CICS Web Interface - Templates In Memory, Improving performance and Management

Version 1.0.2

31 October 2000

Edward McCarthy
DBDC System Specalist
IBM GSA
Canberra
Australia

Take Note!

Before using this report be sure to read the general information under "Notices".

Second Edition, October 2000

This edition applies to Version 1.0.2 of *CICS Web Interface - Templates In Memory, Improving performance and Management* and to all subsequent releases and modifications unless otherwise indicated in new editions.

Contents

Notices

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates.

Information contained in this report has not been submitted to any formal IBM test and is distributed “asis”. The use of this information and the implementation of any of the techniques is the responsibility of the reader. Much depends on the ability of the reader to evaluate these data and project the results to their operational environment.

The performance data contained in this report was measured in a controlled environment and results obtained in other environments may vary significantly.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- CPSM
- CICS
- IBM
- REXX
- MVS/ESA
- OS/390
- SupportPac
- Transaction Server

Other company, product, and service names may be trademarks or service marks of others.

Summary of Amendments

Date	Changes
11 May 2000	Initial release
31 October 2000	Usage clarification added

Preface

To minimise the time taken to include HTML into a document when using the CICS DOCUMENT SET TEMPLATE API, the HTML can be loaded into a CICS User Maintained Data Table, a UMDT. This SupportPac explains how this is done using a set of programs developed by Edward McCarthy of IBM Australia.

This document is spilt into four chapters covering the following topics:

- Chapter 1 - HTML in Memory
- Chapter 2 - DFHWBEP (Web Error Handling Program)
- Chapter 3 - Helpful Programs for Encode Processing
- Chapter 4- CPSM Template Management

Bibliography

- *CICS Internet Guide,* SC34-5445
- *CICS Application Programming Reference,* SC33-1688

Chapter 1 - HTML in Memory

Introduction

HTML is stored as members in a Partitioned Data Sets, PDS. This allows application development staff to easily edit the HTML in the PDS using the standard ISPF editor. However if the CICS API that inserts templates has to access the HTML in a PDS, this will involve a relatively long process, as the data would be read from DASD.

By loading these HTML templates into memory, when the API inserts a template, the access will be to memory, providing the fastest access possible.

Two CICS regions are used to provide a CICS Web Interface environment. One region executes the actual CICS Web Interface code, while the second region is used to perform the process of loading the HTML into the UMDT. This second region is also used to dynamically refresh the HTML in the UMDT if the original member should be changed.

The second region is required as the process of loading and refreshing the HTML in the UMDT involves calling a program which performs dynamic allocation. The dynamic allocation process causes a wait in the region while it is processed, which would impact on any transactions that happened to be running at the time. By having this process in a second region, the other region that is running the CICS Web Interface is not affected.

A program is invoked during PLT whose task it is to load the HTML into the UMDT. This implies an order in the way in which these two regions are started. The region that loads the HTML needs to be started first, when it is has completed startup and all the HTML has been loaded, then the region that runs the CICS Web Interface can be started.

With the HTML stored in a UMDT, a program is needed to read the HTML from the UMDT when the API is used. The program that does this is the SCWIHRDR program. Each DOC type definition installed in the region, specifies the SCWIHRDR program as the exit program to be invoked. When the CICS DOCUMENT INSERT TEMPLATE API is called, CICS will check the DOC definition installed in the region for the template and invoke program SCWIHRDR.

All the REXX/CICS EXECs supplied in the SupportPac must be placed in a PDS under the CICEXEC DDname. For example, if they are placed in a PDS called MYNAME.CA8GPDS then the JCL should be as follows:

```
//CICAUTH  DD DISP=SHR,DSN=CICSTS13.REXX.SCICCMDS
//CICEXEC  DD DISP=SHR,DSN=CICSTS13.REXX.SCICEXEC
//         DD DISP=SHR,DSN=MYNAME.CA8GPDS
//CICUSER  DD DISP=SHR,DSN=CICSTS13.REXX.SCICUSER
```

User Maintained Data Table

This data table used to store the HTML requires a typical CICS file definition, for example:

```
 Name             SCWIHTML     Version        2
 Description      CWI - HTML - User Table - Test
 Time Stamps      Created: 1999/07/26 10:48   Changed: 1999/07/26 10:48
 User Data
VSAM Parameters
 Dsname        Data set name
               CICSTS13.CICS.RUNTIME.CWI.TEST.SCWIHTML
Password                    User access password
 Rlsaccess     NO           CICS opens files in RLS mode (YES,NO)
 Lsrpoolid     1            Local shared resource pool (1-8, NONE, blank)
 Readintegrity UNCOMMITTED Read level(UNCOMMITTED,CONSISTENT,REPEATABLE)
 Dsnsharing    ALLREQS      Dataset sharing (ALLREQS,MODIFYREQS)
 Strings       20           Concurrent file requests (1 - 255, blank)
 Nsrgroup                   Group name for VSAM data set
 Remote Attributes
  Remotename                   Remote file name
  RemoteSystem                 SYSIDENT for Remote System
 Remote and CF Datatable Parameters
  Recordsize      20000        Record size (1 - 32767, blank)
  Keylength       12           Key length  (1 - 255, blank)
                                           (1 - 16 for CF Tables)
 Initial Status
  Status          ENABLED      Status (ENABLED,DISABLED,UNENABLED)
  Opentime        FIRSTREF     Open time (FIRSTREF, STARTUP)
  Disposition     SHARE        File disposition (SHARE, OLD)

 NSR Buffers
  Databuffers     21           Number of data buffers (2-32767, blank)
  Indexbuffers    20           Number of index buffers (1-32767, blank)
 Datatable Parameters

  Table            USER        Data table type (NO, CICS, USER, CF)

  Maxnumrecs       5000        Max entries in data table ...
                                 (NOLIMIT or 1-99,999,999)

 CF Datatable Parameters

  CFDTpool                     Name of coupling facility data table pool

  Tablename                    Data table name

  Updatemodel      LOCKING     Update model (LOCKING or CONTENTION)

  Loadtype         NO          Whether file loads table (YES or NO)

 Record Format

  Recordformat     VARIABLE    Record format (VARIABLE, FIXED)
Operations

 Add               YES      Records can be added to file (YES,NO)

 Browse            YES      Records retrieved sequentially (YES,NO)

 Delete            YES      Records can be deleted (YES,NO)

 Read              YES      Records can be read (YES, NO)

 Update            YES      Records can be updated (YES,NO)


Auto Journalling

 Journal           NO       Journal number (NO, 1-99, blank)

 Jnlread           NONE     Read ops in jrnl (NONE,ALL,READONLY,UPDATEONLY)

 Jnlsyncread       NO       Auto journalling for read (YES,NO)

 Jnlupdate         NO       Rewrite/Delete oprs record on jrnl (YES,NO)

 Jnladd            NONE     Add ops recorded on jrnl(NONE,AFTER,ALL,BEFORE)

 Jnlsyncwrite      NO       Auto journalling for write (YES,NO)

 Recovery Parameters

  Recovery         NONE     Type of recovery (NONE,ALL,BACKOUTONLY)

  Fwdrecovlog      NO       Journal Name used for recovery (NO, 1-99, blank)

  Backuptype       STATIC   CICS VSAM file backup type (STATIC,DYNAMIC)

 Security

  Ressecnum                 Resource security value (0-24,PUBLIC,blank)
```

Programs

SCWIHREF

Language: Assembler

This program is the primary driver of the process to load HTML into the UMDT. It is used in two ways. The first way is from PLT to cause all HTML to be loaded into the UMDT. The second way is from a CICS transaction called SWHR, to allow for one of more HTML members in the UMDT to be refreshed.

When this program is invoked it determines if it has been called during PLT. If it has been called during PLT it will use the CICS API to determine all the DOC type definitions that are installed in the region. The name of each DOC definition is written to a temporary storage queue. The program then called a REXX program called SCWIHLDR. Note that the REXX program is invoked by building a commarea with the name of the REXX in it, and then linking to a program called CICREXR. The REXX program SCWIHLDR then controls the process of loading the HTML into the UMDT.

If the program was not called from PLT, it assumes that it is called via a CICS transaction. The transaction name is SWHR, and when used is invoked with a parameter specifying the names of the HTML members to be refreshed. For example typing:

```
SWHR CIRGA*
```

means that all HTML templates that start with the characters CIRGA are to be refreshed. Note there is a check in the REXX program which requires that at least three specific characters be entered, that is, SWHR * will not work.

When the program is called from the SWHR transaction, it still writes to temporary storage queue the names of all installed DOC templates to a temporary storage queue. The queue name and the mask value are passed to the REXX. When control returns from the REXX, the program displays a map showing the result of the refresh.

Note that when SWHR is run from a CICS region, it is not likely to be run in the same region that actually loaded the HTML into the UMDT. Typically the SWHR transaction would be run in a region that a user could logon to normally. Thus when the SCWIHREF program is invoked this way, it does not LINK to CICREXR, but rather LINKs to program SCICREXR. A program definition is required for SCICREXR which specifies that this a distributed program link to program CICREXR in another CICS region. Also, this implies that all the DOC definitions are installed into the region where SWHR is run, as when program SCWIHREF runs, it will be using the API to determine what DOC type definitions exist in the region.

Sample program defintion for SCICREXR is shown below:

```
Name                  SCICREXR          Version     1
                                                                More:     +
Description           CICS - Remote routing for REXX
Time Stamps           Created: 1999/06/09 15:53   Changed: 1999/07/26 12:16
User Data
Language              ASSEMBLER      ASSEMBLER,C,COBOL,LE370,LEVSE,PLI,RPG,N/A
Reload                NO             New copy of program loaded (NO, YES)
Resident              NO             Resident status (NO, YES)
Usage                 NORMAL         Storage release (NORMAL, TRANSIENT)
UseLPAcopy            NO             Program used from LPA/SVA (NO, YES)
Status                ENABLED        Program status (ENABLED, DISABLED)
Cedf                  YES            CEDF available (YES, NO)
Datalocation          ANY            Data location (BELOW, ANY)
Execkey               CICS           Program key (USER, CICS)
Executionset          FULLAPI        Program run mode (FULLAPI, DPLSUBSET)
Remotesystem          AW09           CICS region for shipped DPL request
Remotename            CICREXR        Program name in remote CICS region
Transid               SDIW           Tranid for remote CICS to attach
Rsl                                  Resource security value(0-24,PUBLIC,blank)
Dynamic               NO             Dynamic routing (NO, YES)
Concurrency           N/A            Concurrency (N/A, QUASIRENT, THREADSAFE)
JVM                   NO             Java Virtual Machine (NO, YES, DEBUG)
JVMClass                             Java Virtual Machine Class
Name                  SDIW           Version        1
 Description          CICS - CWI - Dyn Aloc via REXX
 Time Stamps          Created: 1998/09/09 12:35   Changed: 1998/09/09 12:35
 User Data
```

Sample transaction defintion for SDIW is shown below:

```
Program          DFHMIRS     Name program to process transaction
 Twasize         4096        Transaction work area size (0-32767, blank)
 Profile         RUIER5      Profile definition name
 Partitionset                Application partition set (name, KEEP, OWN)
 Status          ENABLED     Transaction status (ENABLED, DISABLED)
 Taskdataloc     ANY         Task storage location (BELOW, ANY)
 Taskdatakey     USER        Task storage key (USER, CICS)
 Storageclear    NO          Clear task life-time storage (YES, NO)
 Runaway         SYSTEM      Max tasktime  (SYSTEM, 0-2700000, blank)
 Shutdown        DISABLED    Status during shutdown (DISABLED, ENABLED)
 Isolate         YES         Isolate user storage (YES, NO)
```

Sample output from running the SWHR trans is show below:

```
------------------------ HTML Instorage Refresh ------------------------

 HTML Refresh Result:
      HICHR001I HTML Refresh for: ACSSNP*
      6 matches refreshed
      ACSSNPE1 ACSSNPE2 ACSSNPI1 ACSSNPI2 ACSSNPT1 ACSSNP12
```

SCWIHLDR

Language: REXX

This program expects to be passed a commarea. The commarea will contain the name of a temporary storage queue. The temporary storage queue contains the names of all templates installed in the region. The commarea will also contain a mask which in effect specifies what HTML members are to be refreshed.

The program reads through the temporary storage queue to determine if the name in the queue item matches the mask passed in the commarea. If the name is a match then the REXX performs the following logic:

- Call program SCWIMEMI to determine which dataset allocated to SCWIHTML contains the HTML member to be loaded
- Call program SCWITDQI to free TD Queue defn
- Call program SCWIDYNC to invoke program SCWIDYNI
    - Program SCWIDYNI dynamically allocates the dataset and member
- Call program SCWITDQI to dynamically define a TD Queue defnition which points to the DD dynamically allocated
- Uses a Do loop to read the HTML member from the TD Queue defn
    - The HTML is read into a buffer with a maximum length of 10000 bytes
    - Should the buffer fill, it is written to the UMDT
    - As many records of 10000 bytes as needed to store the HTML are written to the UMDT
- When all the HTML for that member has been read from the PDS, a header record is written to the UMDT. The header record specifies the total length of the HTML and how many records in the UMDT are used to store the HTML for that member.

The REXX program will load each HTML member into the UMDT that matches the mask specified in the passed commarea. If an error should occur, for example, there is no member in the PDS for a specified DOC definition, then this halts the loading process. The REXX program passes back information on the success of the HTML loading process to the calling program via the commarea.

SCWIHRDR

Language: Assembler

This program is invoked by CICS when the CICS DOCUMENT INSERT TEMPLATE API is called. The commarea passed to it is documented in the CICS Web Interface manual. The commarea passed will contain the name of the HTML template that is be inserted into the Document. This program will then use this name as a key to read the UMDT to locate the HTML that corresponds to this member.

SCWIMEMI

Language: Assembler

This program is called by the SCWIHLDR REXX. It is passed a commarea containing the name of PDS member to be searched for. This program assumes that the PDSs to be searched are allocated to DD SCWIHTML. The program builds a DCB for this DD card, then uses a BLDL macro call to locate the dataset that contains the specified member, if any. It returns via the commarea the name of the PDS containing the member.

SCWIDYNC

Language: Assembler

This program is called by the SCWIHLDR REXX program. It is passed a commarea containing information about a dataset that is be dynamically allocated. This program then 'calls' the program SCWIDYNI, note it does not EXEC CICS LINK to the SCWIDYNI program. It passes to SCWIDYNI the commarea it was passed.

SCWIDYNI

Language: Assembler

This program is passes a commarea containing information about a dataset to be dynamically allocated. Note the dataset information can be either for just a physical sequential dataset or for a member of a partitioned dataset. This program builds the control blocks necessary to perform dynamic allocation and then invokes SVC 99 to perform the dynamic allocation.

This program **MUST** be compiled as a normal Assembler language program (and **NOT** as a CICS command level program).

SCWITDQI

Language: Assembler

The REXX program needs to be able to dynamically allocate and unallocate TD Queue definitions. However REXX does not support the CICS API that perform this function. This program was written to perform this function. It is called by the SCWIHLDR REXX. The commarea passed contains information which specifies what is required. This program does the following:

Delete a TQ Queue defn
Define a TD Queue defn
Open a TQ Queue
Close a TD Queue

SCWICOND

Language: Assembler

This is a copy lib containing various Dsects used in the above programs.

SCWIMAP

Language: BMS Map

This is a BMS Map. It is used to display the results of a refresh request when the SWHR transaction is used to drive the SCWIHREF program.

Chapter 2 - DFHWBEP - Web Error Handling Program

The IBM supplied web error handling program, DFHWBEP, is invoked by the CICS Web Interface when it detects an error during processing of a request it has received. The supplied IBM sample produces a very basic message.

The version of DFHWBEP supplied here, an assembler program, provides a more informative display should an error occur. For example, if the user is not authorised to run a transaction, this version of DFHWBEP display a message to that effect. The displays from this version of DFHWBEP are not just a one line type error message, but rather a full HTML type page.

Chapter 3 - Helpful Programs for Encode Processing

The process of building a document to send back to a browser typically involves inserting HTML into the document and the setting of symbols and their values to allow for symbol substitution. It is not uncommon for these API calls to fail because for example the template is not defined or the symbol data passed on the API call is invalid.

To assist in development and debugging, the encode program I used for building documents would check the response code from the API calls. Where appropriate the encode program would link to a program called SCWICOEH.

This program would attempt to recover gracefully from API failures. If for example the API to call to insert a HTML TEMPLATE failed because there was no definition for the TEMPLATE installed in the region, then a program called SCWICDOC would be called to dynamically create a definition in the region for the missing template.

If the API call that failed was due to bad data passed for a symbol substituion call, then the SCWICOEH would build a HTML display showing the commarea passed to the encode routine. This HTML display would then be passed back to the encode program for it to display on the browser.

The code for SCWICOEH and SCWICDOC is supplied with this documentation.

Chapter 4 - CPSM Template Management

Introduction

CICS Transaction Server for OS/390 R1.3 supplied new APIs to allow the insertion of templates into documents. This, however, required that for each template being used, that a DOC type definition exist within the region. The DOC definition tells the CICS Web Interface how to access the template.

It does not take long for the number of templates to grow and this could become another definition that CICS system programmers need to define manually, like transaction definitions.

This section describes a process that was setup to automatically manage the doc definitions.

Overview

This description assumes that CPSM is being used to manage the DOC type definitions in the CICS environment.

There where four separate CICS environments involved referred to as Dev, Test, User and Production.

Dev was the environment used by application programming staff to develop and test their programs.

Test was the environment used by testing staff to test that the programs developed by the application staff met the requirements set out by the business area.

User was the environment used by the business staff to test the programs developed by applications, just prior to their promotion to production.

Production is the production environment.

Each environment has it’s own HTML dataset. For example, if a programmer created a HTML member called ABCDHOME, he would put it in the dev HTML PDS. It would then be copied to the test HTML PDS, then the user HTML PDS and finally the production HTML PDS.

Programs where written in REXX, which used the CPSM APIs, to automate the management of the DOC definitions in the CICS environment. Within CPSM, a Resgroup was defined to which all DOCDEFs would be connected. These DOCDEFs where then installed by CPSM into every CICS region via RASGNDEFs. Flags in the description area of each DOCDEF are used to control which DOCDEFs should be installed into the dev, test, user and production environments.

In the description area of each DOCDEF in CPSM, the first five bytes are set as follows:

Byte 1 - Always E

Byte 2 - Set to P if member exists in the prod HTML PDS

Byte 3 - Set to U if member exists in the user HTML PDS

Byte 4 - Set to T if member exists in the test HTML PDS

Byte 5 - Set to D if member exists in the dev HTML PDS

Every morning at approximately 4am, a batch job is run. This batch job runs TSO in batch and invokes a REXX called SCPSMDOC.

This REXX, SCPSMDOC, analyses the contents of each HTML PDS. It builds up a profile of each member, determining which PDS the member belongs to. The REXX then uses the CPSM APIs to determine what DOCDEFs are currently defined in CPSM. The REXX then reconciles the two environments. This may entail adding new DOCDEF definitions, updating the description flags in the DOCDEF definitions, or deleting a DOCDEF definition if the member no longer exists within any HTML PDS.

This batch job thus fully automates the maintenance of the DOCDEFs with the CICS environments.

Other REXX programs called by SCPSMDOC are:

SCPSMGTH - used to get a CPSM thread

SCPSMGDD - used to get DOCDEF names currently defined to CPSM

SCPSMDID - used to get DOCINDEFs currently defined to CPSM

SCPSMDDM - used to add and delete DOCDEFs and DOCINDEFs

-- End of Document ---