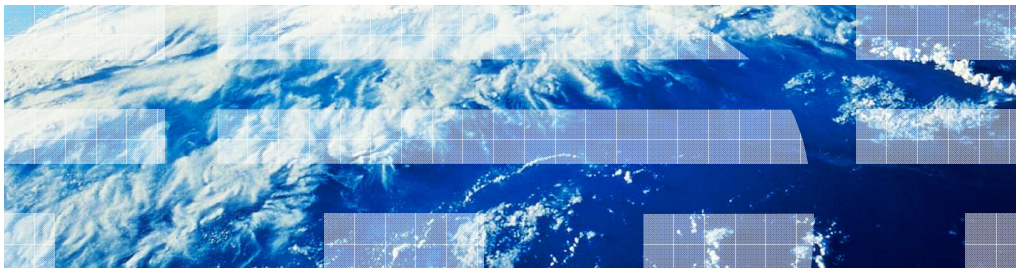
CICS Transaction Server V4.2

Threadsafe enhancements for multi-processor exploitation



This presentation introduces the theme of scalability and the thread safety enhancements introduced in CICS V4.2.

OTE settings in CICS TS 4.1 and below

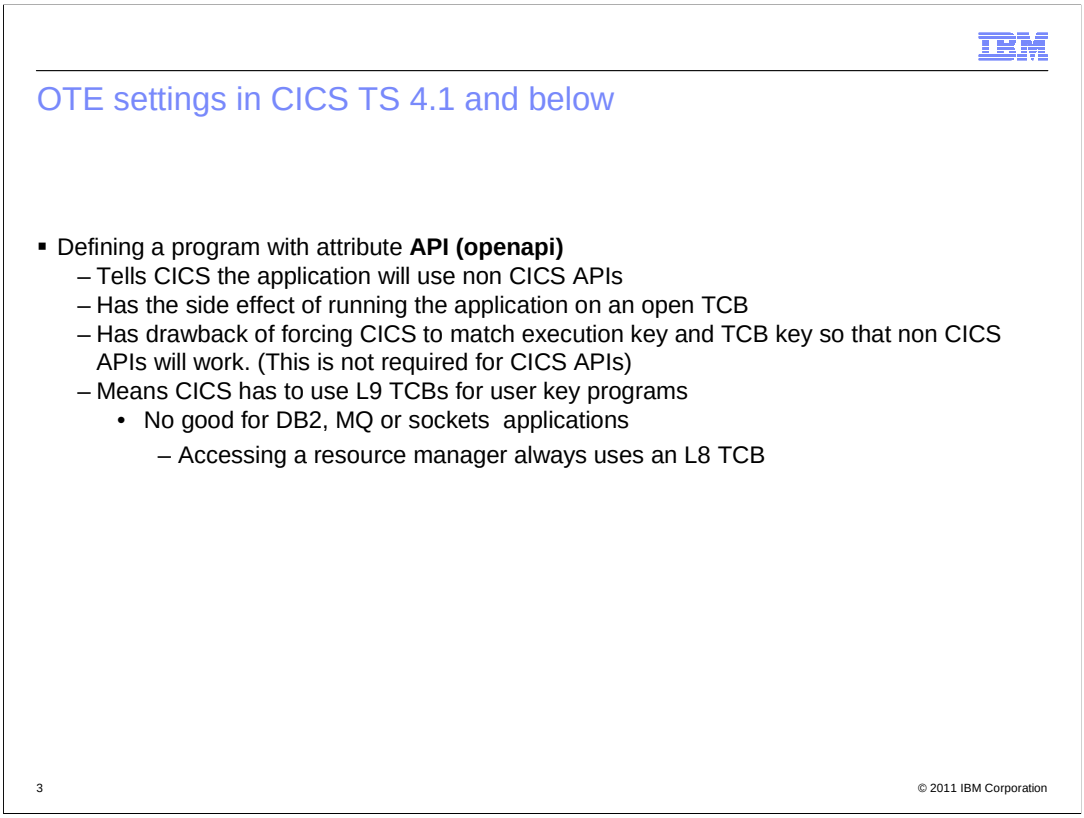
- Up till now you could define the program as
  - **Concurrency (quaisirent)** meaning it must run on QR TCB
  - **Concurrency (threadsafe)** meaning it can run on an open TCB or QR, depending on whichever is in use at the time
- There is no concurrency option to tell CICS that the application must run on an open TCB from the very start
- Instead, the only way you can do this was to define it with attribute **API (openapi)**
  - This has disadvantages

2

A concurrency (QUASIRENT) program always runs on the QR TCB.

A concurrency (THREADSAFE) program, is a program that has been coded to threadsafe standards, containing threadsafe logic. It is capable of running on either the QR TCB or an open TCB. It starts off running on QR TCB, if processing such as a DB2 request causes a switch to an open TCB, then on return to the program, it continues on the open TCB.

Before CICS TS 4.2, there is no concurrency option that specifies that the program should run on an open TCB from the start. Instead this can be achieved by defining it as API (OPENAPI) but this has disadvantages which are explained on the next slide.



Defining a program with attribute API(OPENAPI) has the side effect of causing the program to run from the start on an open TCB.

However, primarily, it is telling CICS that the program will issue non CICS supported API commands, that are something other than EXEC CICS, EXEC SQL, MQ commands for example, a MVS command.

Unlike CICS commands, a MVS command will only work correctly when the key of the TCB, matches the execution key. So when running EXECKEY(USER) which is key nine, an OPENAPI program is given a key nine TCB called an L9 TCB.

Contrast this with a program that only issues CICS supported commands and CICS does not reference the key of the TCB. This means a program can be running in userkey or CICS key and can run under a key eight TCB such as QR or an L8 TCB.

Users who define programs as OPENAPI get the advantage of starting on an open TCB, but suffer the disadvantage that an L9 is used (assuming storage protection is active).

This is a disadvantage because DB2 requires an L8 TCB, so you switch from L9 to L8 and back again for every DB2 request. This is why OPENAPI should not be used for DB2 programs, or MQ programs.

OTE settings in CICS TS 4.2

- CICS TS 4.2 separates whether an application must run on an open TCB, from what type of APIs it uses
- CONCURRENCY(REQUIRED)
  - Application must be coded to threadsafe standards
  - States that the application must run on an open TCB
  - Application starts on an open TCB
  - If a switch to QR is made for a CICS command, it switches back to the open TCB when returning to the application
- Existing API keyword defines what APIS are used
  - This defines what type of TCB is used

CICS TS 4.2 provides a new CONCURRENCY(REQUIRED) setting which specifies that the program requires to run on an open TCB.

It will run an open TCB from the start, and if CICS has to switch to QR TCB to process a non-threadsafe CICS command, it will return to the open TCB when it returns to the application program.

Now, you can define that the program must start on an open TCB, independently of defining what APIs it uses.

The API parameter determines what type of open TCB is used.

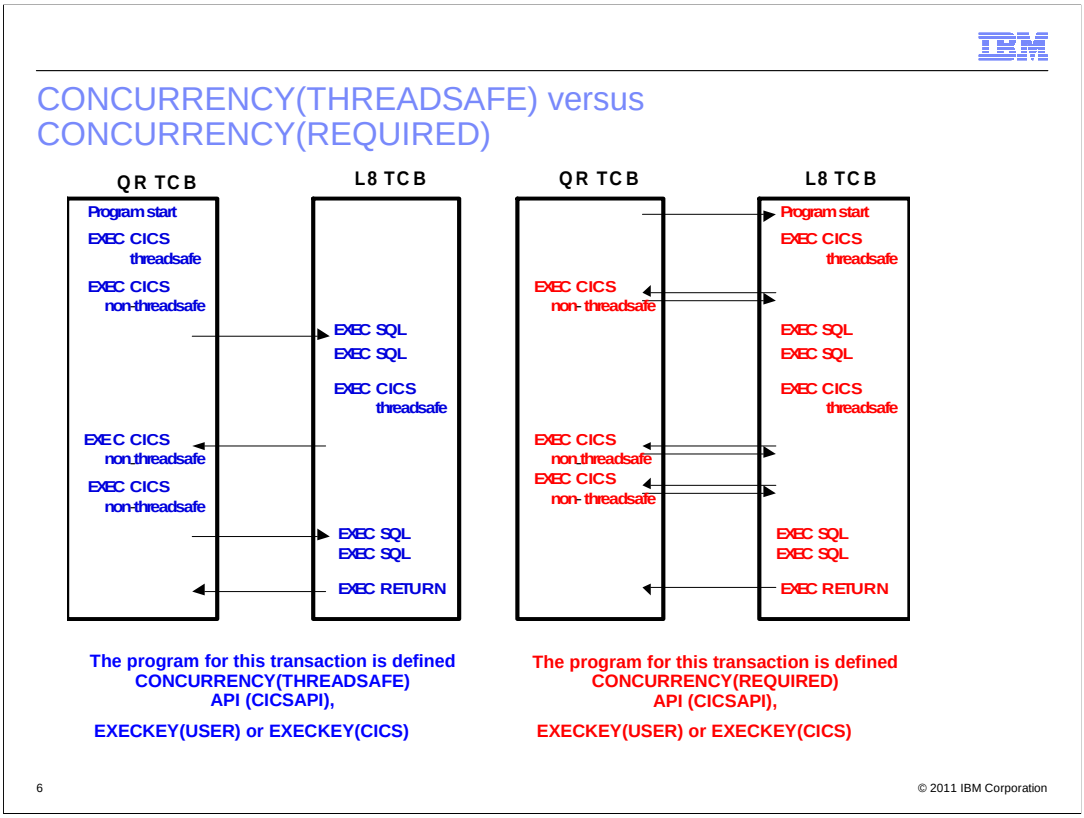
OTE settings in CICS TS 4.2

- CONCURRENCY(REQUIRED) and API(CICSAPI)
  - The application will run on an open TCB from the start
  - It only uses CICS supported APIs (including DB2, IMS and MQ)
    - CICS will always use an L8 TCB in this instance irrespective of the execution key
  - This is great for applications that are going to resource managers like DB2 and MQ as the same L8 is used
- CONCURRENCY(REQUIRED) and API(OPENAPI)
  - The application will run on an open TCB from the start
  - As it will use non CICS APIs, it will run on an L8 or an L9 TCB depending on the execution key. This is the same as CICS TS 4.1 today
  - Only use OPENAPI when non CICS supported APIs are to be used

5

For CONCURRENCY (REQUIRED) programs, the type of open TCB used depends on what APIs the program is to use. If the program uses only CICS supported APIs (including access to external resource managers such as DB2 IMS and WebSphere MQ) then it should be defined with program attribute API(CICSAPI). In this case CICS always uses an L8 open TCB, irrespective of the execution key of the program, because CICS commands do not rely on the key of the TCB.

If the program is to use other non-CICS APIs then it must be defined with program attribute API(OPENAPI). In this case CICS uses an L9 TCB or an L8 TCB depending on the execution key of the program. This is to allow the non-CICS APIs to operate correctly. This OPENAPI behavior is the same as previous releases.



Existing threadsafe CICS-DB2 applications which have taken advantage of the performance gains of being able to run on the same TCB as the DB2 call by being defined as THREADSAFE. CICSAPI can be further enhanced by defining them as REQUIRED CICSAPI. This definition means that the programs can run on an L8 open TCB, irrespective of their execution key. CICS for example, can run the program in key nine on an key eight TCB from the start without waiting for the first DB2 call to move them on to the open TCB. Achieving additional benefit depends on how many, if any, non-threadsafe CICS commands the application executes.

Threadsafe mirror

- DFHMIRS is now threadsafe
  - Supplied definition now specifies CONCURRENCY (THREADSAFE)
- IPIC transformers are now threadsafe. Non IPIC code remain non threadsafe
- Only requests function shipped over IPIC will run on an Open TCB
  - File control and temporary storage
  - Distributed program link
    - If the target program is defined as threadsafe and the mirror already on an open TCB
- Review your DFHSIT specification for FCQRONLY if using IPIC
  - Specify FCQRONLY=NO as there is no longer any need to turn off thread safety in the FOR

The CICS-supplied mirror program DFHMIRS, which is used by all mirror transactions, is now defined as threadsafe. In addition the IPIC transformers have been made threadsafe.

For IPIC connections only, CICS runs the mirror program on an L8 open TCB whenever possible.

For threadsafe applications that function ships commands to other CICS regions using IPIC, the resulting reduction in TCB switching, improves the performance of the application compared with other intercommunication methods.

To gain performance improvement, you must specify the system initialization parameter FCQRONLY=NO in the file-owning region.

File control requests that are function shipped using IPIC connectivity provide threadsafe file control with significant potential throughput improvements over LU6.2 in CICS regions with multiple processors available.

Temporary storage requests that are function shipped using IPIC connectivity are threadsafe and no longer need to switch to QR before being function shipped.

Threadsafe function shipping over IPIC

- Remote file control and TS requests over IPIC are now threadsafe
  - No switch to QR in the AOR
  - Transformers can run on open TCB in the AOR and ship the request
- File control and TS requests in the remote region can run on the open TCB
  - Mirror switches to open TCB for 1st FC or TS request
  - TS request will run on open TCB
  - FC request will run on open TCB provided
    - FCQRONLY=NO is set
    - Its local VSAM LSR file or RLS (for example, existing FC threadsafe support)
- MIRRORLIFE on the IPCONN definition controls lifetime of mirror
  - UOW or TASK allows mirror to continue

8

For remote file control or TS requests shipped over IPIC connections, CICS will no longer force a switch to QR TCB if it is running currently on an open TCB. The requests are shipped running on the open TCB.

In the FOR or QOR, the mirror decides when to switch to an open TCB. It does so for the first file control or TS request received over an IPIC connection. The idea is for long running mirrors to keep the mirror running on an open TCB.

A new option MIRRORLIFE has been added to the IPCONN attributes for function-shipped file control, transient data, and temporary storage requests using an IPIC connection. MIRRORLIFE improves efficiency and provides performance benefits by specifying the lifetime of mirror tasks and the amount of time a session is held.

Threadsafe CICS-DBCTL (1 of 4)

- CICS-DBCTL interface will use OTE when connected to IMS 12
  - At connect time CICS and IMS determine if each other can support OTE
  - With IMS 10 and 11
    - CICS-DBCTL TRUE enabled as QUASIRENT
    - Toleration APAR PM31730 (IMS 10), PM31729 (IMS 11)
  - With IMS 12
    - CICS-DBCTL TRUE enabled as OPENAPI
    - Exploitation APAR PM31420 is required
- General Availability of CICS TS V4.2, IMS 12 is available through a Quality Partnership Program (QPP)
  - For more information, visit http://www.ibm.com/software/data/ims/

9

CICS provides a CICS IMS database control (CICS-DBCTL) interface for IMS to satisfy DL/I requests that are issued by applications running in a CICS region. In CICS TS 4.2 the CICS-DBCTL interface has been defined as threadsafe and CICS can run the CICS-DBCTL task-related user exit (TRUE) on an L8 open task control block (open TCB).

The open transaction environment (OTE) is supported from IMS version 12 with PTFs for APAR PM31420 applied. IMS indicates to CICS during the connection process that the OTE is supported and consequently CICS defines the CICS-DBCTL TRUE as an open API TRUE.

An open API TRUE is run on an L8 open TCB, which is dedicated for use by the calling CICS task. Running an application on an open TCB improves throughput and performance by reducing the use of the QR TCB. Threadsafe CICS applications that run on an L8 open TCB and use threadsafe CICS-DBCTL commands now avoid up to four TCB switches for each call to IMS. For more information about CICS IMS applications and the OTE, see Enabling CICS IMS applications to use the open transaction environment (OTE) through threadsafe programming.
If your IMS version does not support the OTE, CICS runs the CICS-DBCTL TRUE on the QR TCB.

Threadsafe CICS-DBCTL (2 of 4)

- TCB switching is controlled by the IMS database resource adapter (DRA)
  - IMS 12 DRA will not switch TCBs and run the IMS request on the calling l8 TCB
  - < IMS 12 DRA will continue to be called on QR TCB and will switch to a DRA thread TCB
- CICS code implementing CICS-DBCTL interface made threadsafe
- CALLDLI and EXEC DLI API now threadsafe when run with IMS 12

The IMS DRA is the code that, before CICS TS 4.2 will switch from QR TCB to an IMS thread TCB and back again for each IMS request.

With CICS TS 4.2 and IMS V12 the DRA will not switch TCBs, but will run the request on the calling TCB which is an L8 TCB. Hence the CALL DLI and EXEC DLI commands are threadsafe when running with IMS V12 and will run on the L8 TCB.

The CICS side of the CICS-DBCTL interface has been made threadsafe so that the processing can run on an L8 TCB.

Threadsafe CICS-DBCTL (3 of 4)

- For non threadsafe CICS applications there is no change to the amount of TCB switching
  - Switch to L8 to invoke DFHDBAT and IMS
  - Switch back to QR when returning to the application
- For threadsafe CICS applications save TCB switches
  - Remain on L8 after return from IMS request
  - Expect same benefits as seen for threadsafe CICS-DB2 applications
    - Reduced processor usage
    - More throughput

For a non threadsafe application there is no reduction in the amount of switching. Instead of switching from QR TCB to an IMS thread TCB and back again for each IMS request, it switches from QR to L8 and back again.

For a threadsafe application, if it is running on QR TCB, it will switch to L8 and then stay on L8 when returning to the application.

For a threadsafe application that is already running on an L8 TCB, or for a CONCURRENCY(REQUIRED) application running on an L8 TCB, then no TCB switching occurs for the IMS request.

Threadsafe CICS-DBCTL (4 of 4)

- CICS-IMS applications when using IMS 12 will use L8 TCBs
  - Adjust MAXOPENTCBs as necessary
- DRA startup table DFSPZPxx still controls how many IMS threads can be used
  - MINTHRDS - the number of threads that remain 'signed on'
    - DRA keeps IMS threads but dissociated from the TCB
  - MAXTHRDs still limits the number of IMS threads from this CICS region
    - Even though an L8 TCB might be available, IMS limits how many can be used for threads
      - If exceeded, MAXTHRDS will queue in the IMS DRA as it does today (pre OTE)

DRA startup table DFSPZPxx still controls how many IMS threads can be used.

Firstly, MINTHRD=xxx

This parameter specifies the number of threads for this CICS system that once initialized, remains created while the DRA is active. These threads remain allocated until this CICS system is disconnected from DBCTL, except if a thread is stopped by a STOP command or by a thread failure. Additional threads are created, up to the number specified in MAXTHRD, or the number specified in MAXREGN, or the maximum of 999, whichever of these values is the lowest. These additional threads (not the MINTHRDs) are released when there is not enough system activity to require them. The maximum value you can specify for MINTHRD is 999, and the default is one. For information about specifying values for MINTHRD, see specifying numbers of threads. See also MAXREGN in IMS system generation macros used by DBCTL.

Secondly, MAXTHRD=xxx

This parameter specifies the maximum number of transactions for which this CICS system can have PSBs scheduled in DBCTL. Any schedule requests that are over this limit are queued in the DRA. You can balance the load sent to a single DBCTL from multiple CICS systems by specifying the appropriate values for MAXTHRD in each CICS.
The maximum value you can specify for MAXTHRD is 999 (but it should not exceed the value specified for MAXREGN) and the default is one, or the value you specified in MINTHRD. For information about specifying values for MAXTHRD, see specifying numbers of threads. See also MAXREGN in IMS system generation macros used by DBCTL.

Threadsafe sync point

- Commands now made threadsafe*
  - EXEC CICS SYNCPOINT
  - EXEC CICS SYNCPOINT ROLLBACK
  - EXEC CICS RESYNC

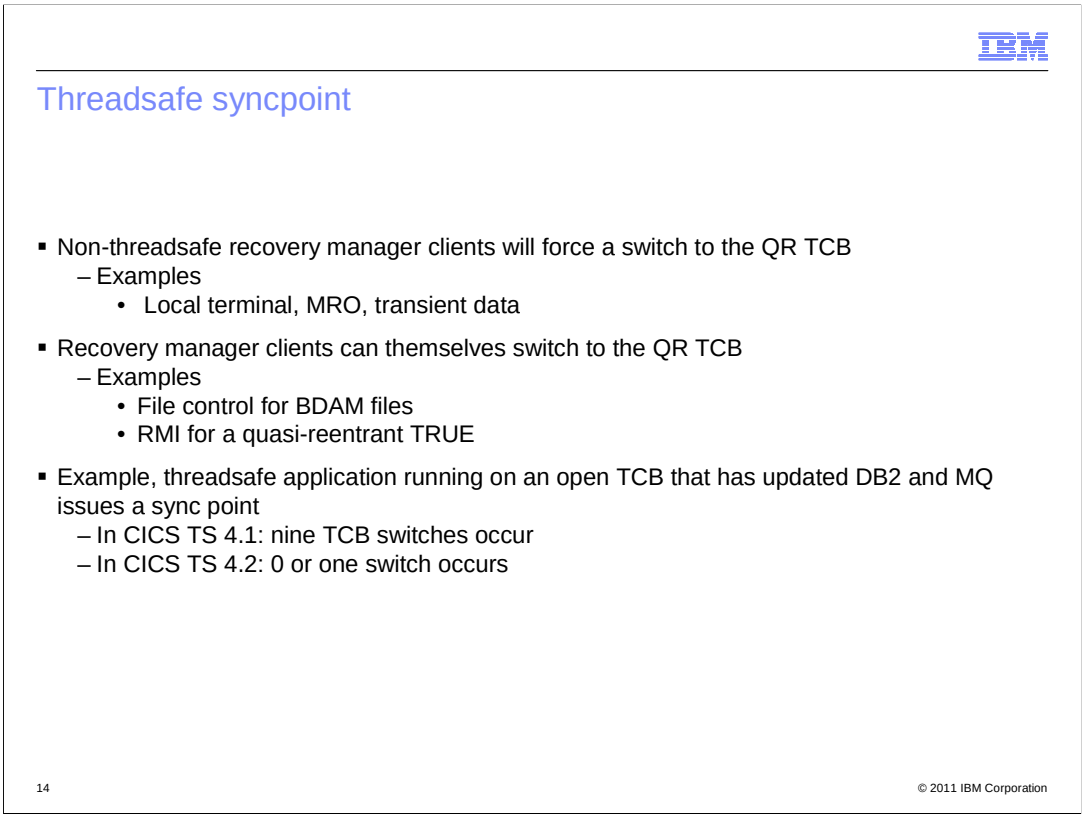
    * some switching can still occur, but is heavily reduced
- EXEC API no longer switches to QR for sync point requests
  - Recovery Manager (RM) domain now makes the decision if a switch is required
  - All RM domain clients register at startup and tell RM if they are threadsafe
  - RM domain keeps it on the open TCB wherever possible
- End of task sync point can run on an open TCB before CICS switches to QR to terminate the task

13

The recovery manager processes this command on an open TCB wherever possible to minimize TCB switching.

Sync point processing can take place on an open TCB for all resource types declared as threadsafe that were accessed in the unit of work. If resource types not declared as threadsafe were accessed in the unit of work, the recovery manager switches to the QR TCB for those resource types. A CICS resource type declares itself to the recovery manager as threadsafe if the EXEC CICS commands relating to the resource type are threadsafe.

Before CICS TS 4.2, CICS used to switch to QR before the end of a task sync point. In CICS TS 4.2 it remains on an open TCB, (if it is running on one) until end of task sync point has been called. Afterwards it switches to QR for the task detach logic.



Terminal driven transactions involve CICS terminal control logic which is not threadsafe, so a switch to QR will occur during phase two sync point processing.

This is an example of recovery manager clients that will cause a switch to QR.

Other recovery manager clients will themselves switch to QR if necessary. For example, file control is threadsafe when processing local VSAM files, but it will switch to QR TCB for BDAM files.

Before CICS TS 4.2, a threadsafe application running on an L8 TCB that had updated DB2 and MQ and then issued a sync point would suffer nine TCB switches.

A switch to QR would be made at the start of sync point.

Switches to L8 and back to QR would occur when calling DB2 for PREPARE

Switches to L8 and back to QR would occur when calling MQ for PREPARE

Switches to L8 and back to QR would occur when calling DB2 for COMMIT

Switches to L8 and back to QR would occur when calling MQ for COMMIT

In CICS TS 4.2, if it was a terminal driven transaction, one TCB switch to QR would occur. For a non terminal driven transaction (and assuming no other non threadsafe resources had been touched) then no TCB switches would occur.

Other APIs made threadsafe

- QUERY SECURITY
- SIGNON, SIGNOFF
- VERIFY PASSWORD, VERIFY PHRASE
- CHANGE PASSWORD, CHANGE PHRASE
- EXTRACT TCPIP, EXTRACT CERTIFICATE
- All Call and Exec Level Named Counter Server commands
- Built in functions for DIGEST and DEEDIT

The listed API commands have been made threadsafe in CICS TS 4.2 and can run on an open TCB.

Threadsafe SPI commands

- New SPI commands that are threadsafe:
  - INQUIRE CAPDATAPRED, INQUIRE CAPINFOSRCE, INQUIRE CAPOPTPRED
  - INQUIRE EPADAPTER, SET EPADAPTER
  - INQUIRE OSGIBUNDLE, INQUIRE OSGISERVICE
  - INQUIRE TEMPSTORAGE, SET TEMPSTORAGE
- Existing SPI commands made threadsafe:
  - INQUIRE CLASSCACHE
  - INQUIRE JVM
  - INQUIRE JVMPOOL
  - INQUIRE JVMPROFILE
  - PERFORM CLASSCACHE
  - PERFORM JVM POOL
  - SET CLASSCACHE
  - SET JVMPOOL

The listed SPI commands have been made threadsafe in CICS TS 4.2 and can run on an open TCB.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_CICSTS42_Scalability_threadsafety.ppt

This module is also available in PDF format at: ../CICSTS42_Scalability_threadsafety.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information