running     11CD1280   12806     JVM1

0058    11CFDB00   ***Running**    11CB4080   13108     HPJ1
0059    11CFE780   ***Running**    11CB6080   13109     HPJ1
```

In this example, JVM1 transids denote JVM programs; HPJ1 transids denote hot-pooled Java programs. As can be seen, a number of tasks are shown in a running state. (Note: HPJ-compiled CICS applications written in Java do not have to execute in a hot-pooled environment. ‘Ordinary’ HPJ support is provided too – such applications run as normal tasks under the QR TCB and compete for being dispatched along with other tasks within the CICS system. As with hot-pooling, ordinary HPJ support is retained for compatibility and application migration purposes in CICS Transaction Server 2.2.)

TRACE INTERPRETATION

CICS trace is the primary tool for problem determination in a CICS system; it is also extremely useful in determining the sequence of events that have led up to a particular event or failure occurring. There are some subtleties to consider when interpreting CICS trace data from a system exploiting the OTE environment, however; these are discussed below.

Below is an edited example of a trace from a ‘traditional’ CICS system:

```
231Ø1 QR    AP ØØE1 EIP   ENTRY READNEXT                           =ØØØØ64=
231Ø1 QR    AP D5ØØ UEH   EVENT LINK-TO-USER-EXIT-PROGRAM          =ØØØØ65=
231Ø1 QR    AP D5Ø1 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM      =ØØØØ66=
231Ø1 QR    AP Ø4EØ FCFR  ENTRY READ_NEXT_INTO                     =ØØØØ67=
231Ø1 QR    DS ØØØ4 DSSR  ENTRY WAIT_MVS                 FCIOWAIT  =ØØØØ68=
22921 QR    DS ØØØ5 DSSR  EXIT  WAIT_MVS/OK                        =ØØØØ69=
22921 QR    DS ØØØ4 DSSR  ENTRY WAIT_MVS                 ØØØØ8F38  =ØØØØ7Ø=
23Ø57 QR    DS ØØØ5 DSSR  EXIT  WAIT_MVS/OK                        =ØØØØ71=
23Ø57 QR    LG Ø2Ø1 LGGL  ENTRY WRITE                    ØØØØØØ5A  =ØØØØ72=
23Ø57 QR    LG 2ØØ1 L2LB  ENTRY GL_WRITE                 DFHJØ7    =ØØØØ73=
23Ø57 QR    KE Ø1Ø1 KETI  ENTRY ADJUST_STCK_TO_LOCAL               =ØØØØ74=
23Ø57 QR    KE Ø1Ø2 KETI  EXIT  ADJUST_STCK_TO_LOCAL/OK            =ØØØØ75=
23Ø57 QR    LG 2ØØ2 L2LB  EXIT  GL_WRITE/OK              ØØØØ16ØE  =ØØØØ76=
23Ø57 QR    LG Ø2Ø2 LGGL  EXIT  WRITE/OK                           =ØØØØ77=
```

The format of the trace entries is as follows. Column 1 shows the task number, column 2 the TCB under which the task is executing, columns 3 and 4 the CICS domain (ie subcomponent) and unique trace point within that domain, column 5 the suffix of the CICS module being executed, column 6 the variable data associated with the operation being traced, and finally column 7 shows the unique trace entry number within the CICS trace table of entries. In the example trace data, the first entry:

```
231Ø1 QR    AP ØØE1 EIP   ENTRY READNEXT                           =ØØØØ64=
```

can therefore be interpreted as follows. Task number 23101 was executing under the QR TCB. It caused trace point AP 00E1 to be issued from DFHEIP (the CICS EXEC interface program which handles CICS commands issued from applications). The type of command was an EXEC CICS READNEXT. This was trace entry 000064 within the trace table.

In the example, tasks 23101, 22921, and 23057 were running within CICS at the time the trace was taken. They were executing sequentially under the QR TCB. CICS subdispatched the tasks on this TCB, so each was getting the opportunity to run in turn. This is multi-tasking, not true concurrency. It is the speed of the CICS Dispatcher domain in juggling them which gives the impression that all the tasks are executing at once. In fact, each would run for a short while, then wait whilst performing an operation that involved a delay of some sort. When this occurred, the CICS Dispatcher domain switched control to another task that was ready to run at that point. These task switches are denoted by the DS domain trace points seen in the example.

It should be noted that the interleaving between tasks is not necessarily delimited by DS trace entries like this. The example traces given in this article were produced on CICS systems with all the standard levels of CICS component tracing set to 1 under CETR. No trace components were suppressed. In production environments, customers may elect to deactivate certain trace components within CICS in an attempt to improve system performance. This is fine until a problem occurs that requires trace analysis. If the resulting trace is inadequate because of the deactivated trace components, the problem may need to be recreated with all trace components set back on in order to resolve it. This defeats the concept of First Failure Data Capture (FFDC). Worse still is the case when a problem occurs and all levels of trace components have been set to off. The trace table will contain only CICS exception trace entries in such a situation; while these can be extremely useful in problem determination, they do not contribute the useful chronological sequence of events that preceded a failure, as provided by standard trace entries.

From analysing the entries in the example above, it is clear, by looking down the TCB column in the trace entries, that all task activity is taking place under the QR TCB. OTE is not being used; there are no L8 or J8 TCBs present in the system at the time, and MVS is not executing other CICS-managed TCBs (such as the CO or RO TCBs) during this period of activity.

Now consider the trace entries shown below:

```
Ø131Ø QR    AP 252Ø ERM   ENTRY CALL-TO-TRUE(DSNCSQL )        =Ø41335=
Ø131Ø QR    US Ø4Ø1 USXM  ENTRY INQUIRE_TRANSACTION_USER      =Ø41336=
Ø131Ø QR    US Ø4Ø2 USXM  EXIT  INQUIRE_TRANSACTION_USER/OK   =Ø41337=
Ø131Ø QR    RM Ø3Ø1 RMLN  ENTRY ADD_LINK             RMI      =Ø41338=
Ø131Ø QR    RM Ø3Ø2 RMLN  EXIT  ADD_LINK/OK                   =Ø41339=
Ø131Ø QR    DS ØØØ2 DSAT  ENTRY CHANGE_MODE          L8       =Ø4134Ø=
Ø131Ø L8ØØ5 DS ØØØ3 DSAT  EXIT  CHANGE_MODE/OK                =Ø41341=
Ø131Ø L8ØØ5 AP 318Ø D2EX1 ENTRY APPLICATION REQUEST EXEC SQL  =Ø41342=
Ø131Ø L8ØØ5 AP 325Ø D2D2  ENTRY DB2_API_CALL                  =Ø41343=
Ø131Ø L8ØØ5 AP 3251 D2D2  EXIT  DB2_API_CALL/OK               =Ø41344=
```

This shows an (edited) example of a trace from a CICS Transaction Server 2.2 system. Task 01310 is in control, executing under the QR TCB. The user program being run by this task issued an EXEC SQL call to invoke a DB2 subsystem. CICS was invoked by the SQL request from the application. It passed control to its External Resource Manager program DFHERM. Since this was a DB2 6.1 system, CICS transferred execution of the program from the QR TCB to an OPENAPI TRUE OTE-managed open TCB. The prefix for such OTE TCBs is L8; in this case, the internally-incremented alphanumerical suffix value for the TCB identifier was 005. The resultant TCB name of L8005 may be seen in the second column of the trace data.

Just by scanning down the trace entries in this manner, and simply looking at such TCB switches to find given task numbers, it is easy to identify the paths within the execution of a task that may require further investigation when trying to analyse system activity whilst exploiting OTE and open TCBs within CICS.

The situation gets more complex when a CICS system is actively exploiting OTE and has many such tasks running under open TCBs within the system. In such an environment, the trace table can be more difficult to interpret. An (edited) example trace demonstrating this is given below:

```
ØØ152 J8ØØ3 AP 21EØ JCICS EXIT  DTCSupport_MakeJavaString     =17393Ø=
ØØ149 J8ØØØ AP 21EØ JCICS ENTRY DTC_Init                      =173931=
ØØ149 J8ØØØ AP ØØE1 EIP   ENTRY ADDRESS                       =173932=
ØØ149 J8ØØØ AP ØØE1 EIP   EXIT  ADDRESS               OK      =173933=
ØØ152 J8ØØ3 AP 21EØ JCICS ENTRY DTCTerminal_getCommonData     =173934=
ØØ149 J8ØØØ AP 21EØ JCICS EXIT  DTC_Init                      =173935=
ØØ152 J8ØØ3 AP ØØE1 EIP   ENTRY ASSIGN                        =173936=
ØØ152 J8ØØ3 AP ØØE1 EIP   EXIT  ASSIGN                OK      =173937=
ØØ149 J8ØØØ AP 21E6 JCICS ENTRY Wrapper_callUserClass HW      =173938=
ØØ149 J8ØØØ AP 21EØ JCICS ENTRY DTCTask_getCommonData         =173939=
ØØ152 J8ØØ3 AP 21EØ JCICS ENTRY DTCSupport_ByteArrayAlloc     =17394Ø=
ØØ152 J8ØØ3 AP 21EØ JCICS EXIT  DTCSupport_ByteArrayAlloc     =173941=
ØØ152 J8ØØ3 AP 21EØ JCICS EXIT  DTCTerminal_getCommonData     =173942=
ØØ15Ø J8ØØ1 AP 21EØ JCICS ENTRY DTCTerminal_SEND_1CONTROL     =173943=
ØØ152 J8ØØ3 AP 21EØ JCICS EXIT  DTCTask_ProcessPrincipalFac   =173944=
ØØ149 J8ØØØ AP ØØE1 EIP   ENTRY ASSIGN                        =173945=
ØØ151 J8ØØ2 AP 21EØ JCICS EXIT  DTCSupport_MakeJavaString     =173946=
ØØ151 J8ØØ2 AP 21EØ JCICS ENTRY DTCSupport_MakeJavaString     =173947=
ØØ151 J8ØØ2 AP 21EØ JCICS EXIT  DTCSupport_MakeJavaString     =173948=
ØØ15Ø J8ØØ1 AP ØØE1 EIP   ENTRY SEND-CONTROL                  =173949=
ØØ151 J8ØØ2 AP 21EØ JCICS ENTRY DTCTerminal_getCommonData     =17395Ø=
ØØ15Ø J8ØØ1 DS ØØØ2 DSAT  ENTRY CHANGE_MODE           QR      =173951=
ØØ15Ø QR    DS ØØØ3 DSAT  EXIT  CHANGE_MODE/OK                =173952=
ØØ15Ø QR    AP FDØ1 ZARQ  ENTRY APPL_REQ   34AA4BØ,WAIT       =173953=
```

Here, four different tasks (00149, 00150, 00151, and 00152) are executing Java programs within JVM environments under CICS. Each task has its own open TCB managed by OTE. These are TCBs J8000, J8001, J8002, and J8003. In this example, task 00150 is subsequently switched back to execute under the QR TCB by CICS; this is because it has issued an EXEC CICS SEND CONTROL command to BMS, which is not defined as threadsafe and so cannot be executed by CICS under an open TCB environment. It is required to be executed under the QR TCB.

In order to improve the readability of such trace entries, the CICS auxiliary trace formatter program may be used to selectively return trace entries for specific task numbers. For example, DFHTU620 could be run against a CICS Transaction Server 2.2

auxiliary trace, specifying options of ABBREV and TASKID=(00150), in order to return only abbreviated trace entries for task 00150. A similar formatting operation for internal trace entries may be performed by the CICS system dump formatter. Again, as an example, a system dump could be formatted under IPCS by a command such as:

```
VERBX DFHPD62Ø 'TR=1,TRS=<TASKID=ØØ15Ø>'
```

in order to return abbreviated trace entries for task 00150.

The results of such a filtering technique are shown below:

```
ØØ15Ø J8ØØ1 AP 21EØ JCICS ENTRY DTCTerminal_SEND_1CONTROL     =173943=
ØØ15Ø J8ØØ1 AP ØØE1 EIP   ENTRY SEND-CONTROL                  =173949=
ØØ15Ø J8ØØ1 DS ØØØ2 DSAT  ENTRY CHANGE_MODE          QR       =173951=
ØØ15Ø QR    DS ØØØ3 DSAT  EXIT  CHANGE_MODE/OK                =173952=
ØØ15Ø QR    AP FDØ1 ZARQ  ENTRY APPL_REQ   34AA4BØ,WAIT       =173953=
```

If task 00150 were being investigated for some specific reason (for example, if it had abended later on and produced the dump and trace being analysed), then this technique makes problem analysis much simpler by cutting out unnecessary work in having to read through unrelated trace entries pertaining to different tasks in the CICS system.

Note that when formatting trace entries using the FULL option (as opposed to ABBREV), the actual address of the TCB is returned as well. This can be useful if further investigation of the program environment is required for whatever reason. Formatting the first trace entry from the example above, using the FULL option, shows (in edited form):

```
AP 21EØ JCICS ENTRY        TASK-ØØ15Ø TCB-J8ØØ1/ØØ8CAA68 RET-95ØECC3Ø
       TIME-Ø9:43:Ø6.65Ø8Ø1ØØ19 INTERVAL-ØØ.ØØØØØ82187       =173943=
```

Finally, it should be noted that on a system with several TCBs executing concurrently, if an error situation occurs at a particular point in time it may pertain to a program running on any of the TCBs. It is not valid to assume that the culprit was the program which produced the trace entries preceding the error. There is also the possibility that certain programs are executing on a TCB during the time period encompassed by the CICS trace table entries, and yet not causing CICS trace entries to be written during that time.

Andy Wright (andy_wright@uk.ibm.com)
CICS Change Team
IBM (UK)

CICS news

NEON Systems has announced support for the recently-available BEA WebLogic Enterprise Platform 8.1, an integrated platform that provides business integration through the convergence of application development and integration, from NEON's Shadow standards-based mainframe integration adapters

Shadow J2CA Connect and JDBC Connect adapters can simplify the task of accessing mainframes, providing J2EE-compliant access for developers and their tools to a broad range of mainframe data including ADABAS, DB2, IMS/DB, VSAM, and applications including CICS/TS and IMS/TM, without sacrificing the performance characteristics of the mainframe.

Shadow adapters provide direct, high volume, transactional integration to mainframe operational data stores and applications in support of mission-critical applications built on WebLogic Server.

For further information contact:
NEON Systems, 14100 Southwest Freeway Suite 500, Sugarland, TX 77478, USA.
Tel: (281) 491 4200.
URL: http://www.neonsys.com/news_&_Events/Press_Releases/2003/030804.asp.

* * *

CONNX Solutions has announced Version 8.9 of its CONNX product. CONNX facilitates access to data regardless of where it physically resides. CONNX enables organizations to unlock mainframe-based, relational, and desktop data sources so business decisions can be made. CONNX provides real-time read/write access to disparate data sources from a single point of connection as if it were a single database.

CICS users will be pleased to learn that the latest version supports access to VSAM files from batch jobs and started tasks as well as CICS. This allows CICS regions to be brought down for maintenance and access to still be available.

The CONNX Enterprise Server Service technology simplifies the deployment of CONNX in both small and large organizations. In cases where CONNX uses the vendor's native database driver to access data (for example, in Oracle or SQL Server databases), configuration and set-up of those drivers takes place on a single middle-tier server, instead of on each client PC. With the Enterprise Server Service, CONNX seamlessly provides a single client-install solution for data access needs throughout the enterprise.

CONNX has been further fine-tuned to provide faster performance when accessing flat-file data sources, such as C-ISAM running on Linux, AIX, HPUX, Sun, or Solaris operating systems.

CONNX gives businesses read/write real-time access to all enterprise data from any platform as if all the data existed in one relational database. This technology has been referred to as a federated database, or a virtual database. All data is then accessible using standard ANSI SQL and any standards-based application. CONNX acts as a reusable data access framework for projects throughout the enterprise.

For further information contact:
CONNX Solutions, 2039 152nd Avenue NE, Redmond, WA 98052, USA.
Tel: (425) 519 6600.
URL: http://www.connx.com/products/products.html.