



CICS Transaction Server for z/OS



Migration from CICS Transaction Server version 1.3 to CICS Transaction Server version 2.2: A System Programmers view



Robert Harris,
CICS Technical Strategy,
IBM Hursley.

Issued:	06 April	2003
Revision Date:	06 April	2003
Previous Revision Date:	01 January	2003
Review Date:	As Required	

Take Note!

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

Third Edition, April 2003.

© **Copyright International Business Machines Corporation 2002**. All rights reserved. Note to US Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

Notices:

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

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

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks:

The following are Trademarks of International Business Machines Corporation in the United States, in other countries, or both:

3090	ACF/VTAM	AD/Cycle
AFP	AIX	AnyNet
Application System/400	APPN	AS/400
AT	BookManager	C Set++
C/370	C/MVS	CBIPO
CBPDO	CICS	CICS/400
CICS/6000	CICS/ESA	CICS/MVS
CICS OS/2	CICS TS	CICS/VM
CICS/VSE	CICSplex	CICSplex SM
COBOL/370	COBOL/2	Common User Access
CUA	DATABASE 2	DB2
DFSMS	DFSMS/MVS	DFSMSdfp
DFSMSdss	DFSMShsm	DFSORT
DXT	eNetwork	Enterprise Systems Architecture/370
Enterprise Systems Architecture/390	ES/3090	ESA/370
ESA/390	ES/9000	ESCON
GDDM	HiperBatch	Hiperspace
InfoWindow	IBM	IBMLink
IMS	IMS/ESA	Language Environment
MQ	MQSeries	MVS
MVS/DFP	MVS/ESA	MVS Parallel Sysplex
MVS/SP	MVS/XA	Multiprise
NetView	OpenEdition	OS/2
OS/390	OS/400	Processor Resource/Systems Manager
Parallel Sysplex	PR/SM	Presentation Manager
RACF	Resource Measurement Facility	RETAIN
RISC System/6000	RMF	RT
S/370	S/390	SAA
SQL/DS	SP	System/36
System/38	System/360	System/370
System/390	SystemView	Systems Application Architecture
VisualAge	VSE/ESA	VTAM
WebExplorer	z/OS	

UNIX is a registered Trademark in the United States and other countries licensed exclusively through X/Open Company Limited

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

INTEL is a registered trademark of Intel Corporation, in the United States, or other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, or other countries, or both.

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

Summary of amendments

Date Of Change	Change made to document
01/12/2002	Creation
01/01/2003	Added some more detail on SIT defaults and how DB2 Group Attach and Warm Restarts interact.
06/04/2003	Updated DB2CONN & DB2ENTRY with PRIORITY settings (introduced in APAR PQ67351)

Reference Material and Bibliography:

This document uses a short reference to the following documentation:

Short reference	Book Title
CICS RDO Book	CICS Resource Definition Guide SC34-5722
CICS SDG	CICS System Definition Guide SC34-5725
CICS CG	CICS Customization Guide SC34-5706
CICS EXT	CICS External Interfaces Guide SC33-1944
CICS JAVA	Java™ Applications in CICS SC34-6000-0
CICS APR	CICS Application Programming Reference SC34-5702
CICS SPR	CICS System Programming Reference SC34-5726
CICS RG	CICS Transaction Server for z/OS Release Guide GC34-5983
CICS MG	CICS Transaction Server for z/OS Migration Guide GC34-5984
JVM Book	New IBM Technology featuring Persistent Reusable Java Virtual Machines SC34-5881
LDAP Red Book	Understanding LDAP SG24-4986

Preface:

This document is aimed at CICS System Programmers who are migrating from CICS Transaction Server for OS/390 version 1.3 to CICS Transaction Server for z/OS version 2.2.

This document does not cover Java™ operation, Enterprise Java Bean™ function in CICS Transaction Server for z/OS, JDBC™ 2.0 activity or the use of a Lightweight Directory Access Protocol server.

You need a detailed knowledge of CICS configuration to get the best out of this document, and are probably a CICS Systems Programmer.

This document is structured in a less formal fashion than the CICS Transaction Server product documentation. You should always consider the latest official CICS TS publications as being correct in the cases where this document implies a mismatch.

The information and code in this document is **only** applicable to CICS Transaction Server for z/OS Version 2.2. It is not applicable to earlier CICS releases.

This document uses Colour to highlight items of interest, so access to the PDF as well as the hard copy in the absence of a colour print is desirable.

Table of Contents

Migration from CICS Transaction Server for OS/390 version 1.3 to CICS Transaction Server for z/OS version 2.2	1
Introduction	1
Documentation	1
Major Internal CICS changes	2
Implementation of JVMs for Java activity	2
TCBs for DB2	2
DB2 v7 and Group Attach	2
Other enhancements	2
Starting with a Conclusion	3
The Main areas of interest	3
Altering the CSD	4
Changing the RECSZ and Upgrading the CSD	4
Checking your copies of IBM Definitions	5
Sharing the CSD with CTS 1.3	5
My recommendation	5
RDO Definitional changes	6
CEDA and lowercase	6
REQUESTMODELS	7
Replaced parameters	7
New parameters	7
Example Definition	8
My Recommendation	8
DB2CONN and DB2ENTRY	9
DB2ENTRY	9
DB2CONN	9
DB2CONN and Group Attach	10
PROGRAM	11
CONCURRENCY Implications for TRUEs	11
The CONCURRENCY (QUASIRENT) setting for DB2 usage	12
The CONCURRENCY (THREADSAFE) setting for DB2 usage	12
Java definitions	13
TCPIPSERVICE	14
PROTOCOL	14
AUTHENTICATE	14
MVS Workload management	14
SSL Certificate for outgoing flows	14
Lists	15
TCPIPSERVICE requirement	15
TYPETERM	15
XRFSIGNOFF	15
Summary	16
DCT Migration	17
DFHCNV issues	18
Compiled Java (including IOP)	19
Converting HPJ programs	19

IOP Applications	19
SSL Certificates	20
JCL and File Changes	21
DFHJVM	21
DFHEJOS & DFHEJDIR	21
DFHGCD	21
DFHCJVM	21
Runtime DB2 libraries	22
Runtime Language Environment Libraries	22
MVS Unix System Services	22
Compilation JCL	22
EXEC CICS SIGNON change	23
The change	23
Temporary bypass	23
Migration planning	24
Finding application programs using the commands	24
Changing programs for transactions that run at terminals	24
Starting a non-terminal transaction with a given security identity	25
Starting a transaction that is to run next at the terminal with another security identity	25
Continuing a transaction at the terminal with a different security identity	26
Coding a signon transaction	26
Temporarily changing into Super-User mode at the terminal	27
Starting a transaction at another terminal with a different security identity from that currently being used at that terminal	27
Language Environment runtime issues	28
What I mean by Language Environment	28
OS/VS COBOL programs	29
JCL and runtime conclusion	29
Compilation	29
VS COBOL II programs	30
Either Mode compilers	30
How CICS decides to use a Language Environment runtime	31
Using a Supported non-Language Environment runtime	32
Using a Supported Language Environment runtime	32
Setting Language Environment Parameters	32
Language Environment parameters and program behaviour	33
Increase the DSA	33
Compilation	33
SIT Changes	34
Deleted parms	34
DCT	34
KEYFILE	34
MNEVE	34
XRFSoFF & XRFSTIME	34
Altered Defaults	35
LGDFINT	35
MNFREQ	35
STNTR & SPCTR	35
MAXOPENTCBS	35

New parameters for resources	36
MAXJVMTCBS	36
MAXHPTCBS	36
MAXSOCKETS	36
KEYRING	36
New parameters relating to intervals	37
RSTSIGNOFF & RSTSIGNTIME	37
STATEOD & STATINT	37
New parameters relating to things not in CTS 1.3	38
BRMAXKEEPTIME	38
AIBRIDGE	38
HIOPLISTENER	38
XEJB	38
EJBROLPRFX	38
Summary	39
Global User Exits	40
GLUEs and TCBs	40
Threadsafe coding	40
Migration Considerations	41
GLUE parmlist changes	41
File Control GLUEs	42
XFCREQ & XFCREQC	43
XFCFRIN & XFCFROUT	44
URMs	46
NEPs	46
Terminal Autoinstall	46
New URM functions	46
Non-migrational changes	47
API, SPI, CEMT and Supplied Transactions	47
RDO Objects	48
URMs	48
CPSM migration	49
CICS and DB2	50
DB2 and CICS TCBs	50
DB2 Purgeability	50
DB2 Group Attach	51
Warm restart considerations	51
DB2 and Threadsafe roadmap	52
Migration Conclusion	53

List of Figures

JCL for the CSD	4
CEDA multi-line fields	6
Example REQUESTMODEL for a CORBA object	8
PROGRAM definition for a Java Class	13
Load Module Scanner JCL	24
Non-terminal Transaction initiation.....	25
Same Terminal initiation	25
Continuing a Transaction.....	26
Local File Control GLUEs.....	44
Remote File Control GLUEs	44

Migration from CICS Transaction Server for OS/390 version 1.3 to CICS Transaction Server for z/OS version 2.2

Introduction

This document describes the migration from CICS Transaction Server for OS/390 version 1.3 to CICS Transaction Server for z/OS version 2.2 from the viewpoint of a CICS System Programmer.

The general philosophy of this text is to group associated things that have to be done together. This approach will not correspond with the way the CICS Migration Guide or other CICS TS Product Material is structured. When this text says something that conflicts with the CICS TS Product documentation - this text is wrong!

I am assuming that you, my reader, are a CICS Systems Programmer and have configured a CICS TS 1.3 region. I am going to describe the things you need to change to get a CICS TS 2.2 region going, but not go into the way new things (like the use of a Java Virtual Machine) are configured.

Documentation

- SC34-5722: *CICS Resource Definition Guide* is the main reference for this document
- SG34-5725: *CICS System Definition Guide* also has concepts which are relevant
- GC34-5984: *CICS Transaction Server for z/OS Migration Guide* is the formal reference for this document
- SG34-5983: *CICS Transaction Server for z/OS Release Guide* is also of interest
- SC34-6000: *Java™ Applications in CICS* is the reference for all Java operations in CICS

Contrary to most CICS Documentation, the next chapter starts with a migration conclusion. But before that, here is a brief overview of the major internal enhancements in CICS TS 2.2.

Implementation of JVMs for Java activity

Java Virtual Machines (JVMs) have been implemented in the CICS environment. As far as migration is concerned, this does not have an impact.

TCBs for DB2

In CTS 1.3 DB2 activity was performed using MVS sub-tasks under the QR TCB. In CTS 2.2, all DB2 activity runs under L8 TCBs. This means:

- There is a significant performance benefit for Threadsafe programs
- Some Resource Definitions have changed
- GLUE programs can be multiply running (on different TCBs)

Some presentations and associated documentation mentioned that DB2 activity is subject to Runaway Task control. After customer suggestions, this has now been changed so that DB2 activity is **not** subject to Runaway Task control in CTS 2.2 (the situation is as it is in CTS 1.3).

DB2 v7 and Group Attach

Group Attach (not available for DB2 v6) enables better DB2 availability for CICS. It enables CICS to connect to one-of-n DB2 instances so providing access when a DB2 instance fails.

Other enhancements

- VTAM Persistent Session support allows a terminal security identity to preserved over certain types of failure
- The VTAM Alias facility is supported and the Terminal Autoinstall URM can extract the Fully Qualified Network name
- CICS Data-Sharing address spaces will automatically restart after a failure
- Statistics & Monitoring have been updated to include new TCB usage

It is unusual to start of a migration document with a conclusion, but you need to know that you are not facing a great challenge to migrate from CTS 1.3 to CTS 2.2. Compared with the jump from CICS/ESA 4.1 to CICS TS 1.n - where use of the MVS logger complicated things - the 1.3 to 2.2 migration is uncomplicated.

The Main areas of interest

I'm going to start by outlining the areas of interest to you which this document addresses:

- The CSD record size increases
- All Java RDO has changed
- EXEC CICS SIGNON behaviour has changed
- Language Environment libraries need to be used
- Some SIT parms change
- The usual recompilations

You can see that this is not an hugely long list of things to worry about.

I am also going to include some information about usage of new DB2 releases.

It's an easy migration

Changing the RECSZ and Upgrading the CSD

The RECSZ (Record Length) of the DFHCSD file has increased to 2000 bytes. This is to accommodate extra-long RDO definitions for Java-related items (especially the REQUESTMODEL definition).

The first thing to do, therefore, is to AMS DEFINE a new 2.2 CSD with RECSZ (200 2000) and AMS REPRO the 1.3 CSD to the (newly defined) 2.2 CSD.

Next, run the DFHCSDUP UPGRADE operation on this new CSD. This will install all the new definitions required for CTS 2.2 and add a DFHCOMPn group for compatibility with CTS 1.3.

```
//CRECSD EXEC PGM=IDCAMS,REGION=0M
//SYSPRINT DD SYSOUT=*
//AMSDUMP DD SYSOUT=*
//SYSIN DD *
DEFINE CLUSTER (
    NAME( RHARRI1.IYCKRAH6.CSD22 ) -
    CYL(2,1) -
    VOLUME(SYSDAV) -
    KEYS( 22 0 ) -
    INDEXED -
    RECORDSIZE( 200 2000 ) -
    FREESPACE( 5 5 ) -
    SHAREOPTIONS( 3 3 ) -
)
INDEX
(
    NAME( RHARRI1.IYCKRAH6.CSD22.INDEX ) -
)
DATA
(
    NAME( RHARRI1.IYCKRAH6.CSD22.DATA ) -
)
/*
//COPYCSD EXEC PGM=IDCAMS,REGION=20M
//SYSPRINT DD SYSOUT=*
//CSD DD DSN=RHARRI1.IYCKRAH6.CSD22,DISP=SHR
//OLDCSD DD DSN=RHARRI1.IYCKRAH5.CSD13,DISP=SHR
//SYSIN DD *
REPRO INFILE(OLDCSD) OUTFILE(CSD) REUSE
/*
//UPGRCSDD EXEC PGM=DFHCSDUP,REGION=20M
//STEPLIB DD DSN=<<CICS22>>.SDFHLOAD,DISP=SHR
// DD DSN=<<CICS22>>.SDFHAUTH,DISP=SHR
//DFHCSD DD DSN=RHARRI1.IYCKRAH6.CSD22,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(100,10))
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSIN DD *
UPGRADE REPLACE
/*
```

Figure 1: JCL for the CSD

Checking your copies of IBM Definitions

There are some changes to the IBM supplied RDO definitions. If you have copies of the CICS-supplied definitions in your own groups, don't forget to do a DFHCSDUP `SCAN` operation to ensure these local copies are still valid.

Sharing the CSD with CTS 1.3

I'm not personally too keen on sharing the 2.2 CSD with an older (CTS 1.3) region due to the change in `RECSZ` and lots of changes to `REQUESTMODEL` definitions (see "REQUESTMODELS" on page 7). However, this is absolutely a RAHism and it shows my prejudices.

If you do not use (Compiled) Java in your CTS 1.3 region, then the case for sharing is stronger than if this is present. The length of time when running with two distinct CSDs has to be born in mind (keeping both up-to-date is more of a problem if the CSD is not shared).

However, if you have got to use the CSD in CTS 2.2 and also in CTS 1.3, you will have to change the Lists used in the CTS 1.3 region to include the relevant `DFHCOMPn` group (and use `CEDA` in Compatibility mode via PF2). This, in itself, introduces change into a stable CTS 1.3 region which may be undesirable. You should also ensure that all `CEDA` operations occur from the CTS 2.2 region.

You will also need to define distinct groups for `REQUESTMODELS` running in the CTS 1.3 environment and the CTS 2.2 environment and ensure that the Lists used to CTS 1.3 and CTS 2.2 use the correct groups.

My recommendation

Don't share a 2.2 CSD with a CTS 1.3 region.

Think hard before sharing the CSD with CTS 1.3

CEDA and lowercase

RDO now contains objects whose settings are multi-line and case dependant. These parameters occur when defining Unixy things:

- HFS
 - ◆ file names
 - ◆ directories
- Java
 - ◆ class names
 - ◆ jar names

In order to permit mixed case input, you have to do a `CEOT NOUC` to set the terminal into mixed case mode (and a `CEOT UC` afterwards to restore back to uppercase). Don't forget to use an uppercase `CEDA`!

An example of this occurs in Figure 2 for a `CORBASERVER` object where the `SHELF` parameter is multi-line and requires mixed case input.

```
CEDA DEFine CORbaserver( a      )
CORbaserver ==> a
Group       ==> T1
DEscription ==>
Jndiprefix  ==>
            ==>
            ==>
            ==>
            ==>
Autopublish ==> No          Yes | No
SEssbeantime ==> 00 , 00 , 10    0-99 (Days,Hours,Mins)
SHElf       ==> /var/cicsts/RAH
            ==>
            ==>
            ==>
            ==>
DJardir     :
+           :
```

Figure 2: CEDA multi-line fields

The `DJAR` parameter is also multi-line, but all of it cannot fit on the current panel. This is shown by the use of `:s` (colons) for the parameter. Use `PF7/PF8` to scroll until all of it is editable (back to the usual `==>` prompt).

Lower case is needed for Java-related RDO parms

REQUESTMODELS

REQUESTMODELS are used to select a CICS Transid for use when running a CORBA object or an EJB object. Migration of REQUESTMODELS is probably the biggest change of the migration.

You will not have got the REQUESTMODEL EJB flavour in your CTS 1.3 region, as they are not available therein. However, you may have a REQUESTMODEL active if you are using CORBA objects.

I'm not going to describe all the things you can use in this definition, only the ones which relate to a CORBA Operation (as this is a migration document!).

Replaced parameters

OMGINTERFACE, OMGMODULE and OMGOPERATION have been replaced with INTERFACE, MODULE and OPERATION settings. These are, basically, merely renames.

New parameters

The meaning of the new parameters is the same as in a 1.3 CSD:

MODULE	The IDL Module Name
INTERFACE	The name in IDL
OPERATION	The routine name

The combination of these settings (don't forget to use a trailing *) selects the Transaction Name under which the requested operation will execute.

I **strongly** recommend keeping these simple (just decide at the MODULE level).

If you are venturing into REQUESTMODELS for EJB, you will be using the INTFACETYPE, OPERATION and BEANNAME settings (only use BEANNAME to keep things simple).

Redo all REQUESTMODELS

Example Definition

Figure 3 shows a REQUESTMODEL definition for a CORBA Object: all CORBA operations are run under the CICS Transid of '2222'.

```
CEDA View Requestmodel( 2          )
Requestmodel      : 2
Group            : T1
Description       : EXAMPLE OMG REQUESTMODEL
Corbaserver      : CS2
TType           : Corba                Corba | Ejb | Generic
EJB PARAMETERS
Beanname         :
                :
INTFacetype      :                    Both | Home | Remote
CORBA PARAMETERS
Module          : *
                :
INTFacetype      :                    Both | Home | Remote
CORBA PARAMETERS
Module          : *
                :
INTERface       : *
                :
COMMON PARAMETERS
Operation       : *
                :
TRANSACTION ATTRIBUTES
TRansid         : 2222
CICS TS V1R3 ATTRIBUTES
OMGModule       :
OMGInterface    :
OMGOperation    :
```

Figure 3: Example REQUESTMODEL for a CORBA object

My Recommendation

It is a lot of pain to ensure that REQUESTMODELS are properly migrated. It is simpler to re-enter the settings manually than attempting to fiddle about in Compatibility mode.

Re-enter REQUESTMODELS

DB2CONN and DB2ENTRY

I am assuming that your CICS region is communicating with a DB2 v7 instance.

This section only comments on DB2 changes from the view of RDO migration. Please look at these sections for other DB2 information:

- “PROGRAM” on page 11
- “MAXOPENTCBS” on page 35
- “CICS and DB2” on page 50

Migration is concerned with changed definitions due to the new TCB structure used by CICS for DB2 access (see “TCBs for DB2” on page 2).

DB2ENTRY

Even though DB2 activity runs on L8 TCBs, the `PRIORITY` setting is still meaningful. It now refers to MVS dispatching priorities relative to the QR TCB (that used for most CICS-related activity). The L8 TCB used for DB2 activity can have a greater (`HIGH` setting), equal (`MEDIUM`) or lower (`LOW`) MVS dispatching priority relative to that used for the QR TCB.

DB2CONN

The `PRIORITY` setting on the `DB2CONN` has a similar meaning.

The `TCBLIMIT` parameter now has a different meaning. It controls the number of L8 TCBs that can be used for DB2 activity. If there are too few L8 TCBs available, DB2 activity will be constrained.

This number of L8 TCBs for DB2 usage comes out of the total number of L8 TCBs active within the CICS region: which is controlled by the `SIT MAXOPENTCBS` parameter (see “MAXOPENTCBS” on page 35).

Therefore, ensure that `SIT.MAXOPENTCBS > DB2CONN.TCBLIMIT`.

The number of L8 TCBs available in the CICS region can be changed using the `EXEC CICS/CEMT SET DISPATCHER` command.

SIT.MAXOPENTCBS > DB2CONN.TCBLIMIT

DB2CONN and Group Attach

A new parameter called `DB2GROUPID` is provided to control DB2 Group Attach facilities (see “DB2 v7 and Group Attach” on page 2). This is the only place whereby the Group Name used by DB2 instances can be specified (you cannot specify this Group via JCL).

If `DB2ID` is coded, this overrides `DB2GROUPID`, so turning off the Group Attach function.

When CICS is communicating with DB2 using Group Attach, the new `RESYNCMEMBER` setting controls how recovery proceeds:

<code>RESYNCMEMBER (YES)</code>	<p>This is the existing behaviour as in CTS 1.3.</p> <p>CICS will wait until the failed DB2 is available before allowing further DB2 access.</p> <p>Operations that were active on failure will have to be resolved before any further DB2 activity can proceed.</p>
<code>RESYNCMEMBER (NO)</code>	<p>CICS will make a single attempt at accessing the failed DB2 instance, and if that is not available contact another instance in the Group.</p> <p>New activity will proceed through the new DB2 instance.</p> <p>Operations that were active on failure will have to be manually resolved.</p>

As far as migration is concerned for Group Attach with CTS 1.3 behaviour (not exploiting the function), use `RESYNCMEMBER (NO)`.

To migrate to Group Attach with the new recovery characteristics use `RESYNCMEMBER (YES)`.

To use the Group Attach function do **not** use the `INITPARM=(DFHD2INI='db2_subsystem_id')` construct in the JCL.

The DB2 type of Attach (Group or Specific) is catalogued and is consequently restored over the CICS Warm restart process. Thus, after a manual switch to a restarted DB2 (to recover inflight activity), you must restore DB2 Group access via CEMT. See “Warm restart considerations” on page 51.

RESYNCMEMBER(NO) gives better DB2 availability

PROGRAM

The RDO PROGRAM definition has a new parameter `CONCURRENCY` which tells CICS about the multi-use standards applying to the program. In particular, whether or not it can safely execute on multiple TCBs at the same time: this is the Threadsafesafe property.

As this is a migration document, you are only concerned with usage of this parameter for CICS Application Programs which access DB2. This is more fully discussed in “Threadsafesafe coding” on page 40.

CONCURRENCY Implications for TRUEs

The `CONCURRENCY` parameter does not only effect programs. It will also apply to TRUEs. When used with a TRUE, it means the TRUE will obey the Threadsafesafe multi-activity rules and so can run on a L8 TCB.

You merely need to be aware that the DB2 TRUE is enabled with `CONCURRENCY (THREADSAFE)` so that DB2 activity runs under a L8 TCB.



The CONCURRENCY (QUASIRENT) setting for DB2 usage

This is the existing behaviour and default. It says that after each DB2 activity the Application Program will operate under the main CICS QR TCB. There is no performance improvement.

The CONCURRENCY (THREADSAFE) setting for DB2 usage

This says that, after DB2 activity, the executing program will remain on the L8 TCB used for DB2 access. It will not swap back to use the CICS QR TCB after DB2 activity, so giving a significant performance improvement for more DB2 activity (because this avoids a subsequent swap onto the L8 TCB as the program is already there).

The Application Program will swap back to the QR TCB when an EXEC CICS command which is not Threadsafe is issued. As there are not very many of these commands (they are listed in the *CICS APG* and *SPG books*), in practice this swapping will occur on the next EXEC CICS command encountered in the Application Program.

However, all traversed GLUEs as well as the Application Code must be correct (obey the Threadsafe rules). A lot of effort is needed to ensure that everything traversed obeys these rules.

If you want to take full advantage of this Threadsafe performance improvement, you will have to restructure your Application Programs to:

- group DB2 activity together without any intervening EXEC CICS commands.
- ensure that Shared Resources are accessed in a Threadsafe fashion

Don't forget that you must ensure all GLUEs obey the Threadsafe rules.

Your DB2 code has to be made Threadsafe to get better performance

Java definitions

As far as migration is concerned, other changes to the PROGRAM definition are for Java and so not relevant. However, I thought it might be of interest to include a brief overview of Java Class definitions.

To use a Java class (a routine that starts with a `main()` method) you must RDO define it as a program with `JVM=YES`. The item to run is named in the `JVMCLASS` parameter (which requires mixed case input on the terminal). You also need to define the attributes of the JVM via the `JVMPROFILE` setting. Observe that `CONCURRENCY (THREADSAFE)` is used.

An example of this definition is shown in Figure 4 .

```
CEDA View PROGRAM( DFHADJR )
PROGRAM      : DFHADJR
Group       : DFHADST
Description  : Jar inquiry program
Language    : Le370                CObol | Assembler | Le370 | C | Pli
RELoad     : No                    No | Yes
RESident    : No                    No | Yes
USAge      : Normal                 Normal | Transient
USElpacopy  : No                    No | Yes
Status     : Enabled                 Enabled | Disabled
RS1        : 00                     0-24 | Public
CEDf       : No                     Yes | No
DAtalocation : Any                   Below | Any
EXECKey    : User                   User | Cics
CONcurrency : Threadsafe             Quasirent | Threadsafe
REMOTE ATTRIBUTES
DYNamic    : No                     No | Yes
REMOTESystem :
REMOTENAME :
Transid    :
EXECutionset : Fullapi              Fullapi | Dplsubset
JVM ATTRIBUTES
JVM        : Yes                     No | Yes
JVMClass   : com.ibm.cics.addeploy.dfhadjr.DFHADJR
           :
           :
JVMProfile : DFHJVMPR
JAVA PROGRAM OBJECT ATTRIBUTES
Hotpool    : No                       No | Yes
```

Figure 4: PROGRAM definition for a Java Class

You must ensure that the `DFHJVM` PDS and the associated `system.properties` file have been correctly defined. This is all described in the *CICS Java Book*.

TCPIPSERVICE

New parameters on the TCPIPSERVICE definition should not cause you any problems for migration.

Just ensure that the `PROTOCOL` option is correctly set.

PROTOCOL

There is a new `PROTOCOL` item which explicitly names the access type for the socket:

- ECI (from the CICS Transaction Gateway)
- HTTP (for general Web Support)
- IIOP (for Java-related activity)

You should check that your TCPIPSERVICE definitions are correct for the traffic on their ports. These settings have a side effect as they control which type of URM is invoked for the flow.

AUTHENTICATE

This new parameter controls the security properties associated with the TCP/IP socket. Consequently, it sets the level of security validation which is applied to a flow. The default of No may not be what you want for a secured connection.

In addition, it can also automatically register SSL Client Certificates under a RACF Userid (usually done outside of a production environment).

MVS Workload management

Information is given to the MVS workload manager using `DNSGROUP` and `GRPCRITICAL` settings.

SSL Certificate for outgoing flows

This will not effect you, but the SSL certificate named in `CERTIFICATE` is stored in the Keyring database not a file (defined via `SIT.KEYRING`).

Set PROTOCOL for port usage

Lists

TCPIPSERVICE requirement

A TCPIPSERVICE definition must be installed before anything that uses it. This requirement does not have much of an implication for you in migration mode as you will not have anything (like a CORBASERVICE definition) which requires this pre-req.

However, in order to ensure that you do not face unexpected problems in the future, place all your TCPIPSERVICE definitions in distinct Groups, and place these Groups near the front of your List.

TYPETERM

XRFSIGNOFF

This will only effect you if you are using XRF - one parameter has got renamed, but the function is unchanged.

The XRFSIGNOFF parameter has been replaced by RSTSIGNOFF.

This interacts with the RSTSIGNOFF setting in the SIT and the XRSOFF entry in the CICS segment of RACF. It has an effect when VTAM Persistent Session support is being used to control the terminal security identity after a session failure.

Put TCPIPSERVICEs in a group early in the list

Summary

- Lower Case settings are required for HFS & Java items
- Redo all REQUESTMODELS
- DB2CONN
 - Group Attach defined via DB2GROUPID
 - RESYNCMEMBER controls recovery actions
 - Volume of activity controlled by TCBLIMIT
 - ◆ Interaction with SIT.MAXOPENTCBS
 - PRIORITY is relative to the QR TCB
- DB2ENTRY
 - PRIORITY is relative to the QR TCB
- PROGRAM has new CONCURRENCY setting
 - Provides enhanced DB2 performance
 - It takes a **lot** of work to get going
- TCPIP SERVICE
 - Must be installed before it gets used
 - PROTOCOL names activity expected on the port
 - Extra security via AUTHENTICATE
 - DNSGROUP and GRPCRITICAL for the MVS Workload Manager
 - CERTIFICATE is now stored in the SIT.KEYRING database
- TYPETERM
 - RSTSIGNOFF replaces XRFSIGNOFF

The Destination Control Table is no more. TDQ information can only be defined using RDO.

You have probably already removed the DCT, but if not then it must be done as part of the migration to CTS 2.2. You can also do this migration in your existing CTS 1.3 region.

To migrate the DCT:

- Assemble it with `TYPE= (INITIAL, MIGRATE)`
- Use `DFHCSDUP` via a `MIGRATE TABLE (name) GROUP=name`
- Put the new group into your list

DFHCNV is used to control CodePage translation. It must be recompiled for use in CTS 2.2. Similarly, all programs directly using this CodePage translation facility must be checked and recompiled (length fields have changed).

There is a new invocation of `TYPE=ENTRY, USREXIT` which permits use of your own alteration program, so avoiding the use of DFHUCNV. This allows you to take control of conversion.

Don't forget to retest applications as part of the migration plan.

Compiled Java (including IOP)

The HPJ compiler used to produce compiled Java code is now out of service. All existing compiled Java applications should be migrated to run natively inside a JVM.

However, as far as CICS is concerned, a compiled Java program is just another type of Language Environment program, and so does not care about how it was produced. Compiled Java programs, therefore, can continue to run unchanged within CICS, but as soon as a change is required, or a problem occurs, you will have to convert it to running inside a JVM.

Therefore, as part of your migration, convert all HPJ compiled Java programs to run natively inside of a JVM.

Converting HPJ programs

All you need to do is:

- FTP the source Java class (that which was compiled) to an HFS file
- Update the RDO PROGRAM definition to use `JVM=YES` and supply the class name & it's HFS location (see "Java definitions" on page 13)

As you will be running this code either via an `EXEC CICS LINK` or as a Transaction, keep the program name the same to avoid other changes. A native Java Class cannot link to another native Java Class (only one JVM can be active per transaction).

As you are not already using a JVM, ensure that the `DFHJVM` PDS and the associated `system.properties` file have been correctly defined. This Java setup is described in the *CICS Java Book*.

IOP Applications

Stateless CORBA Objects now **require** the use of a JVM and compiled applications must similarly be migrated to run natively inside a JVM. The IDLJ compiler from the SDK may prove useful in this migration.

A side effect is that the `GenFacIOR` is no longer needed, and a publish operation just puts the stringified IOR on the Shelf.

If this is a meaningless section: you do not have CORBA Objects in your CICS region, and so no migration to do!

Migrate compiled Java to run natively inside a JVM

SSL Certificates

Secure Socket Layer Certificates are now held in a database rather than a file. This repository change means the certificates have to be moved using Unix System Services Commands (options on the same command used to create and process certificates).

The relevant database is specified in the `SIT KEYRING` parameter.

DFHJVM

This is a new PDS that contains definitions related to the processing of JVMs (such as Storage requirements). You must ensure that it contains a DFHJVMPR member as this is required by CICS for its own JVM usage.

The contents of these members are defined in the *CICS Java Book*, in the *JVM Initialisation Options* section.

You can define other members, containing particular settings for a JVM. These member names get specified in the `JVMPROFILE` setting of the PROGRAM definition (see “Java definitions” on page 13) defining the JVM. When creating your own definitions, base them on `DFJIIRP`.

You should be aware that a PROGRAM object is the mechanism used to define the characteristics of a JVM. The REQUESTMODEL process selects a particular CICS Transid which points to the PROGRAM definition related to the required characteristics.

Inside the DFHJVM member is the name of an HFS file which contains more JVM-related definitions. This is conventionally called the `system.properties` file (ensure that this file name is entered in lowercase).

DFHEJOS & DFHEJDIR

These are new VSAM files used for EJB operation.

DFHGCD

You should increase the size of this file as more things are catalogued in CTS 2.2 than in CTS 1.3.

DFHCJVM

This file is no longer needed and should be deleted.

Runtime DB2 libraries

Don't forget to include the ERLY code libraries for DB2 (IPL required).

Runtime Language Environment Libraries

In order to be supported, you must delete any language-specific runtime libraries (for example, the runtime library supplied with OS/VS COBOL) and only use libraries supplied as part of the Language Environment component of MVS. See "Language Environment runtime issues" on page 28.

MVS Unix System Services

The MVS Unix System Services component must be running in full-function mode.

Compilation JCL

The latest Enterprise versions of the IBM COBOL and PL/I compilers support use of the (new for CTS 2.2) Integrated Translator.

This means that you do not have to do a separate translation step in your compilation JCL: all conversion of EXEC CICS (and EXEC SQL) statements are now done by the compiler in a single pass.

This means that Includes can now contain EXEC statements, and proper line-numbers apply to inserted code.

A document discussing the new facilities is available on the CICS Web Site via this link:

- [G325-5497: Application development improvements with CICS Transaction Server, Version 2.2](#)

Use the Integrated Translator

EXEC CICS SIGNON change

The semantics of the EXEC CICS SIGNON command (XC SIGNON) have altered for CTS 2.2. This command only works for terminal-attached transactions: non-terminal (background) transactions cannot use XC SIGNON (or XC SIGNOFF).

In CTS 1.3 this command can be used to change the security identity for a terminal-attached transaction in two ways:

- It alters the Userid used for the next transaction to be initiated at the terminal
- It alters the Userid currently being used for the terminal-attached transaction

In CTS 2.2 the first behaviour is retained, but the second is withdrawn.

This means that, in CTS 2.2, XC SIGNON will **only** effect the security identity of the next transaction to be initiated at the terminal. It cannot be used to alter the current security identity of an executing terminal-attached transaction.

A presentation is available on the CICS Web Site at this link:

- [CICS TS for z/OS V2.2 - Changes to the SIGNON Command](#)

The change

More formally: in CICS Transaction Server Version 2.2 the ability of a running CICS transaction to change its security identity (userid) is removed. Terminal attached transactions, like non-terminal transactions, have a fixed (at transaction initiation) security identity.

XC SIGNON and XC SIGNOFF now only effect the terminal definition. They do not alter the security identity of the running transaction, only that of the next transaction initiated at the terminal.

Temporary bypass

The existing behaviour can be temporarily restored for CICS TS version 2.n only (by running program DFH\$SNPI in the PLT) - subsequent releases of CICS which are not Transaction Server version 2s **will enforce** the static userid per transaction instance rule.

Cannot change the security identity of a running transaction

Migration planning

Application programs should be checked for the presence of `XC SIGNON` & `XC SIGNOFF` commands and changed to use other techniques to comply with the fixed security identity rule.

Finding application programs using the commands

You must find all application programs which use `EXEC CICS SIGNON` or `EXEC CICS SIGNOFF` commands. This can be done using the Load Module Scanner which is available for CICS TS 1.3 and CICS TS 2.2.

The JCL for this is shown in Figure 5 : the scanner uses a lot of memory, so don't forget the `REGION=0M` setting.

```
//SCANNER EXEC PGM=DFHEISUP, PARM=( ' SUMMARY ' ), REGION=0M
//STEPLIB DD DSN=<CICS>.SDFHLOAD, DISP=SHR
//SYSERR DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//DFHIN DD DSN=<your library>, DISP=SHR
//DFHFLTR DD *
SIGNOFF *
SIGNON *
/*
```

Figure 5: Load Module Scanner JCL

And then you change the found programs!!! This is the more difficult part.

Changing programs for transactions that run at terminals

It's all very well for me to say that once application programs have been located containing `XC SIGNON` and `XC SIGNOFF` commands they have got to be changed. This is not a straightforward thing to do. Each program will have to be examined and the most appropriate technique adopted.

Any modules found by the Scanner must run as terminal-attached transactions (or, at least, part of the code in the program does) as the commands do not work for background transactions.

Here are some code changes for particular uses of the commands.

Starting a non-terminal transaction with a given security identity

Figure 6 shows the code required: assuming the new userid exists and can execute the initiated transaction.

```
/* Start under a new Identity */  
EXEC CICS START TRANSID('RAH1')  
        USERID('RHARRI1')  
        INTERVAL(0) NOCHECK
```

Figure 6: Non-terminal Transaction initiation

Starting a transaction that is to run next at the terminal with another security identity

Figure 7 shows the code required to initiate a second transaction using a different security identity to that currently executing.

```
/* Assume Current Identity is RAH */  
  
/* Change Security Identity for Terminal */  
EXEC CICS SIGNOFF  
EXEC CICS SIGNON USERID('RHARRI1')  
/* Start more Transactions under RHARRI1 */  
EXEC CICS START TRANSID('RAH1')  
        INTERVAL(0) NOCHECK  
        TERMID(eibtrmid)  
  
/* Current Transaction still using RAH */
```

Figure 7: Same Terminal initiation

In this case, the terminal itself is going to have a new security identity for the next (and all subsequent) activity. This is processing for XC SIGNON and XC SIGNOFF commands that is unchanged (the first case) in CICS TS 2.2.

Figure 7 shows the security identity being altered in the terminal control block so subsequent transactions initiated at the terminal acquire this (new, quoted) security identity.

Recall that the TERMID and USERID settings on the XC START command are mutually exclusive.

Continuing a transaction at the terminal with a different security identity

By this I mean starting a new transaction at the terminal without the possibility of an intervening transaction or input interrupting processing. Figure 8 shows the coding sequence. This technique is used for normal transaction continuation - now it is being used to change security identity in the second transaction.

```
/* Assume Current Identity is RAH */  
  
/* Change Security Identity for Terminal */  
EXEC CICS SIGNOFF  
EXEC CICS SIGNON USERID('RHARRI1')  
/* Continue in new Transaction under RHARRI1 */  
EXEC CICS RETURN TRANSID('RAH1')  
IMMEDIATE
```

Figure 8: Continuing a Transaction

This technique is very similar to the previous case, except that an XC RETURN TRANSID is issued with the IMMEDIATE keyword. As the terminal's security identity has been changed, this second transaction will use the new userid.

Coding a signon transaction

This case is more complicated. Existing code is being used to change the security identity of a currently executing transaction (which replaces CESN): part of it executes under the default CICS security identity before changing to the requested Userid.

Code should be replaced with two transactions based on the technique shown in Figure 8 . The first transaction will:

- Request a userid and password
- Validate it
- XC SIGNOFF/SIGNON the terminal to the requested security identity
- XC RETURN TRANSID(second) IMMEDIATE

The second transaction will run under the required security identity to continue initial setup processing for the user.

Temporarily changing into Super-User mode at the terminal

This is now very difficult to achieve under CTS 2.2 as the general philosophy is to run everything for a transaction instance with a fixed security identity.

One way to achieve Super-User functionality at the terminal is to use `XC RETURN TRANSID() IMMEDIATE` techniques to split up the existing transaction into several smaller ones which will run sequentially at the terminal passing information between themselves. However, this means that instead of a single Unit Of Work there will now be several. This might effect LU6.2 and distributed transaction activity as well as altering recoverability aspects of current processing.

An alternative approach is to spin off separate transactions to do Super-User related work in the background, returning information to the original transaction. This technique has it's own difficulties associated with synchronisation and extended processing. The Business Transaction Service facilities of CICS can be used to assist in coupling of transactions.

Starting a transaction at another terminal with a different security identity from that currently being used at that terminal

This operation has never been available within CICS, and you still cannot do it! On the `XC START` command the `TERMID` and `USERID` parameters are mutually exclusive. Therefore, you can only start a transaction at another terminal using the security identity currently associated with that terminal.

However, once you have got a transaction running at this other terminal, it can always change the security identity thereby.

Language Environment runtime issues

I'm only going to discuss varieties of COBOL modules in this section: similar considerations apply to PL/I and C/C++ modules, but the impact in these cases is reduced. Some documentation is available on the CICS Web Site via these links:

- [COBOL and PL/I Compilers and CICS Transaction Server V2.2](#)
- [Language Environment within CICS TS: Questions and Answers](#)

What I mean by Language Environment

This whole matter is very confusing, so I'm going to have to be **very** precise about terminology. Here are some definitions:

Language Environment supplied libraries	Load libraries that are used at execution time and are supplied as part of the MVS Language Environment component. They have a name like MVS.LE370.ZOS140.SCEERUN
Runtime	An environment that is used to run a COBOL module within CICS. This consists of IBM-supplied items within a <i>Language Environment Supplied Library</i>
Language Environment runtime	A <i>runtime</i> that is configured using Language Environment parameters
non-Language Environment runtime	A <i>runtime</i> that is not affected by <i>Language Environment parameters</i>
Language Environment parameters	Items defined in various places which alter the <i>Language Environment runtime</i> environment but do not affect a <i>non-Language Environment runtime</i> environment
Supported	Something upon which IBM will accept an APAR
Unsupported	Something upon which IBM will not accept an APAR
Out of Service	equivalent to <i>unsupported</i>
Language Dependant Load library	An <i>unsupported</i> library supplied along with the compiler which provides execution support for thusly compiled modules and is a <i>non-Language Environment runtime</i>
Language Environment Environment	equivalent to <i>Language Environment runtime</i> , but customised for a specific application program

OS/VS COBOL programs

An OS/VS COBOL module is produced by these *unsupported* compilers:

- OS/VS COBOL (5470-CB1)
- OS/VS COBOL (5470-CB4)

You will have an OS/VS COBOL dependant load library in your JCL. This is now *unsupported*, so you need to move to a *supported* execution-time load library.

In order for you to be *supported* you must use a *Language Environment supplied library* in the CICS JCL. If you use any other library, it will be *unsupported*.

Inside the *Language Environment supplied Library* there are IBM-supplied modules that comprise a *supported runtime* for your OS/VS COBOL modules. This *runtime* is a *non-Language Environment runtime* and has nothing at all to do with a *Language Environment runtime*.

There is no detectable execution change between the existing OS/VS COBOL *out of service language-dependant load library* and the *Language Environment supplied library*.

JCL and runtime conclusion

Get rid of your existing OS/VS COBOL *language dependant load library* and replace it in the JCL with the current *Language Environment supplied library* and all will be well.

Compilation

All compilers for OS/VS COBOL are now *out of service* and the last translator to support usage for this source is CTS 1.3. Therefore, keep your existing CTS 1.3 translator around in case you need to change your OS/VS COBOL source. You will not be able to use any CICS API/SPI introduced after CTS 1.3.

This is allowed by your CICS license.

Use a Language Environment-supplied runtime library for OS/VS COBOL

VS COBOL II programs

Either Mode compilers

A VS COBOL II module can be produced by two sets of compilers:

- EitherMode compilers
 - ◆ VS COBOL II (5668-022)
 - ◆ VS COBOL II (5668-023)
 - ◆ VS COBOL II (5668-958)
- Compilers requiring a *Language Environment runtime*
 - ◆ SAA AD/Cycle COBOL/370 (5688-197)
 - ◆ COBOL for MVS & VM (5688-197)
 - ◆ COBOL for OS/390 & VM v2 (5648-A25)

A module produced by an EitherMode compiler can run either with its own VS COBOL II-supplied execution library (a *non-Language Environment runtime*) or using a *Language Environment runtime*.

This section is only concerned with the output from EitherMode compilers: the others require the use of a *Language Environment environment* and so will be using a *supported Language Environment-runtime* contained in the *Language Environment-supplied libraries*.

Indeed, if you have both sorts of modules in your region, you must be using the *Language Environment-runtime* already, so there are no migration considerations.

You may have amended (via the linkage editor) an EitherMode module so that it requires the use of the *Language Environment-runtime*; in which case you do not have any special migration concerns.

Migrate VS COBOL II modules to run within Language Environment

How CICS decides to use a Language Environment runtime

In CTS 2.2 (as in CTS 1.3) some VS COBOL II programs (those produced by the first set of compilers) can run either with a *non-Language Environment runtime* or with the *Language Environment runtime*. You do not have a choice as to which is used.

When CICS initialises, it attempts to install a *Language Environment runtime*. If this does not work (because, for example, the Language Environment RDO Group is not installed) the *non-Language Environment runtime* will be used for all VS COBOL II programs. If this initialisation works, the *Language Environment-runtime* will be used for all VS COBOL II programs.

However, in future releases of CICS Transaction Server, CICS will not initialise unless the *Language Environment runtime* is present. Consequently, all VS COBOL II modules, in these future releases, **WILL** run in the *Language Environment runtime* environment.

It is recommended that you migrate all your EitherMode VS COBOL II modules to run within the *Language Environment* as part of the migration to CICS TS 2.2.

VS COBOL II modules within Language Environment obey the Language Environment parms

Using a Supported non-Language Environment runtime

In order for your VS COBOL II modules to use a *supported non-Language Environment runtime*, you must use the *Language Environment supplied library* in the JCL (and ensure that CICS cannot initialise Language Environment). If you use any other library, it will be *unsupported*.

There is no detectable execution behavioural change between the existing *out of service VS COBOL II language dependant load library* and the *Language Environment supplied library* when the *Language Environment environment* is not present.

Simply get rid of your existing VS COBOL II *language dependant load library* and replace it with the current *Language Environment supplied library* and all will be well.

Using a Supported Language Environment runtime

There is only one *supported Language Environment runtime* library for VS COBOL II modules, and that is the *Language Environment supplied library*. Language Environment must be initialised in the CICS region (install the CEE RDO group).

As the VS COBOL II runtime is using the *Language Environment runtime* it will be affected by the *Language Environment parameters*. This can alter the behaviour of your VS COBOL II modules, depending upon what coding standards it adopted.

Setting Language Environment Parameters

The *Language Environment environment* for a running module is constructed under the control of various *Language Environment parameters*. These can be globally set for all CICS regions in a MVS image (CEECOPT), apply to everything in a given CICS region (CEEROPT) or be set for a specific module (via CEEUOPT linked to that module).

These Language Environment manuals may be of interest:

- SC28-1944-03: *Language Environment for z/OS and OS/390 Run-Time Migration Guide*
- GC27-1409-00: *Enterprise COBOL for z/OS and OS/390 Compiler and Run-Time Migration Guide*.

Language Environment parameters and program behaviour

The *Language Environment parameters* effect the *Language Environment environment* (more precisely: the characteristics of the Language Environment enclave used for the execution of an application program) and so have an impact upon program execution.

In the majority of cases, these values are merely for tuning, but there is at least one important setting which can effect program execution. This is the `STORAGE` parameter.

A VS COBOL II program may have been written on the assumption that obtained storage will have been cleared to `X'00'` before usage. When running in a *Language Environment Environment* this clearing may have been turned off for performance reasons. Consequently, execution errors can occur if a chunk of storage was assumed, for example, to be `X'00000000'` but was actually `X'40401234'`.

Another Language Environment option that may cause problems is `CBLPSHPOP` which has an effect around `EXEC CICS LINK` calls.

Increase the DSA

If you have not already installed Language Environment into your CICS region, you will need to increase the `EDSALIM` space setting in the SIT and ensure that `REGION=0M` is in the JCL.

Compilation

VS COBOL II modules are translated using the `COBOL2` option. However, this is withdrawn with CTS 2.2. The `COBOL3` option should be used for compilation.

You can keep using the CTS 1.3 translator (although unsupported) together with an unsupported compiler if you are unable to change VS COBOL II modules to use the `COBOL3` option. However, no API/SPI introduced after CTS 1.3 will be accessible. This is allowed by your CICS License.

Beware the `STORAGE` setting and code assumptions

SIT Changes

The SIT has some new parameters associated with Java support, and a few defaults are changed.

The things most likely to effect you are the settings to do with the new TCBs used for DB2 support (MAXJVMTCBS & MAXOPENTCBS) - don't set these too low.

You should always (and especially If you are implementing Language Environment or JVMs into the region), be sure that the CICS Storage settings (EDSALIM, ECSDSASZE, ERDSASZE, ESDSASZE, CDSASZE, RDSASZE, SDSASZE) are big enough.

Deleted parms

DCT

This table has been replaced by RDO. See page 17.

KEYFILE

The SSL Certificate database has been replaced by a database (defined via KEYRING). See page 20.

MNEVE

This setting has been replaced by function residing within the MVS Workload Manager.

XRFSOFF & XRFSTIME

The parameters have been replaced by RSTSIGNOFF and RSTSIGNTIME (just renames).

Altered Defaults

LGDFINT

The Logger Interval now defaults to 5ms (it was 30ms).

MNFREQ

The minimum Monitoring Interval is now 1 minute (previously 15mins).

STNTR & SPCTR

There are new domains: EJ, II, OT, PT, RZ and SJ ; so they take the global trace defaults.

MAXOPENTCBS

This parameter controls the number of L8 TCBs present in the CICS region. The default is now 12 (and the maximum 2000).

Some of these will be used for DB2 activity. Ensure that **SIT.MAXOPENTCBS > DB2CONN.TCBLIMIT** or else DB2 performance problems will appear.

If you do not have DB2 access or Java in your CICS region, you can reduce MAXOPENTCBS to 1.

Do not set MAXOPENTCBS too low

New parameters for resources

MAXJVMTCBS

This controls the number of J8 TCBs used for JVM activity. This is the same thing as saying how many JVMs you want in your CICS region.

Setting this too low will cause Java/EJB activity to wait, but setting it too high will waste storage. The default is 5 (which is probably too low if you are going to use Java).

Set MAXJVMTCBS to 1 if not running Java or EJBs in your region.

MAXHPTCBS

This parameter controls the number of H8 TCBs which are used to run compiled Java (see page 19). It defaults to 5 (which is definitely too high if you are not using compiled Java).

If you have migrated all HPJ-compiled Java classes to run natively inside a JVM (as this document suggests) or have never had any compiled Java in your CTS 1.3 region, set MAXHPTCBS to 1.

MAXSOCKETS

This parameter controls the maximum number of TCP/IP sockets that CICS will accept traffic upon. The default is 65535.

This parameter should only be reduced if you are having capacity problems servicing IP-initiated traffic in the CICS region.

However, if you do not accept IP traffic in the CICS region (no TCPIP SERVICE items installed), then set MAXSOCKETS to 1.

KEYRING

This names (case is significant) the database used for SSL Certificates. These certificates will have been exported from the KEYFILE named in the 1.3 SIT. See page 20.

Do not set MAXJVMTCBS too low for Java activity

New parameters relating to intervals

RSTSIGNOFF & RSTSIGNTIME

These new SIT parameters (essentially renamed from XRF`SOFF` & XRF`STIME` as they now apply to LU2 terminals) take defaults of `Stay_Logged_on` and log every 5 minutes.

STATEOD & STATINT

These two parameters control the End Of Day time for Statistics (defaulting to midnight) and the interval for Statistics collection (defaulting to every 3 hours).

The `STATEOD` parameter may permit the removal of a PLT program that uses an `EXEC CICS SET STATISTICS ENDOFDAY` to set the End Of Day time.

New parameters relating to things not in CTS 1.3

BRMAXKEEPTIME

This setting relates to 3270 Bridge facilities and controls the garbage collection interval (defaulting to the maximum delay of 24 hours).

It is not of any interest to you from a migration viewpoint, so code the default.

AIBRIDGE

This permits the 3270 Bridge URM to define facilities for the 3270 emulation. It defaults to AUTO (which does not invoke the URM).

IIOPLISTENER

This controls whether or not CICS can accept IIOPL requests (those which flow from an external EJB wanting to run an EJB within the CICS region). It defaults to YES.

If you do not intend to run EJBs within CICS, set this parameter to NO.






























XEJB

This controls whether or not EJB role-based security is active in your CICS region. It defaults to Security being enabled. Leave this set to the default until you implement EJB activity in your CICS region (you might want to turn off EJB security in a development environment).

EJBROLPRFX

This is a setting relating to EJB role-based security. Leave it set to the default of ' ' until you implement EJB activity. To use this setting, you will have to turn on EJB security via XEJB and configure RACF definitions.

Summary

	AIBRIDGE	Bridge URM activity
	BRMAXKEEPTIME	Bridge Garbage Collection
	DCT	Deleted
	EJBROLPRFX	EJB Role Security prefix
	IIOPLISTENER	IIOPL Traffic allowed
	KEYFILE	Replaced by KEYRING
	KEYRING	SSL Certificate Database
	LGDFINT	Logger Interval default
	MAXHPTCBS	Compiled Java TCBS
	MAXJVMTCBS	Number of JVM TCBS
	MAXOPENTCBS	Number of L8 TCBS (including DB2)
	MAXSOCKETS	Number of IP Sockets
	MNEVE	Deleted
	MNFREQ	Stats Interval default
	RSTSIGNOFF	XRF Signon retention
	RSTSIGTIME	XRF Persistent Session timeout
	SPCTR	New Domains
	STATEOD	Stats End of Day Time
	STATINT	Stats sample interval
	STNTR	New Domains
	XEJB	EJB Security enablement
	XRFSOFF	Replaced by RSTSIGNOFF
	XRFSTIME	Replaced by RSTSIGTIME
	New parameter	
	Deleted parameter	
	Replaced by another parameter	
	Parameter being renamed	
	Default changed	
	New settings available	

GLUEs and TCBs

All CICS Global User Exits can now (potentially) be invoked on any of the TCBs used by CICS. In addition to the (usual) QR TCB (in which most application programs execute), a GLUE can be invoked (for example) whilst running on L8 (DB2 activity) or J8 (JVM) TCBs.

This means that two instances of a GLUE program can be running at the same time on different TCBs.

Therefore, all GLUE programs **must** be Threadsafe. This means that they must be coded to support multiple access to Shared Resources. Consequently, certain parts of the exit program must be Critical Sectioned (serialised) to prevent multiple access to a Shared Resource. A Shared Resource is, for example, a block of Shared Storage (obtained via an `EXEC CICS GETMAIN SHARED`) or a GLUE workarea.

A Critical Section (as far as CICS is concerned) is a chunk of code that will only ever be executing once within the CICS region. The implementation of this serialisation concept is via ENQs, which means using the XPI ENQ/DEQ function `DFHNQEDX` inside an exit program.

You must ensure that **all** vendor supplied GLUE programs are Threadsafe.

Threadsafe coding

For example, whenever a GLUE program uses the GLUE Workarea access should be Critical Sectioned to prevent multiple accesses (the usual problem of one instance updating a counter whilst another instance gets the previous number). an identical problem arises when the exit program accesses some Shared Storage.

The assembler Compare-and-Swap (CS) instruction can be used to update fields or process ECBs in an atomic fashion.

Use a `DFHNQEDX FUNCTION(ENQUEUE)` to delimit the start of the Critical Section and end with an equivalent `DFHNQEDX FUNCTION(DEQUEUE)`. Included code will only ever be running once in the CICS region, so access to the Shared Resource is serialised.

As you are ENQing upon a name, you need to establish Installation Naming Standards (so that you can easily see through CEMT that a wait is engendered within a GLUE program).

All GLUEs must be Threadsafe to enable DB2 performance improvements

Migration Considerations

- Establish Installation Standards for GLUE Critical Sectioning
 - Decide upon the ENQ names to be used
- Recode your own GLUEs to obey Threadsafe rules
 - Use `DFHNQEDX` to establish Critical Sections
 - Use `CS` instructions to update outside of a Critical Section
 - Use `CS` instructions to process ECBs
- Ensure that all Vendor-supplied GLUEs are Threadsafe
 - By not assuming single TCB activity
 - Critical Section access to Shared Resources

Some GLUEs are more important than others as they are more likely to be multiply running. These are XRMIIIN, XRMIOOUT, XEEIN and XEIOOUT.

If you intend to use the DB2 Performance Improvements, **all** GLUEs **must** be Threadsafe.

GLUE parmlist changes

There are new TCB modes in `DFHUEPAR.UEPGIND` to show which TCB type is being used.

Parmlist changes effect XFAINTU, XRSINDI and XSNON.

Check Vendor-supplied GLUEs for Threadsafeness

File Control GLUEs

Exec-level File Control exit programs (those which use XFCREQ and XFCREQC) **must** be changed.

In CTS 2.2 the function located in these exits has been morphed into four GLUES:

Exit name	Runs in which CICS region	What the GLUE does
XFCREQ	The region in which the EXEC CICS command was issued	<ul style="list-style-type: none">● Change the request or reject it.
XFCFRIN	The region in which the file is physically located	<ul style="list-style-type: none">● Accept or Reject the Operation.● Route to another region.● Do your own I/O.
XFCFROUT	The region in which the file is physically located	<ul style="list-style-type: none">● Alter the data to be returned as the operations result.
XFCREQC	The region in which the EXEC CICS command was issued	<ul style="list-style-type: none">● Change the return code.

You must ensure that **all** your vendor-supplied File Control GLUE code is suitable for use in CTS 2.2.

You need to read the *CICS CG* manual to understand what is going on.

File Control GLUES have split

XFCREQ & XFCREQC

The XFCREQ & XFCREQC exits run in the Exec Interface layer. They are not part of a CICS domain and API/SPI commands can be issued.

In CTS 1.3 these exits handled everything associated with the processing of a file I/O request. You could:

- ◆ alter the request
- ◆ do you own I/O
- ◆ alter the results.

They got invoked for both locally-initiated and routed requests.

In CTS 2.2 these exits are **only run once** in the region issuing the EXEC CICS filecontrol request. They are not invoked (unlike in CTS 1.3) in the region containing the physical file.

Consequently in CTS 2.2, XFCREQ only intercepts the request in the region that issued it. You can amend the request or reject it.

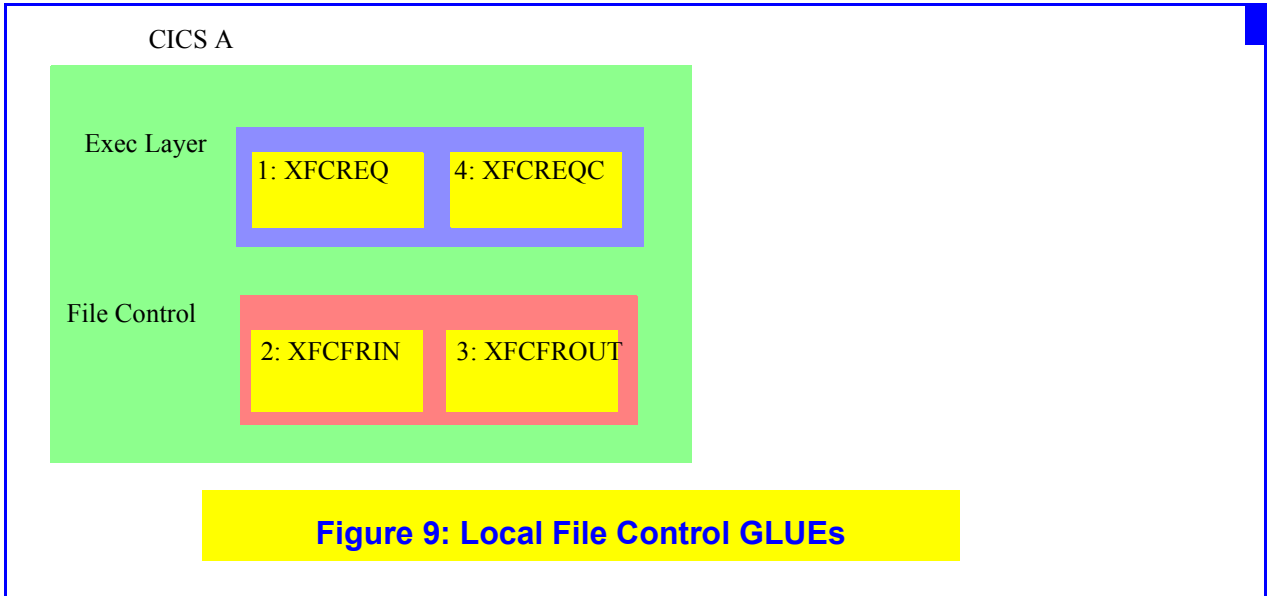
Similarly, XFCREQC is only executed in the region that issued the request after the request has finished processing. You can only amend EIB return codes. If you want to alter the response itself (such as amend a returned record), you must do that in XFCFROUT.

XFCREQ/C only run in originating region

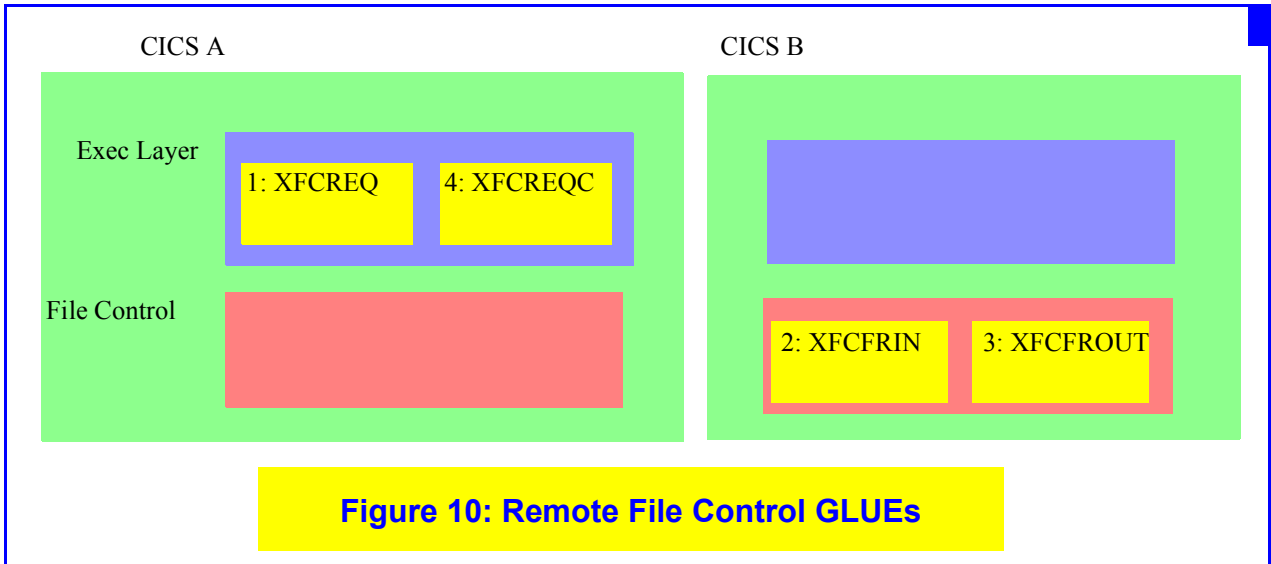
XFCFRIN & XFCFROUT

The XFCFRIN & XFCROUT exits provide points where the rest of the CTS 1.3-GLUE provided function is executed. These GLUEs run in the region where the file is physically located. These exits are not located within the Exec layers, and so CICS API/SPI commands are not available.

If the file is local, Figure 9 shows XFCFRIN/OUT running in the same region as XFCREQ/C.



If the file is remote, Figure 10 shows XFCFRIN/OUT running in a different region to XFCREQ/C.



XFCFRIN/OUT run where the file is located

XFCFROUT enables you to change the data sent back to the originating EXEC CICS filecontrol command.

XFCFRIN can decide to reject the request, process it locally or route it to another region for processing. In this case you can get multiple usages of XFCFRIN/OUT, but still only one use of XFCREQ/C.

If XFCFRIN decides that the request is to be processed locally, then either standard CICS File Control can be used to satisfy the request, or all standard operations can be replaced by your own I/O operations.

If a vendor-supplied product uses the File Control GLUEs, you **must** ensure it is suitable for use with CTS 2.2 (ask if the function has morphed into XFCFRIN & XFCFROUT).

There is another side effect for the XFCFRIN/OUT GLUEs: some file operations initiated by CICS code also traverse these exits. Don't alter them in any way!

CICS activity will flow through these GLUEs: do not alter!

User Replaceable Modules should also obey the Threadsafe Criteria (as described in “Threadsafe coding” on page 40). You must check that all vendor supplied URMs are Threadsafe, as well as checking your own URMs.

However, URMs tend only to decide upon some sort of action for the current transaction/activity and so do not usually access Shared Resources. Consequently, the probability is that all your URMs are already Threadsafe (as they do not do anything that could fail when multiply executing).

NEPs

Node Error Programs have a new option enabling the printing of the Network Qualified Name.

Terminal Autoinstall

Some sample code has been provided for the terminal Autoinstall URM (DFHZATDX) to extract the NQN (Network Qualified Name) from a CINIT/BIND.

New URM functions

There are some new User Replaceable modules for functions not available in CTS 1.3, so these should not be of any interest to you (yet!).

- DFHDSRP
 - ◆ RequestStreams added
- DFHDYP
 - ◆ Dynamic Routing added for the Link3270 Bridge
- DFHPGADX
 - ◆ JVMPROFILE parameters can be set on Autoinstall
- DFHZATDX
 - ◆ Link3270 Bridge added

Non-migrational changes

This section outlines other changes which are not of great importance from the migration viewpoint.

API, SPI, CEMT and Supplied Transactions

File Control API	RESP2s are returned for remote files
Program Control API	RESP2s are returned for Java errors
XC ASSIGN NETNAME	Can return an LU alias
XC SIGNON/SIGNOFF	Reminder: does not change the security identity for the current executing terminal-attached transaction
XC INQ DISPATCHER	New options to return the number of TCBs (HPJ, JVM, Open), the region exit time, the terminal scan delay and the runaway task interval
XC SET DISPATCHER	Equivalent new options
New INQ/SET	CORBASERVER, DJAR, BRFACTILITY, JVMPOOL
CEMT	INQ/SET DISPATCHER is available
CEOT	Switching for mixed case input
CREA	Generation of REQUESTMODEL definitions
CREC	Display DJAR transactions via REQUESTMODEL
CLER	Display current Language Environment region settings

(XC is a shorthand for EXEC CICS)

Some more RESP2s are returned for File Control

RDO Objects

CORBASERVER	New object
DJAR	New object
TRANSACTION	OTSTIMEOUT relates to an interval for EJBs to undertake Unit Of Work activity
TCPIPSERVICE	Must be installed before anything that uses it (like a CORBASERVER)
PROGRAM and Java	<ul style="list-style-type: none">● JVM(YES) to run a Java class in a JVM.● The debug option has moved into the JVMPROFILE member● The JVMPROFILE names the DFHJVM member for JVM configuration options● DFHPRJVM is the JVMPROFILE used by CICS-supplied Java transactions
PROFILE	<ul style="list-style-type: none">● DFHCICSI is a new group for IIOp requests● RTIMOUT applies to IIOp request processors

URMs

DFHSJJ80	Sets Language Environment options for a JVM
DFHXOPUS	Userid setting for inbound IIOp requests
DFHEJDNX	Provide a name where no SSL Client Certificate is used
DFHEJEP	Monitors EJBish events
DFHJVMAT	Tailors JVM options

A TCPIPSERVICE must be installed before a reference to it

CPSM migration

CPSM migration is the same as usual:

- Migrate CPSM by MVS image
 - Migrate the Maintenance Point first
 - The CAS, CMAS and MAS agent code must all be at the same CPSM release level
- Link3270 Bridge Routing is not supported
- Some functions have been removed
 - Tivoli Global Enterprise Manager agent code
 - RODM and RMAS (for both CICS/ESA and CICS TS)

Normal CPSM migration

In this document, I have covered disparate aspects of DB2 migration (to DB2 v7.1) within CICS TS 2.2. I summarise these in this chapter.

DB2 and CICS TCBs

DB2 activity now occurs under L8 TCBs, and the affinity between the DB2 Thread and the TCB has been removed (see “DB2CONN and DB2ENTRY” on page 9). Standard CICS debugging and control techniques apply. This means that DB2 activity can now be purged through CEMT and the number of TCBs is both configurable (see “MAXOPENTCBS” on page 35) and changeable at runtime.

A performance improvement is available by permitting an Application Program to remain on the L8 TCB after DB2 activity - this avoids instructions associated with TCB switching. However, the CICS region and its contents must be properly setup to exploit this performance improvement.

This is the Threadsafe criteria. See “Threadsafe coding” on page 40

DB2 Purgeability

This change...

- The DB2 TRUE is enabled as PURGEABLE
 - DB2 activity can be purged by CEMT

implies that...

- Operators
 - Can attempt to alleviate DB2 waits

Some presentations and associated documentation mentioned that DB2 activity is subject to Runaway Task Control. After customer suggestions, this has now been changed so that DB2 activity is **not** subject to Runaway Task control (the situation is as it is in CTS 1.3).

DB2 activity now uses L8 TCBs

DB2 Group Attach

This function enables CICS to talk to 1-of-n DB2 v7s in the MVS Image (not the Plex). CICS gets any **one** DB2 image in the group (selection is not influencable).

The specific DB2 instance being contacted can be changed by use of CEMT/EXEC CICS SET DB2CONN DB2ID(y) or DSNCLSTRT y.

Actions on a failure are configurable. This means that the Recovery actions for Indoubt Units Of Work can be postponed until a favourable time interval.

When a failure occurs CICS will make one attempt to contact the last used DB2. If this fails, then the action taken depends on the RESYNCMEMBER setting within the DB2CONN definition (see "DB2CONN and Group Attach" on page 10).

RESYNCMEMBER (YES) gives an equivalent behaviour to CTS 1.3: CICS will wait forever until the failed DB2 restarts. No new DB2 activity is permitted until all outstanding Units Of Work have been resolved.

RESYNCMEMBER (NO) gives new behaviour which increases DB2 availability. CICS will make one attempt to recontact the failed DB2 instance. If this is unavailable, CICS will contact any other DB2 instance (which one selected is not influencable) in the group, and send new database activity to it. The inflight Units of Work are not resolved, and will stay around. The advantage of RESYNCMEMBER (NO) is that new database activity can flow, so still enabling new CICS Transactions to proceed : a big availability increase.

Later on, when convenient, CICS must be manually switched to use the restarted (lately failed) DB2 instance in order to resolve the Inflight Units Of Work.

Warm restart considerations

The DB2 type of Attach (Group or Specific) is catalogued and consequently restored over the CICS Warm restart process. Thus, after a manual switch to a restarted DB2 (to recover inflight activity), you must restore DB2 Group Attach access via CEMT/EXEC CICS SET DB2CONN DB2GROUPID.

If you do not do this, Group Attach is effectively turned off until CICS is Cold started. As the catalogue will say that a Specific Attach was last active, CICS will contact that given DB2 instance on a Warm restart. If that DB2 instance is not available, then DB2 activity will not proceed.

So, don't forget to restore DB2 Group Attach functionality after recovery.

Group Attach gives better DB2 availability

DB2 and Threadsafes roadmap

Assuming CTS 2.2 with DB2 v7.1:

1: Set <code>SIT.FORCEQR=NO</code>	Using the L8 TCB gives a small performance benefit
2: Use Group Attach with <code>RESYNCMEMBER (YES)</code>	Gives CTS 1.3 existing behaviour (wait for everything to resync)
3: Change to <code>RESYNCMEMBER (YES)</code>	Resync postponed until convenient - better DB2 availability
4: Get Vendors to assert GLUEs, TRUEs and URM are Threadsafes	
5: Check your GLUEs, TRUEs and URM are Threadsafes	
6: Check Application Programs for Threadsafeness	This mostly means protecting use of Shared Storage areas
7: Amend Application Programs to group DB2 activity	No intervening <code>EXEC CICS</code> commands amongst DB2 activity
8: test, Test & TEST again in a properly multiuse environment	
9: Copy the existing RDO PROGRAM definition	So you have a quick backout by reinstalling the old definition
10: Energise by using <code>PROGRAM.CONCURRENCY (THREADSAFE)</code>	
11: Keep on moving	

Migration Conclusion

Compared with the transition from CICS/ESA 4.1 to CICS TS 1.3 - which required getting to grips with the MVS Logger - the migration from CICS TS 1.3 to CICS TS 2.2 is easy:

- Normal Recompilations
 - GLUEs, URM, Conversions, Tables
- RDO/CSD
 - Create the CSD with a bigger RECSZ
 - Convert the DCT
 - Check DB2 RDO definitions
 - Redo all REQUESTMODELS
- Set new parms in the SIT
 - `SIT.MAXOPENTCBS > DB2CONN.TCBLIMIT`
- Recode GLUEs for ThreadSafeness
 - Do not assume QR single threading
 - Critical Section access to Shared Areas (including GWAs)
- Morph function into new File Control GLUEs (XFCFRIN/OUT)
- 1.3 Java modules
 - Replace compiled Java with native JVM execution
- Alter signon programs to cope with the `EXEC CICS SIGNON` change
- Convert VS COBOL II modules to run within Language Environment
- Talk to Vendors
 - GLUEs to be ThreadSafe
 - URM to be ThreadSafe
 - XFCFRIN/OUT used to File Control operations

