CICS Open Transaction Environment (OTE)

CICS Development
Hursley Park, Winchester, UK

ibm.com/cics

Notes

CICS® Transaction Server for z/OS® (CICS TS) V1.3 introduced the first phase of Open Transaction Environment (OTE) whose long term aim is to make the CICS application execution environment truly open, allowing applications to be defined to execute under their own TCBs within CICS and allowing CICS to better exploit multiple processors.

In CICS TS 2.2 the Java Virtual Machine and the CICS-DB2 Attachment Facility exploited OTE and OTE capabilities were further extended in CICS TS 3.1. Further components have jumped on the OTE bandwagon in the newest CICS TS release.

This slide deck is based on a presentation by John Tilling, which has been updated by Catherine Moxey.

These materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

This slide left intentionally blank.

Introduction to Open Transaction Environment (OTE)

- **Objectives**
  - Enable applications on CICS TS to use non-CICS APIs
  - Open CICS TS to new types of client
  - Early support of new technology, e.g. Java Virtual Machine (JVM)
- **Function**
  - Access to POSIX functions, HLL functions, HFS, sockets
  - Improved performance for resource manager adapters eg. DB2, MQ
- **Stage 1 (delivered in CICS TS 1.3)**
  - CICS base infrastructure changed.
  - OTE used for JVM and Java Hot Pooling
- **Stage 2 (delivered in CICS TS 2.2)**
  - Support for Task Related User Exits (TRUEs)
- **Stage 3 (delivered in CICS TS 3.1)**
  - Support for OPENAPI programs and C/C++ programs using XPLINK

Notes

- Examples of non-CICS APIs would be MVS services and MVS Unix System Services POSIX functions.
- An example of opening CICS up to new types of client would be that OTE would enable listener tasks written for other platforms to be imported to run under CICS and hence service requests from their relevant clients.
- Use of the MVS JVM inside CICS is a good example of migration to new technologies.
- Open Transaction Environment (OTE) has been implemented over several releases.
- CICS TS 1.3 implemented stage 1 of OTE. In CICS TS 1.3, the base infrastructure of CICS was changed to eventually allow multiple TCBs for application use. Also many of the external changes were put in place e.g. changes to program definition panels, SIT parameters, options on the ENABLE command. However, many of these changes are 'dormant' in CICS TS 1.3.
- In CICS TS 2.2, support for exploitation of OTE by Task Related User Exits was added (stage 2).
- In CICS TS 3.1 support was added for OPENAPI programs, which allows applications to request their own TCB to run on from the start. A special case of this is support for C and C++ programs using XPLINK support, which is described later.
- OTE necessitates changes in other products such as MVS BCP, MVS Unix System Services, Language Environment (LE), DB2. These have also been delivered over several releases.

Terminology

- **Quasi-reentrant (QR) TCB**
  - The main CICS TCB - under which **all** application code ran prior to OTE
  - CICS dispatcher subdispatches work, so each CICS task has a slice of the action
  - A CICS task gives up control via a CICS dispatcher wait
  - Only one CICS user task is active at any one time

- **Quasi-reentrant programs**
  - Same program can be invoked by more than one CICS task
  - But only one CICS task is active at any one time
  - Quasi-reentrancy allows programs to share virtual storage e.g. CWA without the need to protect against concurrent update
  - CICS code takes advantage of quasi-reentrancy, e.g. field CSACDTA in the CSA addresses the TCA of the currently dispatched CICS task (field renamed in CICS TS 3.1 and fetch protected in CICS TS 3.2).

Notes

- Prior to OTE, all application code ran under the main CICS TCB called the quasi-reentrant (QR) TCB. The CICS dispatcher subdispatches use of the TCB between the CICS tasks. Each task voluntarily gives up control when it issues a CICS service that issues a CICS dispatcher wait. There is only ever one CICS task active at any one time on the QR TCB.
- Programs are said to be quasi-reentrant programs because they take advantage of the behaviour of the CICS dispatcher and the QR TCB, in particular that there is only ever one CICS task active under the QR TCB. This means that although the same program can be being executed by multiple CICS tasks, only one of those CICS tasks is active at any one point in time. Contrast this with a situation whereby multiple instances of the same program are executing each under a separate TCB. In this situation multiple tasks would be active in the same program at the same time and the program would have to be fully MVS reentrant.
- Quasi reentrant programs can access shared resources such as the common work area (CWA) or shared storage obtained via EXEC CICS GETMAIN SHARED safe in the knowledge that they are the only CICS user task running at that instant.
- Field CSACDTA in the CICS CSA is a field that relies on quasi-reentrancy. It points to the TCA of the currently dispatched CICS task. It only has meaning when running under the QR TCB. In CICS TS 3.1 all CICS use of CSACDTA was removed and the field renamed CSAQRTCA. **In CICS TS 3.2 the field is loaded with a fetch protected address. If used, an abend ASRD will result.**

Terminology

- **Open TCBs**
  - A new class of CICS TCB available for use by applications
  - Each TCB is for the sole use of the owning CICS task but can be reused by a later task.
  - No subdispatching under Open TCBs, blocking by applications allowed
  - There are several different types or modes of Open TCB.
  - CICS dispatcher domain manages a pool of TCBs for each mode
  - CICS will switch between an Open TCB and the QR TCB as required
  - Open TCBs are 'daughters' of QR TCB in the overall CICS TCB hierarchy

Notes

- OTE introduces a new class of TCB called an open TCB that can be used by applications. An open TCB is characterised by the fact that once it is assigned to a CICS task there is no subdispatching of other CICS tasks under the open TCB. The application can use it to execute under and issue non-CICS API requests that may involve the TCB being blocked, e.g. MVS GETMAIN. Blocking is allowed because only this open TCB is halted, and not the whole of CICS. When the CICS tasks ends, the open TCB can be reused by a different CICS task.

- Within the overall class of open TCB there are various modes of TCB. A mode is be identified by a two character name.

- CICS dispatcher will manage 'pools' of open TCBs, one for each mode.

Terminology

- **Open TCB modes**
  - Mode J8 is a key 8 TCB used by the CICS-JVM interface
  - Mode J9 is a key 9 TCB used by the CICS-JVM interface
    - Added in CICS TS 2.3
  - Mode L8 is a key 8 TCB used for OPENAPI TRUEs and OPENAPI programs
    - Support for OPENAPI TRUEs added in CICS TS 2.2, e.g. the CICS-DB2 interface.
    - Support for OPENAPI PROGRAMS added in CICS TS 3.1
  - Mode L9 is a key 9 TCB used for OPENAPI programs
    - Added in CICS TS 3.1
  - Mode X8 is a key 8 TCB used for XPLINK support
    - Support for XPLINK added in CICS TS 3.1
  - Mode X9 is a key 9 TCB used for XPLINK support

Notes

- Within the overall class of open TCB there are various modes of TCB. A mode is identified by a two character name.
- The J8 open TCB mode is the type of TCB used to execute a Java Virtual machine (JVM). In CICS TS 2.3 this was expanded to have J9 open TCBs which are used to run userkey Java applications.
- L8 is an open TCB mode used in CICS TS 2.2 for TRUEs that exploit OTE. In CICS TS 3.1 they are also used for CICS key openapi programs in CICS TS 3.1
- L9 is an open TCB used in CICS TS 3.1 for user key openapi programs.
- X8 is an open TCB used in CICS TS 3.1 for cics key C or C++ programs using XPLINK.
- X9 is an open TCB used in CICS TS 3.1 for user key C or C++ programs using XPLINK.
- (H8 was a open TCB mode used for Java Program objects using Hotpooling. This was support provided in CICS TS 1.3 prior to support for the IBM Persistent Reusable JVM (PrJVM). Hotpooling was removed in CICS TS 3.1.)

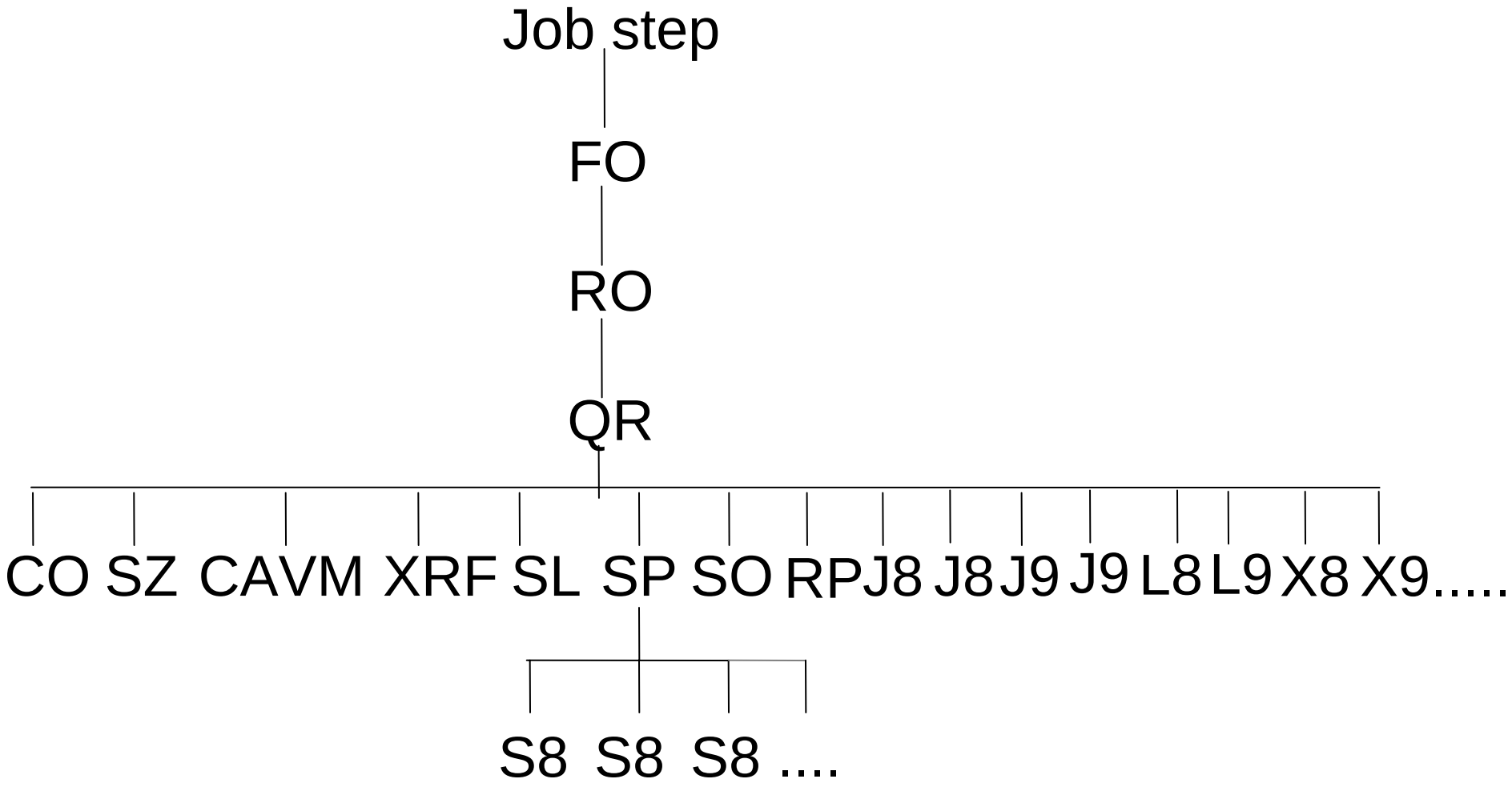
Allocation of OPEN TCBs

- **L8/L9 TCBs**
  - Allocated for the life of the task
- **J8, J9, X8, and X9 TCBs**
  - Allocated for the duration of the program

Notes

- When an L8/L9 TCB is assigned to a task it remains allocated for the life of the task, while other types of open TCBs are allocated only for duration of the function (for example an EXEC CICS LINK to a JAVA program).
- The environment can get rather complicated. For example, if a JVM program, running under a J8 mode TCB, issues a JDBC or SQLJ request which is processed through the CICS DB2 adapter, CICS switches execution to an L8 mode TCB for the DB2 request. On completion of the DB2 request, CICS returns control to the JVM under its original J8 mode TCB.

CICS TS 3.2 TCB hierarchy



Notes

- The picture shows the overall TCB hierarchy for CICS TS 3.2
- Under the QR TCB a number of daughter TCBs can be seen. Multiple J8 and J9 TCBs can be seen and represent multiple CICS tasks, each using a JVM. Also multiple L8 TCBs used by the CICS-DB2 Attachment Facility.
- Other than the L8, L9, J8, J9, X8 and X9 TCBs, the other TCBs are all CICS system TCBs
  - FO is the file owning TCB used to offload various file operations
  - RO is the resource owning TCB under which program loads are performed, amongst other things
  - QR is the main CICS TCB under which applications are subdispatched
  - CO is the concurrent TCB used to offload I/O by various domains
  - SZ is used for FEPI
  - RP is used for ONC RPC calls
  - CAVM and XRF are used for XRF support
  - SL is the sockets domain listener TCB and SO is used by the Sockets domain
  - SP TCB owns the S8
  - Multiple S8 TCBs are used by Sockets domain for SSL support1

Terminology

- **Threadsafe programs**
  - Are capable of being invoked on multiple TCBs concurrently
  - Are normally read-only, they do not in general overwrite themselves
    - Could overwrite themselves if updates are serialised correctly like shared storage
  - Cannot rely on quasi-reentrancy to serialise access to resources and storage
  - Must use serialisation techniques such as compare and swap (CS) or enqueue/dequeue to access shared resources with integrity
  - All programs accessing a shared resource must be made threadsafe e.g. an existing program's reliance on quasi-reentrancy to serialise access to the CWA is made invalid if just one other program can run concurrently on another TCB and access the same CWA field
  - Are sometimes referred to as fully MVS reentrant programs
    - MVS reentrant is often misunderstood to mean that programs do not overwrite themselves. We use the term threadsafe to avoid confusion

Notes

- The term THREADSAFE is used in preference to the term 'fully MVS reentrant' as the latter term is often misunderstood to just mean that a program is linkedited with the RENT option and hence is read-only and does not overwrite itself.

- For a program to be threadsafe it has to behave correctly when invoked concurrently under multiple execution units, i.e. TCBs. If it accesses shared storage for example it must ensure that updates to shared storage are serialised. If the program overwrites itself then this is a kind of shared storage and has to be serialised accordingly.

Terminology

- **CICSAPI - services available prior to OTE (under QR TCB)**
  - CICS command level application programming interface
  - CICS system programming interface
  - CICS Resource Manager Interface (RMI)
  - CICS Exit Programming Interface (XPI) - for Global User exits
  - Systems Application Architecture (SAA) Common Programming Interfaces
    - CPI-C and CPI-RR
  - LE callable services

- **OPENAPI - <u>additional</u> APIs possible under Open TCBs**
  - Use of MVS services
  - Use of a specified set of POSIX services via MVS Unix System Services

Notes

- Cicsapi is the term used to describe the set of services available prior to OTE. (Prior to CICS TS V3, for exit programs, the term BASEAPI was used rather than CICSAPI).
- Openapi is the term used to describe the set of APIs available to a program once it is running on an open TCB. An openapi program can issue any of the services available for a cicsapi program but in addition it is able to issue 'foreign API' calls, like MVS services.

Problems addressed by OTE (1 of 3)

- **TCB blocking**
  - Prior to OTE all application code ran under a single TCB, the CICS quasi-reentrant (QR) TCB
  - Non-CICS APIs are disallowed because they are likely to issue MVS WAITs which would bring CICS to a halt by **blocking** the QR TCB
- **Storage keys**
  - Applications running USERKEY cannot access storage acquired via MVS Getmain or MVS Load
  - MVS Getmain and Load determine the required key from TCBPKF not the PSW
  - Not a problem prior to OTE as use of MVS services was outlawed

Notes

- Non-CICS APIs are disallowed prior to OTE because all applications ran under the QR TCB. A non-CICS API may involve performing an operating system wait for example. An MVS wait issued under the QR TCB would halt the QR TCB and hence the whole of CICS until the wait was completed. Such a request is described as a 'blocking' request as it blocks the TCB.

- This is why for example the Resource Manager Interface (RMI) exists. It gives the Resource Managers (via their adapters) a way of offloading the request onto another TCB. This other TCB is then used to access the Resource Manager and non-CICS services can be used on this TCB without affecting CICS.

- The introduction of storage protection and use of USERKEY (key 9) also causes problems with use of MVS services. For example if an application running userkey were to issue an MVS GETMAIN or MVS LOAD of a program, the storage getmained, or the program loaded, would be in key 8 storage and inaccessible by the application. This is because although the application is running with a PSW key of 9, it is running under the QR TCB which has a key of 8 (TCBPKF=8). MVS uses the TCBPKF key rather than the PSW key to determine what key storage is required. Prior to OTE use of MVS services was not allowed (and would not have worked for this reason).

Problems addressed by OTE (2 of 3)

- **CICS code**
  - The CICS API code was not threadsafe, it all relied on QR TCB to provide the serialisation to shared resources.
- **Other products**
  - Do they cope with CICS applications running under multiple TCBs ?
- **TCB Failures**
  - An operating system abend on an open TCB should not cause CICS to terminate
- **Recovery from failures**
  - Non CICS ESTAE and ESPIE exits will intercept abends or program checks before the CICS ESTAE, when control is in non-CICS code or CICS is called from this environment. CICS recovery must cope with this, and OTE must ensure CICS recovery is driven

Notes

- None of the CICS code that implements the API was originally threadsafe. It could not be executed concurrently on multiple TCBs, but relied on QR TCB for serialisation.
- CICS will be changed over time so that more and more of the API can be executed concurrently on open TCBs without the need to switch to QR TCB for serialisation.
- Not just CICS is affected. We must ensure other products such as MVS itself, the Language Environment and others can cope with CICS applications running on multiple TCBs.
- Prior to OTE, all TCBs in CICS were deemed to be essential. If a TCB terminates, then CICS terminates. With OTE, open TCBs must be allowed to terminate without affecting the rest of CICS.
- CICS Recovery must also be able to cope with non-CICS recovery exits being driven ahead of it on open TCBs. Also, when CICS is called from an environment where a non-CICS recovery exit is active, we must ensure that CICS recovery is stacked on top when running in CICS code.

Problems addressed by OTE (3 of 3)

- **Task Cancel**
  - Pre-OTE, CICS dispatcher only allowed a task to be purged or forcepurged when in a wait, i.e. a CICS wait, not in a running state.
  - A CICS task running on an open TCB issuing an MVS service that issued an MVS wait would be in a running state according to the CICS dispatcher
  - CICS must allow task cancel when in a running state in non-CICS code
- **Performance**
  - OTE changes must not degrade CICS performance

Notes

- Before OTE, the CICS dispatcher did not allow a task to be cancelled in what it deems to be a 'running' state. Typically a task is only allowed to be cancelled when it is in a CICS wait.
- When a CICS task is executing under an open TCB and it is not in CICS code, that code may issue an MVS service for example that issues an MVS WAIT. The CICS dispatcher will be unaware of the MVS WAIT, and its state will show the task to be in a running state.
- CICS must allow CICS tasks to be cancelled in a running state when they are not executing CICS code.
- Changes made to the CICS infrastructure to support OTE, for example making code threadsafe, must not have a detrimental impact on performance - especially for CICS workload not using OTE.

Enhancements provided by OTE (1 of 3)

- **A CICS task can have one or more Open TCBs**
  - Requires that application code and CICS code running under an open TCB is threadsafe
  - A blocking call under its own TCB will halt just that TCB not CICS
  - Requires subspaces to work on more than one TCB for transaction isolation
  - Userkey applications will get an Open TCB mode that creates a key9 TCB
    - Application will run under a key9 TCB when running in userkey
    - Application will run under a key8 TCB when running in CICS key
  - CICS will switch between the QR TCB, the open key8 TCB and the open key9 TCB as required
- **Open TCBs are non-essential**
  - Open TCBs can be terminated without affecting the rest of CICS

Notes

- The main function provided by OTE is the ability to run application code under its own open TCB thereby freeing the application from the constraints applied now, due to it running under the CICS QR TCB.
- From CICS TS 3.1 onwards applications can issue API calls that block the open TCB, but applications have to be threadsafe as multiple application instances will be able to run concurrently.
- CICS provides the application with several open TCBs (each of a different mode) if necessary, to fulfill its requirements. CICS does the necessary switching between TCBs, not the application.
- Open TCBs can terminate normally and abnormally without affecting the rest of CICS.

Enhancements provided by OTE (2 of 3)

- **Exploitation of Open TCBs by Resource Managers**
  - New options on ENABLE command for Task Related User Exits (TRUEs)
  - Allows TRUEs to be invoked on an Open TCB
  - Eliminates the need for Adapters to manage their own set of private TCBs
  - If calling applications are threadsafe this means elimination of TCB switches
  - Elimination of large number of TCB switches will produce a big performance gain
- **CICS RMI and CICS Global User Exit interface code made threadsafe**
- **Code for some common CICS API commands made threadsafe**
  - Over time more and more commands will be made threadsafe

Notes

- The RMI has been enhanced to allow Task Related User Exits (TRUEs) to specify that they wish to be invoked on an open TCB. This allows Resource Manager adapters such as the CICS-DB2 Attachment Facility to avoid having to manage a private set of TCBs onto which requests to the Resource Manager are offloaded. Instead the Resource Manager can be called on the open TCB. This avoids two TCB switches per Resource Manager request and can produce a significant performance improvement.
  - note this implies that the TRUE has to be threadsafe if it is to use the new function
- The CICS code that implements the RMI, and implements the global user exit interface has been made threadsafe, so that it can be called on an open TCB.
- CICS API commands are being made threadsafe over time, focussing on those used most commonly, and those commonly used in situations where otherwise TCB switching would occur.

Threadsafe CICS API

- Commands made threadsafe in CICS TS 1.3:
  - ABEND, ADDRESS, ASSIGN, HANDLE ABEND, HANDLE AID, HANDLE CONDITION, IGNORE, PUSH, POP, ENTER TRACENUM, MONITOR, GETMAIN, FREEMAIN, INQUIRE EXITPROGRAM, INQUIRE TASK (without list), LINK, LOAD, RELEASE, RETURN, XCTL, all Temporary Storage commands
- Commands made threadsafe in CICS TS 2.2:
  - DEQ, ENQ, INQUIRE/SET DB2CONN, INQUIRE/SET DB2ENTRY, INQUIRE/SET DB2TRAN, SUSPEND, WAIT EXTERNAL
- Commands made threadsafe in CICS TS 2.3:
  - ASKTIME, FORMATTIME, CHANGE TASK, all Document commands
- Commands made threadsafe in CICS TS 3.1:
  - all WEB commands, CONTAINER commands, URIMAP commands and SOAP commands
- Commands made threadsafe in CICS TS 3.2:
  - File Control API commands for local VSAM and VSAM RLS files, MQ commands, Journal commands

Notes

This shows the sequence of introduction of threadsafety to the CICS API

- The following EXEC CICS API and SPI commands in CICS TS 1.3 were made threadsafe and so can be executed under an open TCB:
  - abend, address, assign, handle abend, handle aid, handle condition, ignore, push, pop, enter tracenum, monitor, getmain, freemain, inquire exitprogram, inquire task (without list), link, load, release, return, xctl, all temporary storage commands
- The following commands were threadsafe in CICS TS 2.2:
  - deq, enq, inquire/set db2conn, inquire/set db2entry, inquire/set db2tran, suspend, wait external
- The following commands were made threadsafe in CICS TS 2.3:
  - asktime, formattime, change task, all document commands
- The following commands were made threadsafe in CICS TS 3.1:
  - all WEB commands, CONTAINER commands, URIMAP commands and SOAP commands
- The following commands have been made threadsafe in CICS TS 3.2:
  - File Control API commands for local VSAM and VSAM RLS files, MQ commands, Journal commands

Enhancements provided by OTE (3 of 3)

- **XPI enqueue/dequeue function added**
  - To aid making Global User Exits threadsafe
- **Support for task cancel and runaway when in 'running' status**
- **RDO and SPI changes**
  - Option to define a program as quasi-reentrant or threadsafe
  - Option to define whether a program uses CICSAPI or OPENAPI
- **System definition changes**
  - To limit the number of open TCBs
  - To force use of QR TCB if required

Notes

- To help global user exits become threadsafe, functions have been added to the XPI. The Enqueue and Dequeue functions use the services of the NQ domain to provide a serialisation capability for global user exits.
- The CICS kernel and dispatcher have been changed to allow CICS tasks to be cancelled (and subject to runaway) when in running state if they are not executing CICS code. This capability is used in CICS TS 1.3 to provide runaway and task cancel support for JVM programs.
- A program can indicate to CICS that it is threadsafe, and that it requires an open TCB to run on, via options on the program definition.
- Options on inquire and set program allow the new program definition keywords to be manipulated.
- At the CICS system level, there are options in the SIT to limit the number of open TCBs in the system, and to force all programs onto the QR TCB if necessary. Again, there are SPI changes to inquire and set system to manipulate the new keywords.

OTE Externals - Program definitions & Program auto-install

- **CONCURRENCY (QUASIRENT | THREADSAFE)**
  - Attribute of program. Applies to applications, TRUEs, GLUEs, URMs, PLT
    - QUASIRENT (the default) means the code must run under QR TCB
    - THREADSAFE means the code may run under QR or an open TCB
      - We say the program can 'float' between TCBs as required
      - Must be LE enabled or assembler
  - Tells CICS if **application logic** is threadsafe or not
  - CICS handles threadsafe issues for the CICS API.

- **API (CICSAPI | OPENAPI) - implemented in CICS TS 3.1**
  - Attribute of program. Applies to applications, TRUEs, URMs, PLT (ignored for GLUEs)
  - CICSAPI (the default) means the program only uses CICS permitted interfaces
  - OPENAPI means the program requires an Open TCB to use other APIs
    - OPENAPI requires CONCURRENCY(THREADSAFE)

Notes

- The CONCURRENCY keyword on the program definition defines whether the application program code is threadsafe or not. The default is QUASIRENT meaning that the program is quasi-reentrant and will be always be executed on the QR TCB. This is how all programs executed prior to OTE.
- A value of THREADSAFE defines that the program is threadsafe and hence can be executed concurrently on multiple TCBs. It can be run on the QR TCB or on an open TCB. We say that the program is capable of 'floating' between TCBs as required, meaning that it can be executed on whatever TCB is running at the time. Apart from assembler programs, programs must be LE enabled in order to float between TCBs.
- The CONCURRENCY attribute applies to application programs, task related user exits, global user exits, user replaceable modules and PLT programs. (Task related user exits can override the setting on the program definition via options on the ENABLE command).
- In setting the concurrency attribute you are tell CICS whether or not the application code is threadsafe or not. CICS itself handles whether the CICS commands issued by the application are threadsafe or not.
- The API keyword was added in CICS TS 3.1 and defines whether the program is to use the standard CICS APIs or open APIs. The OPENAPI keyword defines that the program must be given its own open TCB to execute on and requires the program to be threadsafe.

OTE Externals - CICS environment variable for program definitions

- **Environment variable CICSVAR - implemented in CICS TS 3.1**
  - Allows concurrency and api attributes to be associated with program load module (however increases size of load module)
  - Overrides program definition (CEDA or autoinstalled)
  - Can be used for all LE supported languages but not non LE assembler and not Java
- **CICSVAR=QUASIRENT**
  - equivalent to CONCURRENCY(QUASIRENT) API(CICSAPI)
- **CICSVAR=THREADSAFE**
  - equivalent to CONCURRENCY(THREADSAFE) API(CICSAPI)
- **CICSVAR=OPENAPI**
  - equivalent to CONCURRENCY(THREADSAFE) API(OPENAPI)

Notes

- CICS provides an environment variable called CICSVAR to allow the CONCURRENCY and API program attributes to be closely associated with the application program itself by using the ENVAR runtime option. Whilst it may be used in CEEDOPT CSECT to set an installation default, it is most useful to be set in a CEEUOPT CSECT linkedited with an individual program, or set via a #pragma statement in the source of a C/C++ program, or set via a PLIXOPT statement in a PL/I program.
- For example, when a program has been coded to threadsafe standards it can be defined as such without having to change an RDO program definition, or adhere to an installation defined naming standard to allow a program autoinstall exit to install it with the correct attributes. CICSVAR can be used for LE assembler, PLI, Cobol and C/C++ (XPLINK and non-XPLINK) programs that have been compiled using a LE-supported compiler.
- CICSVAR cannot be used for non-LE assembler programs, XPLINK programs or java programs.
- Use of CICSVAR will override the settings on a program definition installed via standard RDO interfaces, or via program autoinstall. Prior to the program being run for the first time, an inquire program command will show the keyword settings from the program definition. Once the application has been run once, an INQUIRE PROGRAM command will show the settings with any CICSVAR overrides applied. CICSVAR can take one of three values, QUASIRENT, THREADSAFE or OPENAPI.

OTE Externals - Environment variable (continued)

- **Syntax**
  - In a C or C++ program at the start before any other C statements
    - #pragma runopts(ENVAR(CICSVAR=THREADSAFE))
  - In a PL/I program following the PL/I MAIN procedure statement
    - DCL PLIXOPT CHAR(25) VAR STATIC EXTERNAL INIT('ENVAR(CICSVAR=THREADSAFE)');
  - For any LE-supported language the ENVAR can be coded in a CEEUOPT CSECT and linkedited with the program
    - CEEUOPT CSECT
CEEUOPT AMODE ANY
CEEUOPT RMODE ANY
CEEXOPT ENVAR=('CICSVAR=THREADSAFE')
END

Notes

- CICSVAR statements can be placed in the source of C, C++ or PL/I programs. They result in the compiler producing an LE runtime options CSECT.
- Alternatively the statements can be coded in a separate CEEUOPT CSECT which is assembled and linkedited into a PDS. The CSECT can then be linkedited with the desired application programs.

OTE Externals - COBOL Static and Dynamic Calls

- **All programs called statically or dynamically from a CONCURRENCY(THREADSAFE) program must also be threadsafe.**
  - EXEC CICS LINK increments the stack level
  - Static or Dynamic calls do not increment stack level, so not regarded by CICS as change of program e.g. COBOL calls
- **When a DB2 call is issued from a static or dynamically called routine, the CONCURRENCY definition of the program issuing the static/dynamic call determines the mode used after the DB2 call completes.**

Notes

- If you define a program with CONCURRENCY(THREADSAFE), all routines which are statically or dynamically called from that program (for example, Cobol routines) must also be coded to threadsafe standards.
- When an EXEC CICS LINK command is used to link from one program to another, the program link stack level is incremented. However, a routine which is statically called, or dynamically called, does not involve passing through the CICS command level interface, and so does not cause the program link stack level to be incremented. With Cobol routines, for a static call, a simple branch and link is involved to an address resolved at linkedit time. For a dynamic call, there is a program definition involved, this is required only to allow Language Environment to load the program. After the load, a simple branch and link is executed. When a routine is called by either of these methods, CICS does not regard this as a change of program. The program which called the routine is still considered to be executing, and so the program definition for that program is still considered to be the current one.
- If the program definition for the calling program states CONCURRENCY(THREADSAFE), the called routine must also comply with this specification. Programs with the CONCURRENCY(THREADSAFE) attribute remain on an open TCB when they return from a DB2 call, and this is not appropriate for a program which is not threadsafe.
  - **For example, consider the situation where the initial program of a transaction, program A, issues a dynamic call to program B, which is a Cobol routine. Because the CICS command level interface was not involved, CICS is unaware of the call to program B, and considers the current program to be program A.**
  - **Program B issues a DB2 call. On return from the DB2 call, CICS needs to determine whether the program can remain on the open TCB, or whether the program must switch back to the QR TCB to ensure threadsafe processing. To do this, CICS examines the CONCURRENCY attribute of what it considers to be the current program, which is program A. If program A is defined as CONCURRENCY(THREADSAFE), then CICS allows processing to continue on the open TCB. In fact program B is executing, so if processing is to continue safely, program B must be coded to threadsafe standards.**

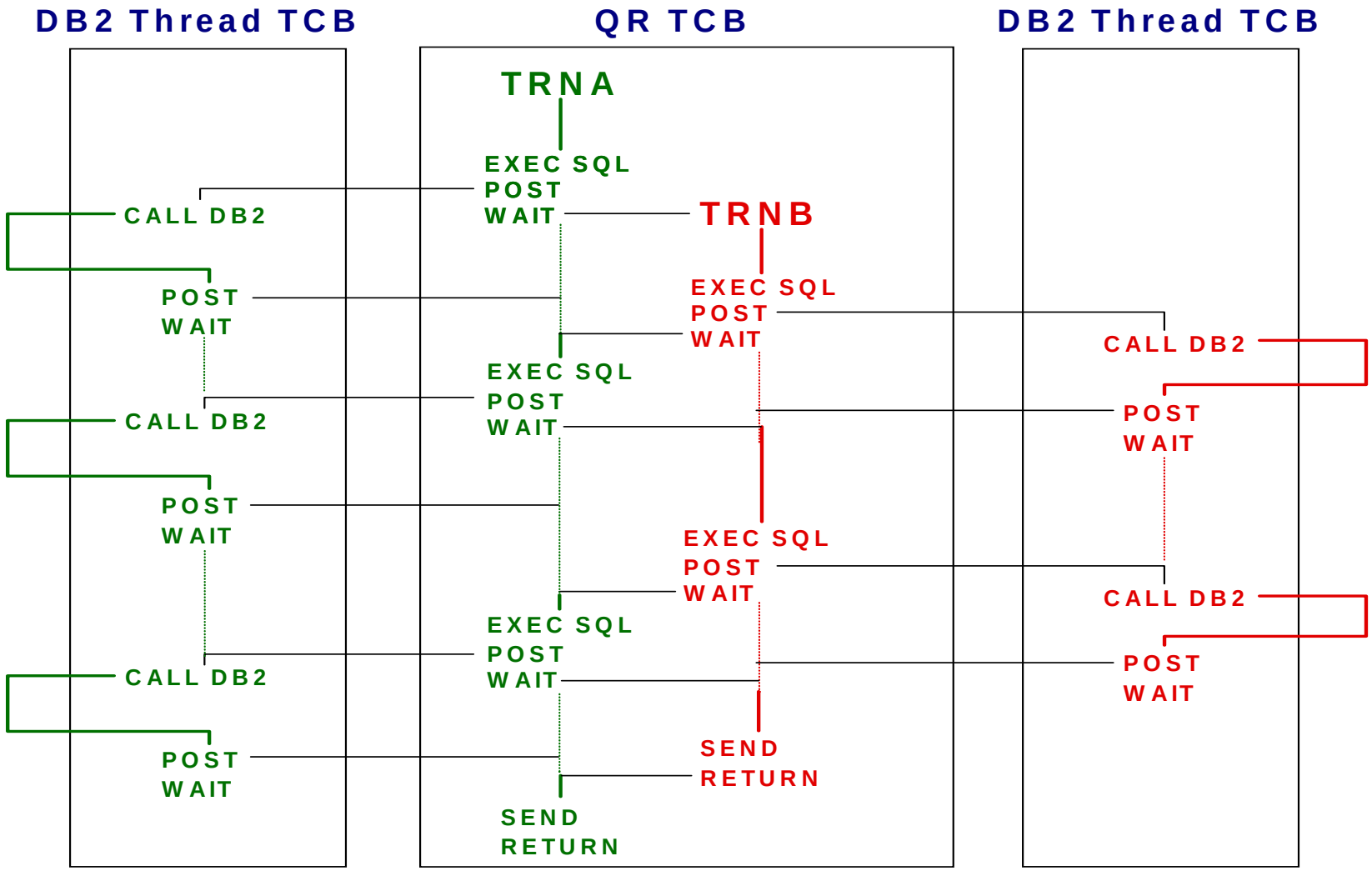
OTE Externals - Enable options for TRUEs & GLUEs

- **Options on ENABLE for a TRUE only (CICS TS 2.2 & above)**
  - ENABLE options for a TRUE override the program definition.
- **QUASIRENT | THREADSAFE**
  - QUASIRENT (the default) means the TRUE will be invoked on QR TCB
  - THREADSAFE means the TRUE can be invoked on QR or an open TCB
- **OPENAPI**
  - Means the TRUE will be invoked on an open TCB
  - Means the Resource Manager adapter does not need private TCBs
- **CICS-DB2 TRUE uses OPENAPI in CICS TS 2.2 and above**
  - when connected to DB2 V6 or higher
- **z/OS V1R7 Communications Server TRUE can use OPENAPI**
- **CICS-MQ TRUE uses OPENAPI in CICS TS 3.2**
- **New options on ENABLE for a GLUE (CICS TS 3.2)**
  - QUASIRENT or THREADSAFE as above, but not OPENAPI
  - Allows GLUEs enabled at phase 1 PLT to override the system auto-installed default of QUASIRENT

Notes

- For Task Related User Exits (TRUEs) there are keywords on ENABLE that allow it to override the CONCURRENCY and API keywords on the program definition. This is required because a resource manager adapter needs to decide at runtime how a TRUE is going to be invoked. This for example could be based on what release the backend resource manager is running and hence whether it is capable of handling the new TCBs or if it requires its own private set of TCBs.
- When activated, the QUASIRENT, THREADSAFE and OPENAPI keywords have the same meanings as on the program definition.
- The CICS-DB2 TRUE is enabled as OPENAPI when connected to DB2 V6 or higher
- The z/OS V1R7 Communications Server TRUE that implements the EZA sockets API can be configured to use OTE and hence be enabled as OPENAPI.
- New in CICS TS 3.2, the CICS-MQ TRUE is enabled as OPENAPI.
- New in CICS TS 3.2 is the ability for GLUEs to override their program definition and specify QUAISRENT or THREADSAFE (but not OPENAPI) when they are enabled. This was introduced to allow GLUEs that are enabled at phase 1 PLT to be enabled as threadsafe since CICS system autoinstalls these exits with program definitions that specify quasirent, and there was no way to override this.

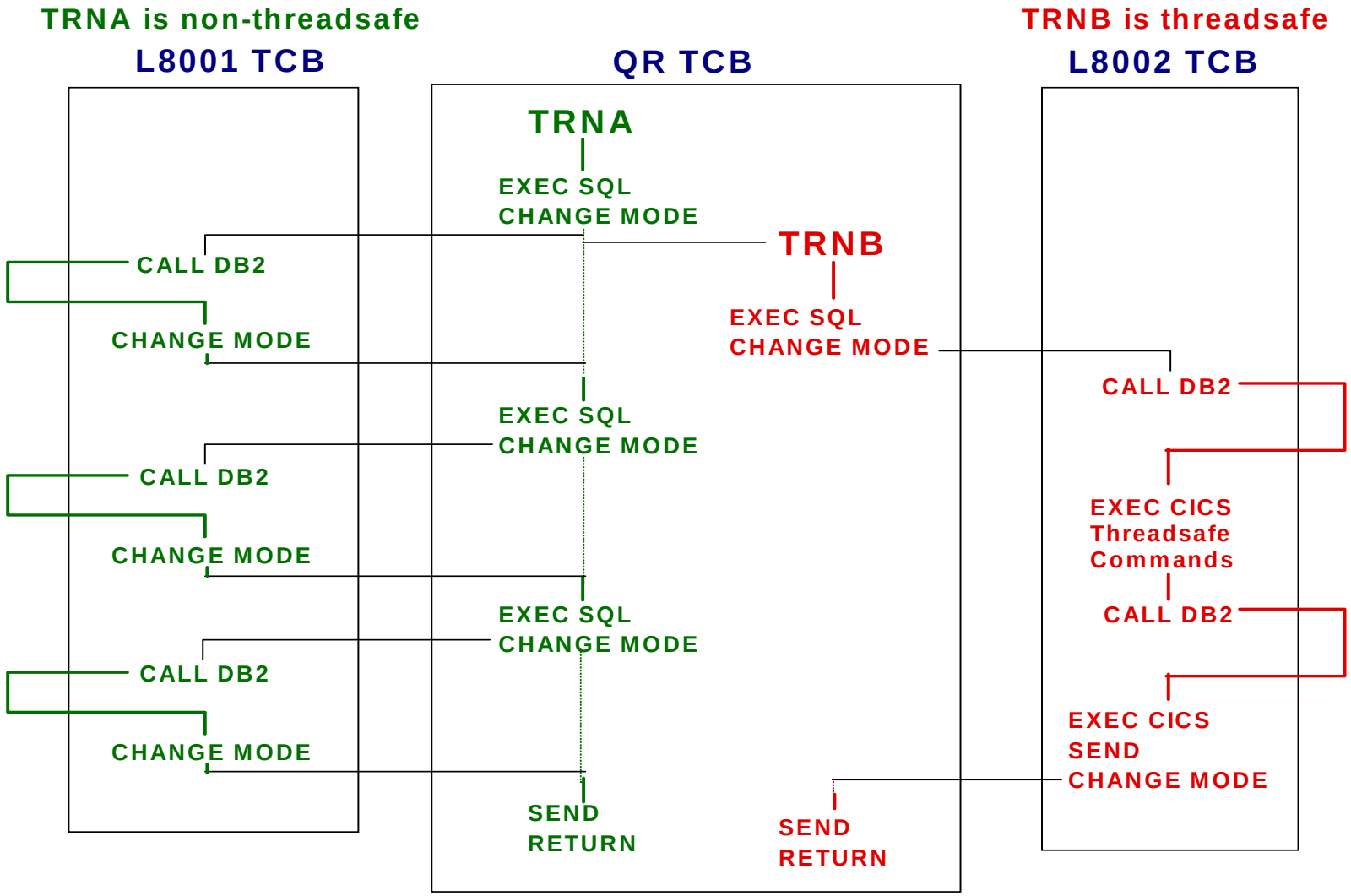
CICS-DB2 Transactions in CICS TS 1.3



Notes

- The picture shows what happened in CICS TS 1.3 when running CICS DB2 applications containing both EXEC CICS commands and EXEC SQL commands. In particular it shows the TCB switching that goes on when executing DB2 calls.
- The CICS-DB2 Attach has to manage its own set of private TCBs onto which the DB2 request is offloaded. A DB2 Attach TCB is a TCB under which calls are made to DB2 as we cannot use the QR TCB, because it would be blocked by the DB2 request.
- For each EXEC SQL request there are two TCB switches, one to offload the request onto the DB2 Attach TCB, another TCB switch to return control to the application on the QR TCB when the DB2 request has completed.

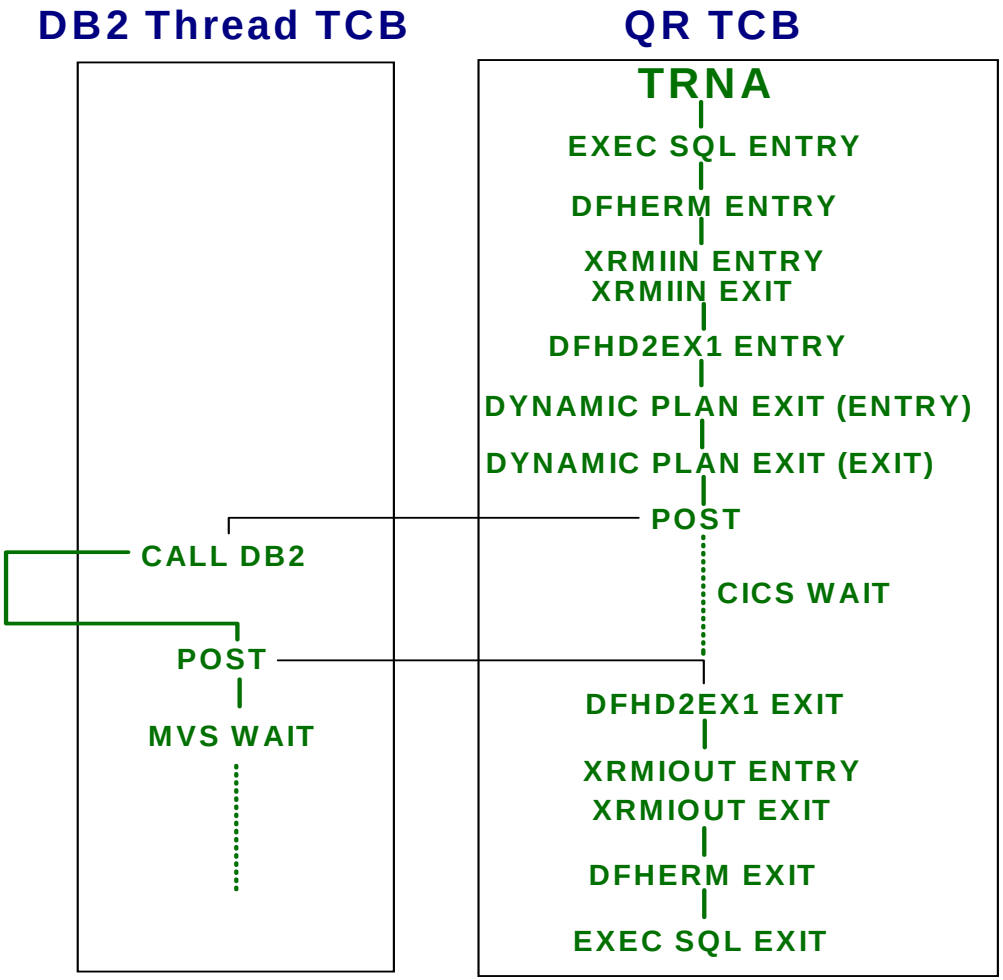
CICS-DB2 Transactions in CICS TS 2.2 and higher



Notes

- The picture shows how the CICS-DB2 Attachment facility was upgraded in CICS TS 2.2 to take advantage of OTE and use L8 open TCBs to call DB2 (when using DB2 V6 or higher) rather than privately managed DB2 Attach TCBs.
- TRNA executes a non threadsafe application program. Here the amount of TCB switching is the same as the previous release. Instead of switching to a DB2 Attach TCB, we use an L8 TCB. Switching is achieved via use of a dispatcher domain Change_mode request, instead of use of WAIT/POST logic as in previous releases.
- TRNB executes an application that is threadsafe, and has been defined to CICS with CONCURRENCY(THREADSAFE). When executing the first EXEC SQL request, the TRUE is invoked on an open TCB, and that TCB is used to call DB2.
- On return from the RMI, because the application is defined as threadsafe, we stay on the open TCB and return to the application on it. Any threadsafe CICS commands issued by the application will be executed on the L8 TCB, as will any further SQL requests. For the EXEC CICS SEND command (which is non threadsafe) we switch back to the QR TCB. Control will stay on the QR TCB until another DB2 request is issued. In this example there are no more DB2 requests. The EXEC CICS RETURN is executed on the QR TCB.
- Not shown in the diagram is the end of task syncpoint which will result in four TCB switches for the two phase commit protocol, i.e. switch to L8 to issue PREPARE to DB2, then back to QR, then switch to L8 to issue COMMIT to DB2, then back to QR.

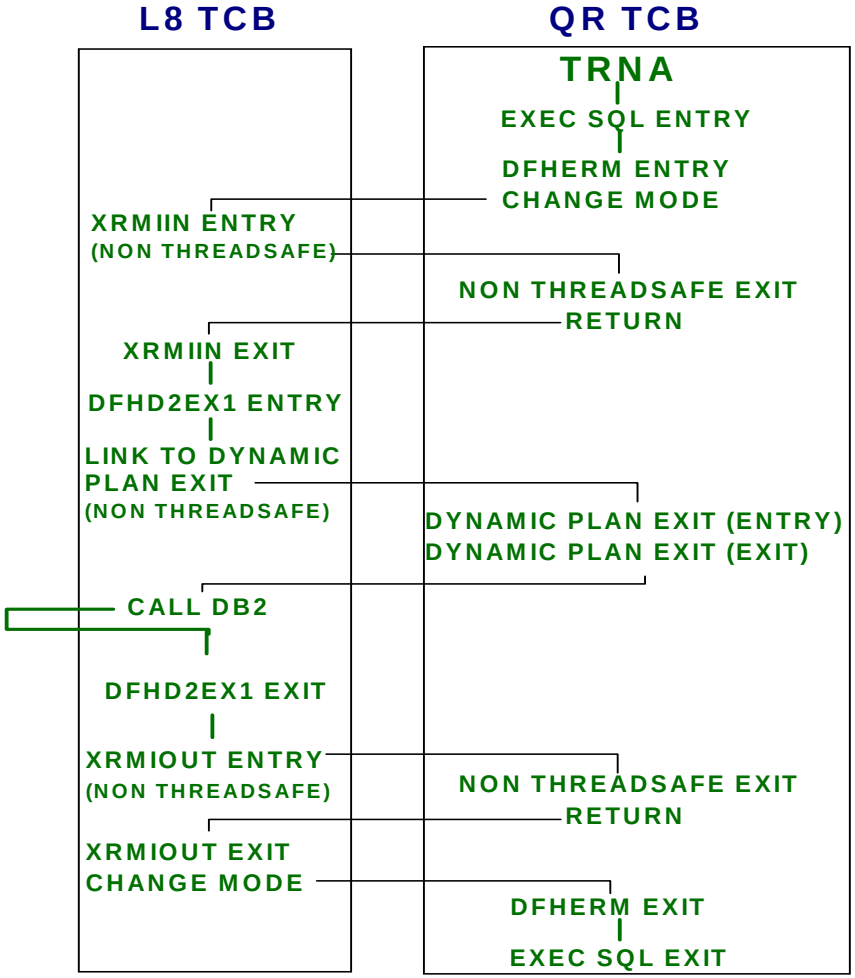
Global User Exits and a CICS TS 1.3 DB2 transaction



Notes

- The picture shows that in CICS TS 1.3 dynamic plan exits and Global user exits always ran on the QR TCB, so the concurrency parameter of the global user exit program, or the dynamic plan exit does not affect the amount of TCB switching.

GLUEs and a CICS TS 2.2+ DB2 transaction - (the Worst case)



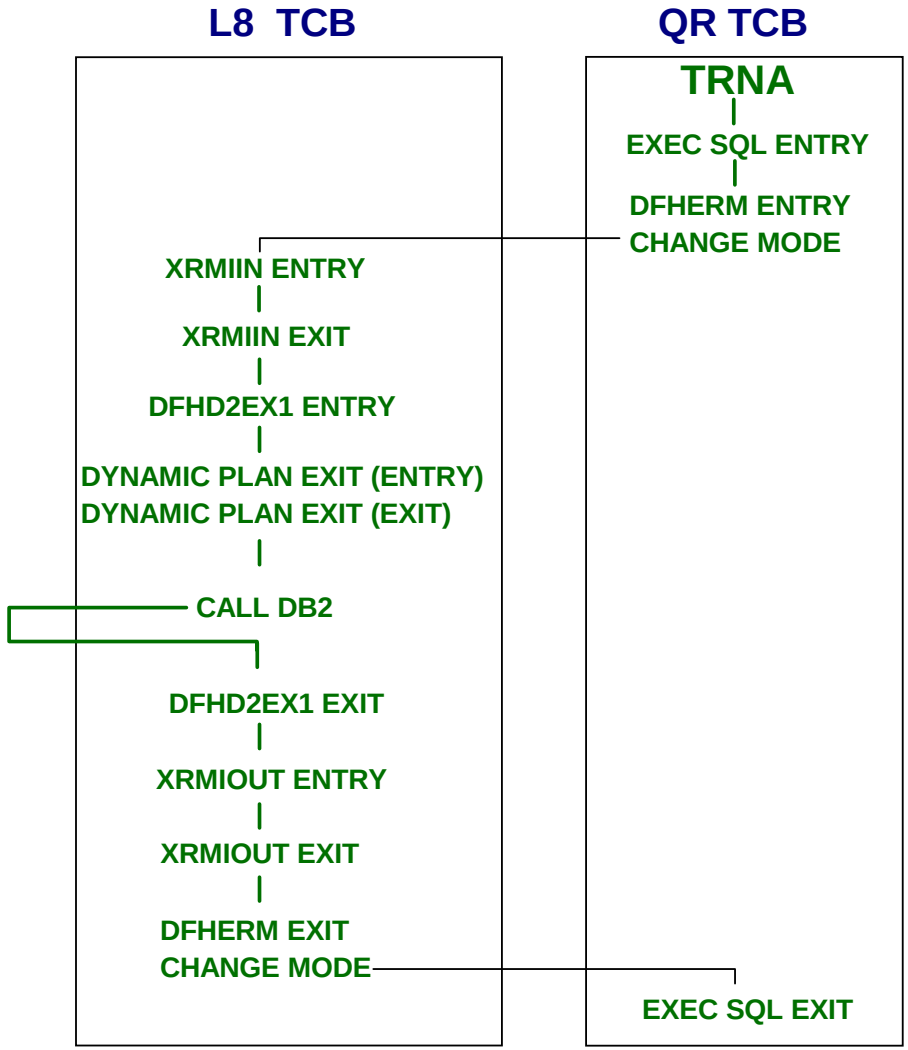
- **Program defined Quasirent**
- **Exits enabled Quasirent**

Notes

- The picture shows what happens if you move to CICS TS 2.2 or higher and do not make any changes to Global user exits. The program definition for an Global user program, like an application program, defaults to quasirent. This means the Global user exit must be run on QR TCB.

- With CICS TS 2.2 and higher, we have already switched to the L8 TCB in preparation of calling the CICS-DB2 TRUE, before we invoke any global user exits or a dynamic plan exit. When invoking these exits we have to switch back to QR, and then back to the L8 TCB afterwards. This results in far more TCB switching than in previous releases.

GLUEs and a CICS TS 2.2+ DB2 transaction - (a Better case)

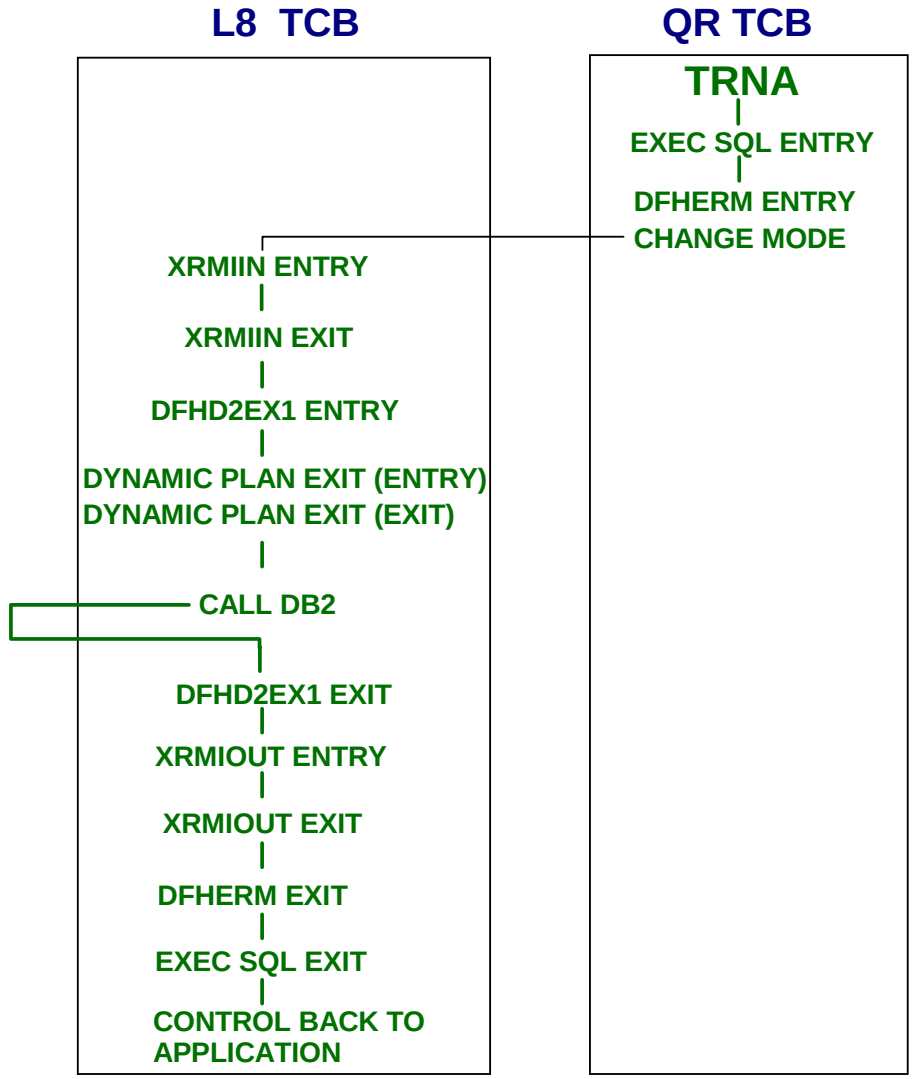


- **Program defined Quasirent**
- **Exits Enabled Threadsafe**

Notes

- The picture shows what happens when Global user exits, and dynamic plan exits have been checked for threadsafety and are defined to CICS as threadsafe. They can now run on an L8 TCB. The amount of TCB switching for a quasirent application is back to that of CICS TS 1.3, i.e. two TCB switches per EXEC SQL request.

GLUEs and a CICS TS 2.2+ DB2 transaction - (the Best case)



- **Program defined Threadsafe**
- **Exits Enabled Threadsafe**

The task will remain on the L8 TCB until it issues a non-threadsafe command or completes.

A mode switch will return the task to the QR to complete task termination.

Notes

- The picture shows the best case as far as the amount of TCB switching is concerned, i.e. when the application and exits are defined as threadsafe.

Detecting Global Exit impact in a trace

```
00130 QR    AP 00E1 EIP   EXIT  LINK OK                    REQ(00F4) FIELD-A(00000000 ....) FIELD-B(00000E02 ....)
                                                                    RET-9CDED516 7:42:55.328 0.0000075 =00029=
00130 QR    AP 2520 ERM   ENTRY COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL) RET-9CDEFB42 7:42:55.328 0.0000182 =00030=
00130 QR    DS 0002 DSAT  ENTRY CHANGE_MODE MODENAME_TOKEN(0000000A)    RET-8009625A 7:42:55.328 0.0000072 =00031=
00130 L800D DS 0003 DSAT  EXIT  CHANGE_MODE/OK                         RET-8009625A 7:42:55.328 0.0000022 =00041=
00130 L800D AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM JGXRMIIN AT EXIT POINT XRMIIN
                                                                    RET-800978C8 7:42:55.328 0.0000076 =00043=
00130 L800D AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM JGXRMIIN WITH RETURN CODE 0
                                                                    RET-800978C8 7:42:55.328 0.0000020 =00047=
00130 L800D AP 3180 D2EX1 ENTRY APPLICATION REQUEST - EXEC SQL INSERT  RET-8009640E 7:42:55.328 0.0000009 =00049=
00130 L800D AP 3250 D2D2  ENTRY DB2_API_CALL CSUB_TOKEN(1C8FDC30)      RET-9B4E7F52 7:42:55.559 0.2306609*=00056=
00130 L800D AP 3251 D2D2  EXIT  DB2_API_CALL/OK                        RET-9B4E7F52 7:42:55.565 0.0003492 =00086=
00130 L800D AP 3181 D2EX1 EXIT  APPLICATION-REQUEST SQLCODE 0 RETURNED ON EXEC SQL INSERT
                                                                    RET-8009640E 7:42:55.565 0.0000113 =00087=
00130 L800D AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM JGXRMIOT AT EXIT POINT XRMIOUT
                                                                    RET-80097950 7:42:55.565 0.0000120 =00088=
00130 L800D DS 0002 DSAT  ENTRY CHANGE_MODE MODENAME(QR)               RET-9B0E9670 7:42:55.565 0.0000021 =00089=
00130 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                         RET-9B0E9670 7:42:55.568 0.0033349*=00095=
00130 QR    AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM JGXRMIOT WITH RETURN CODE 0
                                                                    RET-80097950 7:42:55.568 0.0001310 =00102=
00130 QR    DS 0002 DSAT  ENTRY CHANGE_MODE MODENAME(L8)               RET-9B0E9D6A 7:42:55.568 0.0000034 =00103=
00130 L800D DS 0003 DSAT  EXIT  CHANGE_MODE/OK                         RET-9B0E9D6A 7:42:55.575 0.0000696 =00119=
00130 L800D AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM JRMIOUT2 AT EXIT POINT XRMIOUT
                                                                    RET-80097950 7:42:55.575 0.0000061 =00120=
00130 L800D AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM JRMIOUT2 WITH RETURN CODE 0
                                                                    RET-80097950 7:42:55.575 0.0000186 =00121=
00130 L800D DS 0002 DSAT  ENTRY CHANGE_MODE MODENAME_TOKEN(00000001)    RET-8009704C 7:42:55.575 0.0000004 =00124=
00130 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                         RET-8009704C 7:42:55.575 0.0000075 =00134=
00130 QR    AP 2521 ERM   EXIT  COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL) RET-9CDEFB42 7:42:55.575 0.0000023 =00135=
00130 QR    AP 00E1 EIP   ENTRY READ                       REQ(0004) FIELD-A(00206A10 ....) FIELD-B(0900060 ....)
                                                                    RET-9CDE3196 7:42:55.575 0.0000133 =00136=
00130 QR    AP 04E0 FCFR  ENTRY READ_INTO FILE_NAME(VSAMFIL1) BUFFER_ADDRESS(1CB4D778) BUFFER_LENGTH(1409) ENVIRON
                      MT_ID(00000000) RECORD_ID_ADDRESS(1CB4D778) RECORD_ID_LENGTH(45) GENERIC(NO) KEY_COMPARISON
                      (EQUAL) READ_INTEGRITY(FCT_VALUE) RECORD_ID_TYPE(KEY) CONDITIONAL(NO) BYPASS_SECURITY_CHECK
                           (NO)                                     RET-9AF5C4FE 7:42:55.575 0.0000077 =00137=
```

Notes

- The sample trace on this slide shows user exit modules being called for XRMIIN and XRMIOUT.
- The trace was formatted using the SHORT option in order to see not only the R14 values, but the timestamps and interval between the entries. This same information is available with full trace formatting but would take additional presentation space.
- Trace number 29 shows the SQL call being issued from the COBOL application. Trace 30 shows the entry into DFHERM which is followed by a change_mode to an L8 TCB (in this case L800D). Trace 43 shows user exit JGXRMIIN being called at exit point XRMIIN. Since the exit executes on the L8 TCB, we know JGXRMIIN has been defined as threadsafe.
- Trace 56 shows the call being passed to DB2, with the response being traced in trace number 86.
- Upon completion of the actual call to DB2 the XRMIOUT exit module(s) are driven. In this example, there are actually two (2) modules which are called, JGXRMIOT and JRMIOUT2. The first module called (JGXRMIOT) is defined as QUASIRENT. Notice the change_mode to the QR TCB in trace entry 89. Once the exit completes, a change_mode is issued to return to the L8 TCB, see trace entry 119. Note the length of time the exit was in control -- refer to trace entry 102.
- The second XRMIOUT exit module (JRMIOUT2) is given control in trace entry 120. This module is defined threadsafe; notice the exit runs on the L8 TCB.
- When the SQL request is complete, the task is returned to the QR TCB - note the exit shown in trace 135. From there the application issues a file control read.

Checking Concurrency -- USING DFH0STAT

```
Applid IYK2Z2G1  Sysid JOHN  Jobname CI13JTD5    Date 02/07/2003  Time 15:19:20                CICS 6.2.0

User Exit Programs
                        <---- Global Area ---->   No.
  Program     Entry      Entry                 Use    of    Program    Program
    Name       Name       Name      Length     Count  Exits  Status   Concurrency
  ------------------------------------------------------------------------------
  DFHEDP     DLI                          0       0      0  Started  Quasi Rent
  JSTEXIT    JSTEXIT                      0       0      1  Started  Thread Safe
  JSTEXIT2   JSTEXIT2                     0       0      1  Started  Quasi Rent
  DFHD2EX1   DSNCSQL    DSNCSQL          16       1      0  Started  Quasi Rent

  Program     Entry           Concurrency                          <------- Task Related User Exit Options -------->
    Name       Name      API      Status      Qualifier  Length    Taskstart  EDF  Shutdown  Indoubt  SPI  Purgeable
  ------------------------------------------------------------------------------------------------------------------
  DFHEDP     DLI         Base   Quasi Rent                  284       No      No     No     No Wait   No      No
  JSTEXIT    JSTEXIT     Base   Thread Safe                   0       No      No     No     No Wait   No      No
  JSTEXIT2   JSTEXIT2    Base   Quasi Rent                    0       No      No     No     No Wait   No      No
  DFHD2EX1   DSNCSQL     Open   Thread Safe   DF2D          222       No      Yes    Yes       Wait   Yes     Yes


Global User Exits
    Exit     Program     Entry     <------- Global Area ------->   Number   Program    Program
    Name       Name       Name     Entry Name  Length  Use Count  of Exits  Status   Concurrency
  ------------------------------------------------------------------------------------------------
  XRMIIN     JSTEXIT    JSTEXIT                   0          0         1   Started  Thread Safe
  XRMIIN     JSTEXIT2   JSTEXIT2                  0          0         1   Started  Quasi Rent
  XRMOUT     JSTEXIT    JSTEXIT                   0          0         1   Started  Thread Safe
  XRMOUT     JSTEXIT2   JSTEXIT2                  0          0         1   Started  Quasi Rent
```

Notes

- An easy way to see what Global user exits are installed in a CICS region and their concurrency attribute is to request a global user exit report using the sample DFH0STAT statistics program. How to use DFH0STAT is documented in the CICS Performance Guide.
- The example shown is from CICS TS 2.2, and an example from CICS TS 3.2 is shown on the next slide.

Checking Concurrency -- USING DFH0STAT on CICS TS 3.2

```
Applid IYK2Z1V3  Sysid CJB3  Jobname CI07CJB3    Date 05/17/2007  Time 13:12:23                CICS 6.5.0                PAGE    57

  Program    Entry            Program    Concurrency                          <------- Task Related User Exit Options -------->
    Name      Name     API   Concurrency     Status     Qualifier  Length   Taskstart  EDF  Shutdown  Indoubt  SPI  Purgeable

  DFHEDP     DLI       Cics  Quasirent    Quasirent                  284       No      No     No     No Wait   No      No
  DFHMQTRU   MQM       Open  Threadsafe   Threadsafe   MQCD          224       No      No    Yes       Wait   Yes     No
  CBTRUE1    CBTRUE1   Open  Threadsafe   Threadsafe                   0       No      No     No     No Wait   No      No
  RMIIN      RMIIN     Cics  Threadsafe   Threadsafe                   0       No      No     No     No Wait   No      No
  RSINDI     RSINDI    Cics  Threadsafe   Threadsafe                   0       No      No     No     No Wait   No      No
  JTRUE1     JTRUE1    Open  Threadsafe   Threadsafe                   0       No      No     No     No Wait   No      No
  EZACIC01   EZACIC01  Open  Quasirent    Threadsafe                 788       No      No    Yes     No Wait   No     Yes
```

```
Applid IYK2Z1V3  Sysid CJB3  Jobname CI07CJB3    Date 05/17/2007  Time 13:12:23                CICS 6.5.0                PAGE    58

Global User Exits

    Exit     Program     Entry     <------- Global Area ------->   Number   Program    Program     Concurrency
    Name      Name       Name     Entry Name  Length  Use Count  of Exits  Status   Concurrency      Status

  XRSINDI    RSINDI     RSINDI                    0          0         1   Started  Threadsafe   Threadsafe

  XRMIIN     RMIIN      RMIIN                     0          0         1   Started  Threadsafe   Threadsafe
```

Notes

- The DFH0STAT reports have been enhanced in CICS TS 3.2 to provide an easier way to see the concurrency status of programs.

OTE Changes in CICS TS 3.1 : OPENAPI Programs (1 of 2)

- **New API Keyword on program definition**
  - Attribute of program. Applies to applications, TRUEs, URMs, PLT (ignored for GLUEs)
  - CICSAPI (the default) means the program only uses CICS permitted interfaces
  - OPENAPI means the program requires an Open TCB to use other APIs (OPENAPI requires CONCURRENCY(THREADSAFE))
- **New L9 TCB for running userkey OPENAPI programs**
  - For OPENAPI programs TCB key must match PSW key for non-CICS APIs to work (whereas CICS APIs run in either key irrespective of TCB key)
  - SIT parm MAXOPENTCBS now covers L8 and L9 TCBs
- **New CICS uses of L8 TCBs**
  - When accessing DOCTEMPLATEs or static HTTP responses held on HFS
  - WebServices and XML support implemented using cicskey OPENAPI programs

Notes

- A new API keyword on the program definition tells CICS whether the program is to use CICS APIs or whether it is potentially going to use other APIs as well as CICS APIs. The default is CICSAPI. OPENAPI is the trigger to tell CICS that this application must run on an open TCB (as opposed to THREADSAFE, which says the application is able to run on an open TCB if CICS code or TRUE code is executing on an open TCB at the time control is to be passed back to application code).
- OPENAPI programs must be threadsafe and defined to CICS as such.
- Because OPENAPI programs can potentially use non-CICS APIs, the key of the TCB becomes important, and the key of the TCB must match the execution key. This is in contrast with CICSAPI threadsafe programs that can execute in cics key or user key irrespective of the TCB key. This is because CICS services are implemented independent of the key of the TCB they are running on, whereas MVS services for example use the TCB key.
- The pool of open TCBs, whose size is specified via SIT parameter MAXOPENTCBS, now contains both L8 and L9 TCBs. Its size needs to be adjusted to cater for L9 TCBs and new CICS uses of L8 TCBs.
- CICS internally now uses L8 TCBs when accessing HFS to retrieve doctemplates stored on HFS, or to retrieve static HTTP responses held on HFS specified via the URIMAP resource definition. Additionally, support for WebServices and XML employ cicskey OPENAPI programs that likewise use L8 TCBs.

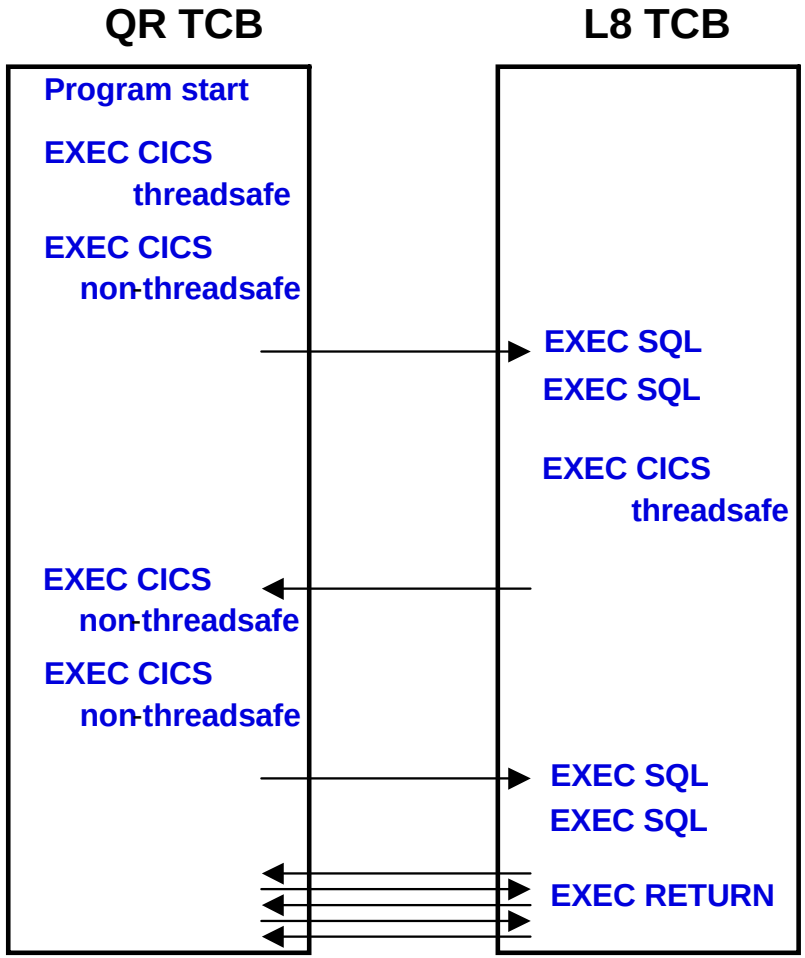
OTE Changes in CICS TS 3.1 : OPENAPI Programs (2 of 2)

- **OPENAPI programs - allows an application to specify it** must **run on an open TCB.**
  - User key OPENAPI programs run on an L9 TCB, CICS key OPENAPI on an L8 TCB
  - Non threadsafe CICS requests switch to QR then back to the open TCB when returning to the application
  - Application logic must be threadsafe.
  - Primary purpose is to allow application workloads to be moved off QR TCB to provide better exploitation of machine resources and achieve better throughput.
  - Allows application to use non-CICS APIs as well as CICS APIs
    - **Use of non-CICS APIs within CICS is entirely at the risk of the user. No testing of non-CICS APIs within CICS has been undertaken and is not supported by IBM service.**

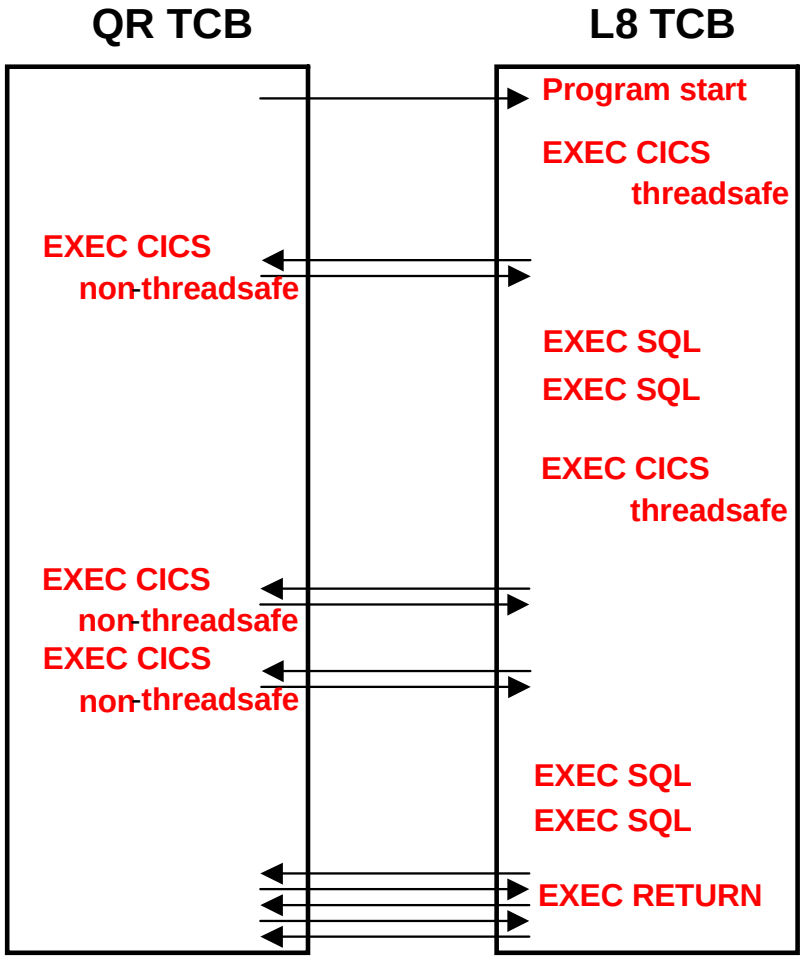
Notes

- OPENAPI programs must be coded to threadsafe standards and defined to CICS as such.
- The primary purpose for support of OPENAPI programs is to allow Application workloads to be moved off QR TCB. This allows better use of machine resources and better throughput can be achieved.
- Openapi programs can use non-CICS API requests because they are not running on the QR TCB. However use of non-CICS APIs within CICS is entirely at the risk of the user. No testing of non-CICS APIs within CICS has been undertaken and is not supported by IBM service.

CICS TS 3.1 - TCB switching



The program for transaction BLUE is defined THREADSAFE , API= CICSAPI



The program for transaction RED is defined THREADSAFE, OPENAPI, EXECKEY=CICS

Notes

- In CICS Transaction Server 3.1, a program may be defined with the OPENAPI keyword, to indicate it should execute on an OPEN TCB (L8 or an L9, based on the EXECKEY specification). However, it should be noted *all non-threadsafe commands will continue to be processed on the QR TCB.*
- In the diagram, the program used for transaction BLUE is defined as THREADSAFE, with API=CICSAPI. The execution of the task and TCB switching is the same as CICS Transaction Server 2.2 and 2.3. The program executes on the QR TCB until it issues an DB2 request, at which point it will switch to an L8 TCB. It will remain on the L8 until a non-threadsafe command is issued, causing it to switch back to the QR where it will remain until another SQL request is issued.
- The application used for transaction RED is defined THREADSAFE, with API=OPENAPI and EXECKEY=CICS. The program is given control on an L8 TCB. Each time it issues a non-threadsafe command it will switch to the QR TCB to process the command and then return control to the application on the L8 TCB. SQL commands are executed on the L8 TCB along with any threadsafe commands. Note that if there are many non-threadsafe commands, the overhead of switching can be greater than running with API=CICSAPI.
- Another important point to consider: if the program for transaction RED had been defined to run in EXECKEY USER, it will be given control on an L9 TCB, rather than an L8. The processing will be the same as noted above, EXCEPT when an SQL call is issued, when the task is switched to an L8 TCB. Upon completion of the SQL request, control will be returned to the application on the L9 TCB.

CICSAPI & THREADSAFE versus OPENAPI & THREADSAFE

- **NB: OPENAPI TRUEs must run CICS key on an L8 TCB**
  - A user key threadsafe OPENAPI program calling DB2 will switch from L9 to L8 then back to L9 for each DB2 request.
  - A user key threadsafe CICSAPI program running on an L8 TCB will remain on the L8 TCB to call DB2
- **Candidates for CICSAPI with THREADSAFE**
  - SQL programs with some non-threadsafe API
  - SQL programs with USER key
- **Candidates for OPENAPI with THREADSAFE**
  - Programs with threadsafe APIs only
  - SQL programs with CICS key
  - CPU intensive programs

Notes

- It should be noted that use of OPENAPI programs can increase TCB switching. For example Task Related User Exits (TRUEs) have to run in CICS key on L8 TCBs. Hence if an OPENAPI user key application calls DB2, then it will switch from the L9 TCB to the L8 TCB to call DB2, and then back to L9 to return to the application. User key DB2 applications are therefore best left defined as CICSAPI Threadsafe applications.
- CICS key OPENAPI programs will receive control on an L8 TCB and no switching will occur when calling an OPENAPI TRUE.
- If a non-threadsafe CICS command is issued from an OPENAPI program, then CICS switches to QR TCB to execute the CICS request, and then switches back to the open TCB when returning to the application program. This is because when you define a program as OPENAPI you are saying that it must run on an open TCB, whereas a CICSAPI threadsafe application is one that is happy to run on whatever TCB CICS deems fit. In this case a non-threadsafe CICS command will cause a switch to QR TCB, but then control will remain on the QR TCB when we return to the application, until something happens that forces a switch back to the open TCB, e.g. a DB2 call. Hence use of non-threadsafe CICS commands will cause more TCB switching for a OPENAPI threadsafe program than a CICSAPI threadsafe program.

OTE Changes in CICS TS 3.1 : C and C++ XPLINK support

- **XPLINK introduced in OS/390 V2.10**
  - Feature of z/OS that provides high performance subroutine call and return mechanism
  - Supported by C/C++ compiler of z/OS
  - Specified by C/C++ compiler option XPLINK
  - Previously not supported in CICS
- **XPLINK requires MVS LE rather than CICS LE**
  - Therefore application requires own TCB to run on
- **XPLINK programs execute like OPENAPI programs**
  - Run on an X8 or X9 TCB with MVS LE rather than L8 or L9 with CICS/LE
  - CICS detects at runtime that program compiled with XPLINK

Notes

- Extra Performance Linkage (from here on abbreviated to XPLink), is a z/OS feature which provides high performance subroutine call and return mechanisms. These result in short and highly optimized execution path lengths. XPLINK is a feature of z/OS and activated by use of a XPLINK compiler option. It is available for C and C++ applications only.
- XPLINK requires the use of MVS LE services rather than CICS LE services, and hence applications cannot run on QR TCB, but must have their own TCB to run on.
- An XPLINK program executes in the same way as an OPENAPI program except that it uses an X8 or X9 TCB (depending on the execution key required) as opposed to a L8 or L9 TCB.
- XPLINK programs must be threadsafe, and must be defined as CONCURRENCY(THREADSAFE). However they do not need to specify OPENAPI (and indeed it is ignored if specified). CICS detects from the application program load module at runtime that it has been compiled with the XPLINK option, and so will give control to it on an X8 or X9 TCB.
- The same rules apply to XPLINK programs as to OPENAPI programs with regards use of non-CICS APIs, and the same amount of switching occurs as with OPENAPI programs when invoking non-threadsafe CICS services.

OTE changes in CICS TS 3.2 : Journal Commands

- **EXEC CICS WRITE JOURNALNAME/JOURNALNUM is now a threadsafe command**
- **EXEC CICS WAIT JOURNALNAME/JOURNALNUM is now a threadsafe command**
- **DFHJCJCX WRITE_JOURNAL_DATA is now threadsafe**
  - Only one XPI command is now not threadsafe, i.e. DFHDUDUX TRANSACTION_DUMP

Notes

- In CICS TS 3.2 the EXEC API journal commands are threadsafe, as is the XPI command to write to a user journal from a global user exit.

OTE enhancements in CICS TS 3.2 : File Control – API and SPI

- **File Control API Commands - READ, WRITE, REWRITE, DELETE, UNLOCK, STARTBR, RESETBR, READNEXT, READPREV and ENDBR**
  - These commands are threadsafe when accessing Local VSAM and VSAM RLS files
    - i.e. can be executed on an open TCB or QR TCB
  - These commands are not threadsafe when accessing Coupling Facility Data tables, shared data tables, remote files or BDAM files
    - i.e. must be executed on QR TCB and CICS will force a TCB switch if necessary
- **File Control SPI Commands**
  - INQUIRE FILE is now threadsafe
  - SET FILE, INQUIRE DSNAME, SET DSNAME, DISCARD FILE, CREATE FILE are still non threadsafe

Notes

- In CICS TS 3.2 the File control API commands have been made threadsafe for a subset of the File Control function, dependent on the access method involved.
- All File control API commands that are issued against local VSAM files or VSAM RLS files are threadsafe, and so will execute on an open TCB without incurring a TCB switch back to QR TCB. This can give significant throughput improvements.
- All File control API commands that are issued against remote files, coupling facility data tables, shared data tables, and BDAM files are non threadsafe, and so a TCB switch back to QR TCB will be incurred if the application is running on an open TCB at the time it issues the request.
- All SPI commands against all types of file are still non threadsafe, except for INQUIRE FILE which has been made threadsafe.

File Control – Global User Exits

- **It is important to make GLUEs that run at exit points within File Control threadsafe and define them as threadsafe.**
  - Non threadsafe GLUEs will cause excessive TCB switching for threadsafe and openapi applications
  - Exit points are: XFCREQ, XFCREQC, XFCSREQ, XFCSREQC, XFCLGERR, XFCBOVER, XFCBOUT, XFCLDEL, XFCBFAIL, XFCFRIN, XFCFROUT, XFCSREQ, XFCSREQC, XFCQUIS, XFCVSDS
  - Also XDTLC, XDTRD, XDTAD

Notes

- It is strongly recommended that all Global user exits are made threadsafe and defined to CICS as threadsafe. As more and more of the CICS API is made threadsafe, then this encompasses more and more exit points in CICS. A non threadsafe exit program will have a detrimental impact on performance.
- File Control has a great many exit points, and so if not done so already, any programs running at these exit points should be made threadsafe. This may involve contacting ISVs for maintenance, if a purchased package uses File Control exit points. Exit points XDTLC, XDTRD and XDTAD are used by data tables.

OTE enhancements in CICS TS 3.2 : CICS-MQ Interfaces

- **The following components have been transferred from the WebSphere MQ product into CICS TS 3.2:**
  - CICS-MQ Attach
  - MQ trigger monitor for CICS
  - MQ bridge (includes the DPL bridge and link 3270 bridge)
- **The CICS shipped MQ Attach will operate with all supported releases of WMQ**
- **The CICS shipped trigger monitor will operate with all supported releases of WMQ**
- **The CICS shipped MQ bridge will operate with WMQ V6 and higher**
  - When connected to WMQ v5.3.1, the CICS shipped bridge will transfer control to the WMQ shipped bridge
- **CICS shipped components will be serviced by CICS Level 2 and Level 3**
- **Websphere MQ will continue to ship the above components for use with CICS TS 3.1 and below**
  - Until such time as all releases of CICS TS prior to CICS TS 3.2 are out of service
  - Limited enhancements over time
  - Will functionally stabilize
  - MQ Level 2 and Level 3 will continue to service MQ shipped components

Notes

- CICS TS 3.2 has taken over ownership of the CICS-MQ Adapter, trigger monitor and bridge, in the same way as was done for the CICS-DB2 adapter a few releases ago.
- The CICS shipped components must be used with CICS TS 3.2 and higher. The WMQ shipped components must be used for CICS TS 3.1 and below. In addition the WMQ shipped bridge will be used when using WMQ V531.

CICS-MQ Interfaces - functionality

- **The functionality of the CICS shipped components is that of those shipped with Websphere MQ for z/OS V6.0 *plus*:**
  - CICS-MQ Attach is enhanced to use OTE
    - The CICS-MQ TRUE is enabled as OPENAPI
      - The CICS-MQ TRUE uses L8 TCBs, not its own private TCBs
    - MQ API commands from CICS applications are threadsafe
- **Existing applications will run unchanged, without recompile, and without re-linkedit**
- **The WebSphere MQ shipped Attach will NOT support OTE**

Notes

- The CICS shipped CICS-MQ adapter has been enhanced to exploit OTE in the same way as the CICS-DB2 adapter. In addition the CICS-MQ trigger monitor and bridge have been made threadsafe.
- The MQ API is now threadsafe and will not incur switching back to QR TCB and then to a private MQ TCB when processing an MQ API request. The CICS-MQ adapter uses L8 open TCBs.
- A threadsafe application running on an L8 open TCB will not incur any TCB switches when it issues an MQ API request.
- Similar to CICS-DB2 applications, a CICS-MQ application is not a good candidate to make an OPENAPI application when running with storage protection active, since the application will run on an L9 open TCB, whereas MQ requires an L8 open TCB. Such applications should be defined as THREADSAFE, CICSAPI rather than THREADSAFE, OPENAPI, which means they will switch to an open TCB when the first MQ request is issued and the application and MQ can share an L8 TCB even with storage protection active. This is because the CICSAPI does not take any notice of the key of the TCB. For an OPENAPI application, it may issue MVS requests, and MVS requires a TCB of the correct key (8 or 9).

Further information

- **CICS TS 2.2 / 2.3 / 3.1 / 3.2 Information Centers**
  - Refreshed regularly, can be downloaded from the IBM Publications Center:
  - http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi

- **Redbook: Threadsafe considerations for CICS SG24-6351**
  - http://www.ibm.com/redbooks
  - Second edition (July 2006) incorporates CICS TS 3.1 enhancements
  - Third edition planned

Questions