

# CICS TS DB2 Interface for All



Thomas Dunlap Themis, Inc. [tmd@themisinc.com](mailto:tmd@themisinc.com)

Please use the 'Question Tab' for questions instead of the Chat window.

Link to get the PDF : [www.themisinc.com/webinars](http://www.themisinc.com/webinars)



# **A LOOK AT THE APPLICATION PROGRAM**

## DB2 SQL Commands in COBOL Program

WORKING-STORAGE SECTION.

```
EXEC SQL INCLUDE SQLCA END-EXEC.  
EXEC SQL INCLUDE DCLEMP END-EXEC.  
EXEC SQL INCLUDE DCLDEPT END-EXEC.
```

### One of included items:

```
EXEC SQL DECLARE THEMIS82.DEPT TABLE  
( DEPTNO          CHAR(3) NOT NULL,  
  DEPTNAME        VARCHAR(36) NOT NULL,  
  MGRNO           CHAR(6),  
  ADMRDEPT        CHAR(3) NOT NULL,  
  LOCATION        CHAR(16)  
) END-EXEC.
```

\*\*\*\*\*

\* COBOL DECLARATION FOR TABLE THEMIS82.DEPT \*

\*\*\*\*\*

```
01 DCLDEPT.  
 10 WS-DCLDEPT-DEPTNO  PIC X(3).  
 10 WS-DCLDEPT-DEPTNAME.  
    49 WS-DCLDEPT-DEPTNAME-LEN  
      PIC S9(4) USAGE COMP.  
    49 WS-DCLDEPT-DEPTNAME-TEXT  
      PIC X(36).  
 10 WS-DCLDEPT-MGRNO   PIC X(6).  
 10 WS-DCLDEPT-ADMRDEPT PIC X(3).  
 10 WS-DCLDEPT-LOCATION PIC X(16).
```

Includes necessary for program execution

Each application program which will execute using SQL statements require a few basic inclusions of generated code. The DECLARE statements have been generated using the DCLGEN option of the DB2 Interactive panels. The SQLCA is distributed as part of the DB2 product libraries. The DCLEMP and DCLDEPT includes represent the DB2 tables being used by the application. The prime reason for using the DB2 INCLUDE statements is due to the SQL preprocessor being invoked before the COBOL compile. If you utilize the integrated DB2 processor as we do, then you could use a standard COPY statement in place of the INCLUDE.


We have included on this slide one of the included DECLARE statements. This SQL statement is required to provide the table name and column names for use by the application. This statement also provides the DB2 data type definitions for the columns of the table. Also as part of the DCLGEN output is a COBOL data structure representing the columns of the table and their data types. This data area can be used on the SQL statements to supply the "host variables" to accept the data being processed. One special style of definitions is related to DB2 VARCHAR data type, where you will notice the COBOL structure contains a half-word binary length field along with the character data attribute. This length will contain the actual number of significant characters within the character attribute.

**DB2 SQL Commands in COBOL Program**

```
WORKING-STORAGE SECTION.  
  
EXEC SQL  
  DECLARE EMP_CURSOR CURSOR FOR  
  SELECT EMPNO, LASTNAME, FIRSTNME, SALARY, BONUS, COMM  
  FROM   THEMIS82.EMP  
  WHERE  DEPTNO = :OL-DEPTNO  
  ORDER BY SALARY DESC  
END-EXEC
```

---

Application includes a SQL CURSOR to create the results set for processing

 4

Our application does utilize an SQL CURSOR to process a multiple row results set. We have included the DECLARE for the cursor at the end of our WORKING-STORAGE SECTION. This is due to the fact that there is no executable code produced by this statement. It is our recommendation that all DECLARE statements for cursors be included at the end of the WORKING-STORAGE SECTION.

Ours is a rather simple SQL statement which will take in a department number entered by the user (OL-DEPTNO) and produce a results set of the employees working in that department. The information returned is the key money attributes. You will notice the ORDER BY clause, which will cause DB2 to sort the results set by the specified column. Depending upon the size of the results set, this can add significant time to complete the query.

Also, depending upon the complexity of the SQL statement and the number of tables involved, the processing of the cursor at SQL OPEN can take a significant amount of time. Performance of DB2 SQL statements is mostly based upon the manor in which they are coded.

**DB2 SQL Commands in COBOL Program**

```

PROCEDURE DIVISION.

EXEC SQL
  SELECT  AVG(SALARY), MAX(SALARY)
         INTO :AVG-SALARY, :MAX-SALARY
  FROM    THEMIS82.EMP
  WHERE   DEPTNO = :OL-DEPTNO
END-EXEC


EXEC SQL OPEN EMP_CURSOR
END-EXEC

EXEC SQL FETCH EMP_CURSOR
  INTO   :WS-EMP-EMPNO, :WS-EMP-LASTNAME,
         :WS-EMP-FIRSTNAME,
         :WS-EMP-SALARY INDICATOR :WS-SALARY-NI,
         :WS-EMP-BONUS INDICATOR :WS-BONUS-NI,
         :WS-EMP-COMM INDICATOR :WS-COMM-NI
END-EXEC

EXEC SQL CLOSE EMP_CURSOR
END-EXEC

SQL statements executed within the PROCEDURE DIVISION

```

 5

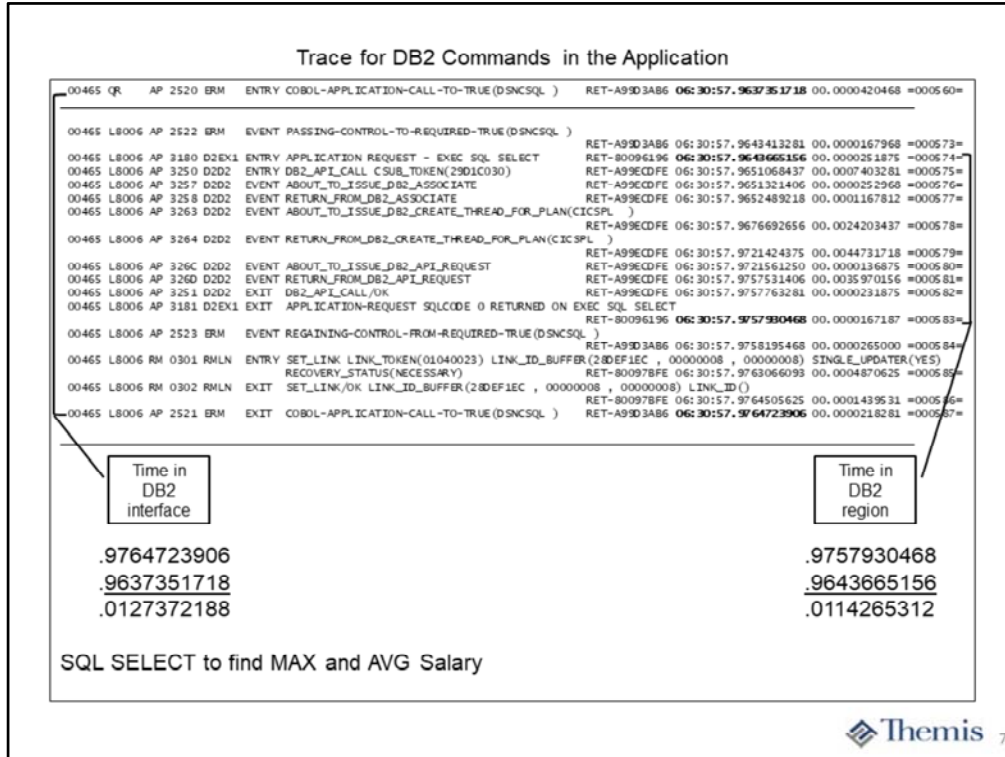
Now to the actual executable SQL statements on this slide. The first SQL statement is a simple SELECT to find the average and maximum SALARY for the department number entered by the user. While this may seem like a simple request, its performance is highly dependent upon how many people work in the requested department, plus the availability of an index on DEPTNO.

The second statement, SQL OPEN, can also be of highly variable performance. It is dependent upon the complexity of the DECLARE CURSOR statement coded in WORKING-STORAGE. Once again, when clauses like ORDER BY are included, the results set must be sorted. However, it is at SQL OPEN when the actual SELECT within the DECLARE CURSOR statement is executed.

Next is the SQL FETCH statement to return one row at a time from the results set. The statement itself is quite simple, but it is performed in a loop until the last row of the results set has been retrieved. SO its performance is highly dependent upon the number of rows in the results set.

The last statement, SQL CLOSE, is performed when the end of the results set has been reached. While the statement itself is quite simple, it will cause DB2 to cleanup the results set for that cursor upon execution.

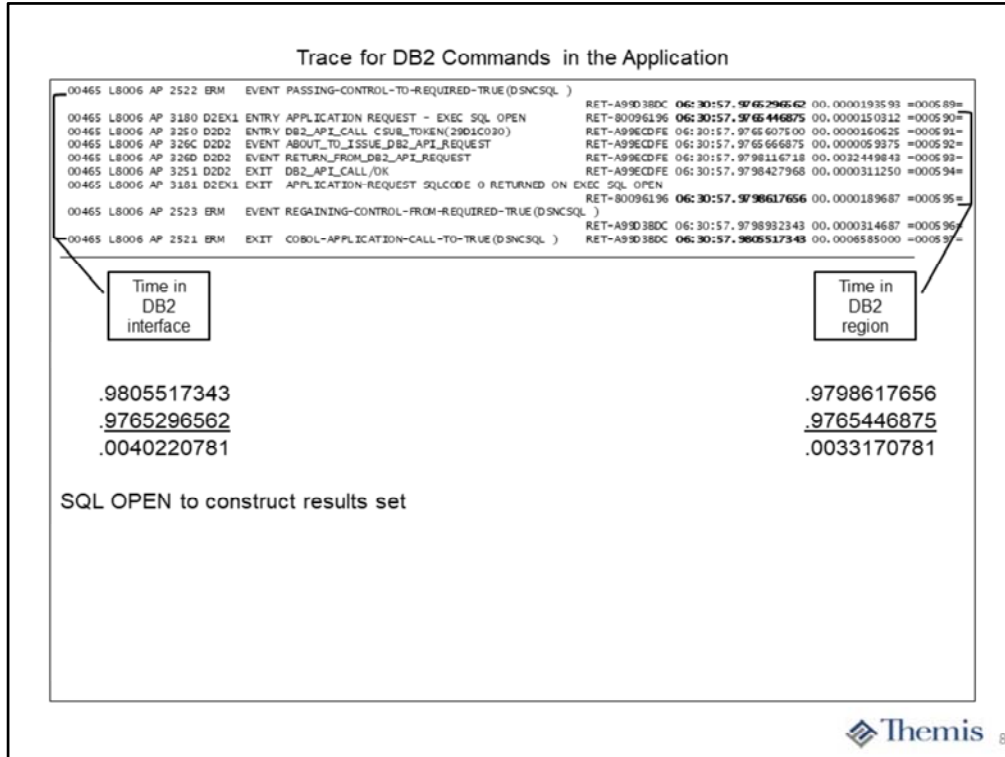
# **A LOOK AT TRACE FOR THE APPLICATION PROGRAM**



This slide contains the portion of trace entries representing the SQL SELECT statement to find the average and maximum SALARY. The bracketed entries on the left represent the total time spent in the CICS DB2 interface. Most of this time (processor time) will actually be captured and reported on the DB2 region side. This is because control is transferred from the DB2 interface to DB2 as a secondary address space very quickly. We have always classified the time spent between “ERM ENTRY” and “ERM EXIT” as time when the application has turned control over to DB2 and entered a wait state.

The bracketed entries on the right represent the total time spent in the DB2 region itself. The “D2EX1 ENTRY” trace entry occurs just before giving control to DB2. The “D2EX1 EXIT” occurs just after DB2 give control back to the CICS DB2 interface. This time is considered to be solely done on behalf of the DB2 region and the CICS application is in a wait state for its completion.

While the times shown on the slide might seem to be relatively low, on our small little mainframe we have executed around 1.5 million instructions to complete this SELECT statement.

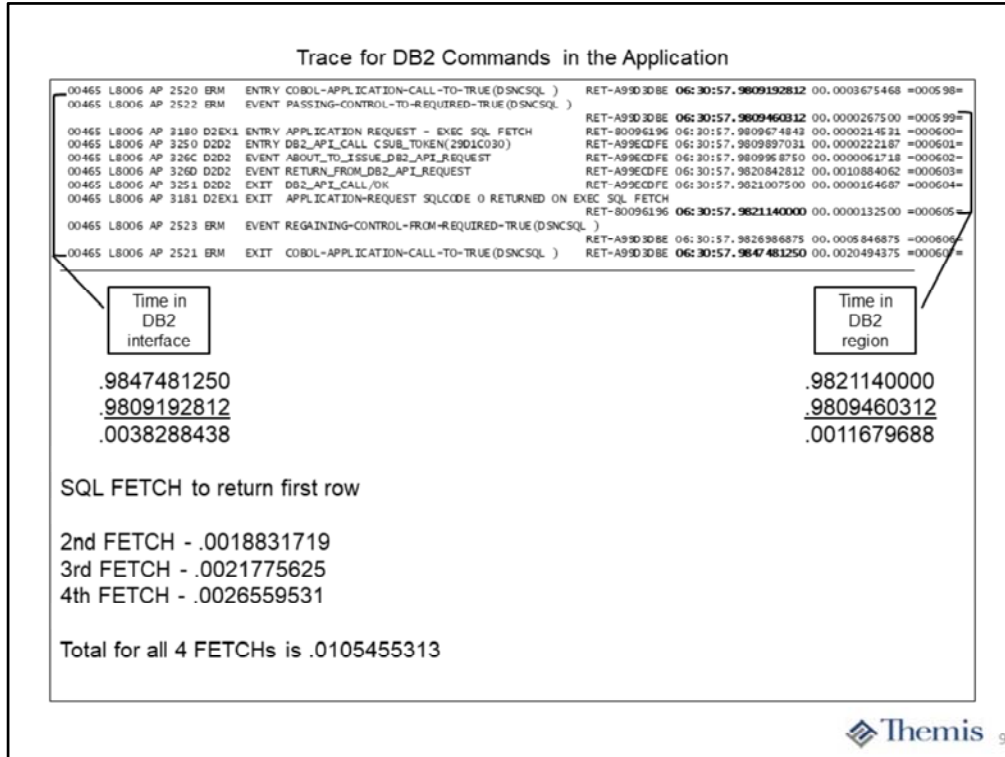


Once again, the bracketed trace entries on the left side represent the total time spent in the CICS DB2 interface. This is the execution of the SQL OPEN statement to perform the SELECT from the DECLARE CURSOR statement.

The bracketed entries on the right side represent the actual time spent in the DB2 region itself. The total time spent in DB2 is highly dependent upon the complexity of the SQL statement coded.

In our case the time is relatively low. This is primarily due to the simplicity of the SQL SELECT statement we have coded. Even so, we have executed around 400,00 instructions to complete the SQL statement.

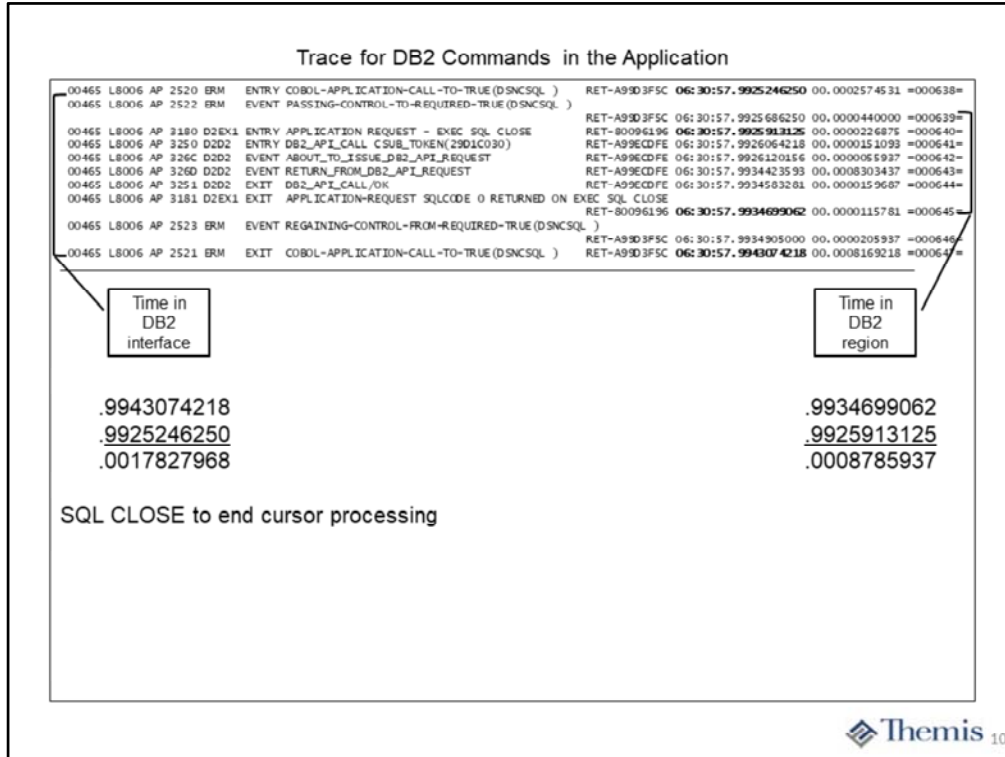




On this slide we show only the first SQL FETCH statement being executed. The bracketed entries on the left show the total time spent within the CICS DB2 interface. The bracketed entries on the right show the total time spent actually in the DB2 region. We noticed that the time in the interface is three times the amount actually spent in DB2 itself. It seems that the prime reason this happened is that we only have 3 CPUs and our CICS region was contending with many other tasks currently running on our system.

We have captured the times in the CICS DB2 interface for the other 3 FETCH statements executed to return all 4 rows from the results set. We do not have any concrete reason for the slight increase for each subsequent FETCH statement. We believe that it was due to other DB2 applications executing on our system at the time. We do not have a performance monitor like Omegamon implemented on our system so it is hard to determine the actual contention at any point in time.

Given our small little mainframe, we still required over 1 million instructions to process the 4 rows within the results set.



The last SQL statement executed by our application is the SQL CLOSE shown on this slide. Once again it seems that we spent twice as much time in the CICS DB2 interface as within the DB2 region itself. Since all of the activity within our application occurred within a very short period of time, we chalk this difference up to contention for our 3 CPUs with many other tasks occurring on our little mainframe.

The important facts to take away from all of these trace samples is the “**Time spent in the CICS DB2 interface**” illustrated by the bracketed entries on the left and the “**Time spent in the DB2 region**” illustrated by the bracketed entries on the right.

### Trace for DB2 Commands in the Application

00465	QR	AP	00E1	EZIP	ENTRY RETURN	REQ(0004) FIELD-A(28B0AAE0 ...)\ FIELD-B(09000E08 ...)
					BOUNDARY(0200)	RET-A98D4074 06:30:57.9964523750 00.0000124062 =000669=
00465	QR	AP	E160	EXEC	ENTRY RETURN COBOL II STMT_#(00056)	RET-800850A2 06:30:57.9964698906 00.0000175156 =000670=
<hr/>						
00465	QR	RM	FA11	RMUD	ENTRY COMMIT_UOW CONTINUE(NO)	RET-A635A0E8 06:30:57.9984583598 00.0000170927 =000696=
<hr/>						
00465	QR	RM	0361	RMLSD	ENTRY Decide CONTINUE(NO) READONLY(YES)	RET-A63F31EA 06:30:57.9986072187 00.0000135000 =000704=
00465	QR	RM	0368	RMLSD	EVENT About_to_call Client(RMI) Local_UOW_Id(D33EE9EDFE9F040)	RET-A63F31EA 06:30:57.9986140468 00.0000068281 =000705=
00465	QR	AP	2500	BRMSP	ENTRY SEND_DO_COMMIT RMC_TOKEN(280EF190) CONTINUE(DSNCSQL)	RET-A63CCE0E 06:30:57.9986198906 00.0000058437 =000706=
00465	QR	AP	2509	BRMSP	EVENT INVOKE_RMI TRUE(DSNCSQL) FOR ONLY UPDATER (UERTONLY) REQUEST	RET-A63CCE0E 06:30:57.9986265156 00.0000066250 =000707=
00465	QR	RM	520	ERM	ENTRY SYNCPOINT-MANAGER-CALL-TO-TRUE(DSNCSQL)	RET-27491CF8 06:30:57.9986501718 00.0000236562 =000708=
<hr/>						
00465	L8006	AP	2523	ERM	EVENT PASSING-CONTROL-TO-REQUIRED-TRUE(DSNCSQL)	RET-27491CF8 06:30:58.0000513906 00.0000197187 =000717=
00465	L8006	AP	318	D2EX1	ENTRY SYNCPOINT-MANAGER REQUEST	RET-80096196 06:30:58.0000774843 00.0000260937 =000718=
00465	L8006	AP	325	D2D2	ENTRY SINGLE_PHASE_COMMIT CSUB_TOKEN(28D1C030)	RET-A99ED002 06:30:58.0000996562 00.0000221718 =000719=
00465	L8006	AP	327	D2D2	EVENT ABOUT_TO_ISSUE_DB2_SINGLE_PHASE_COMMIT	RET-A99ED002 06:30:58.0001087031 00.0000090468 =000720=
00465	L8006	AP	327	D2D2	EVENT RETURN_FROM_DB2_SINGLE_PHASE_COMMIT	RET-A99ED002 06:30:58.0029120937 00.0028033906 =000721=
00465	L8006	AP	325	D2D2	EVENT ABOUT_TO_ISSUE_DB2_DISSOCIATE	RET-A99ED002 06:30:58.0029414843 00.0000293906 =000722=
00465	L8006	AP	325	D2D2	EVENT RETURN_FROM_DB2_DISSOCIATE	RET-A99ED002 06:30:58.0030393750 00.0000978906 =000723=
00465	L8006	AP	325	D2D2	EXIT SINGLE_PHASE_COMMIT/OK	RET-A99ED002 06:30:58.0035397968 00.000504218 =000724=
00465	L8006	AP	318	D2EX1	EXIT SYNCPOINT-MANAGER REQUEST	RET-80096196 06:30:58.0038278281 00.0002880312 =000725=
00465	L8006	AP	2523	ERM	EVENT REGAINING-CONTROL-FROM-REQUIRED-TRUE(DSNCSQL)	RET-27491CF8 06:30:58.0038580468 00.0000302187 =000726=
<hr/>						
00465	QR	AP	2521	ERM	EXIT SYNCPOINT-MANAGER-CALL-TO-TRUE(DSNCSQL)	RET-27491CF8 06:30:58.0046982656 00.0005428281 =000734=
00465	QR	AP	2510	BRMSP	EVENT RETURN FROM RMI TRUE(DSNCSQL) WITH OK (UERFOK) RESPONSE	RET-A63CCE0E 06:30:58.0047151875 00.0000219218 =000735=
00465	QR	AP	2501	BRMSP	EXIT SEND_DO_COMMIT/OK ACCESSIBLE(YES) VOTE(YES) TRUE(DSNCSQL)	RET-A63CCE0E 06:30:58.0047234521 00.0000082656 =000736=
00465	QR	RM	0362	RMLSD	EXIT Decide RESPONSE(OK) CONTINUE(NO) READONLY(NO) VOTE(YES)	RET-A63F31EA 06:30:58.0047366093 00.0000131562 =000737=
<hr/>						
00465	QR	RM	FA12	RMUD	EXIT COMMIT_UOW/OK FAILED_LINK(00000000)	RET-A635A0E8 06:30:58.0085086093 00.0004678750 =000762=

**EXEC CICS RETURN issued, includes commit with DB2**

11

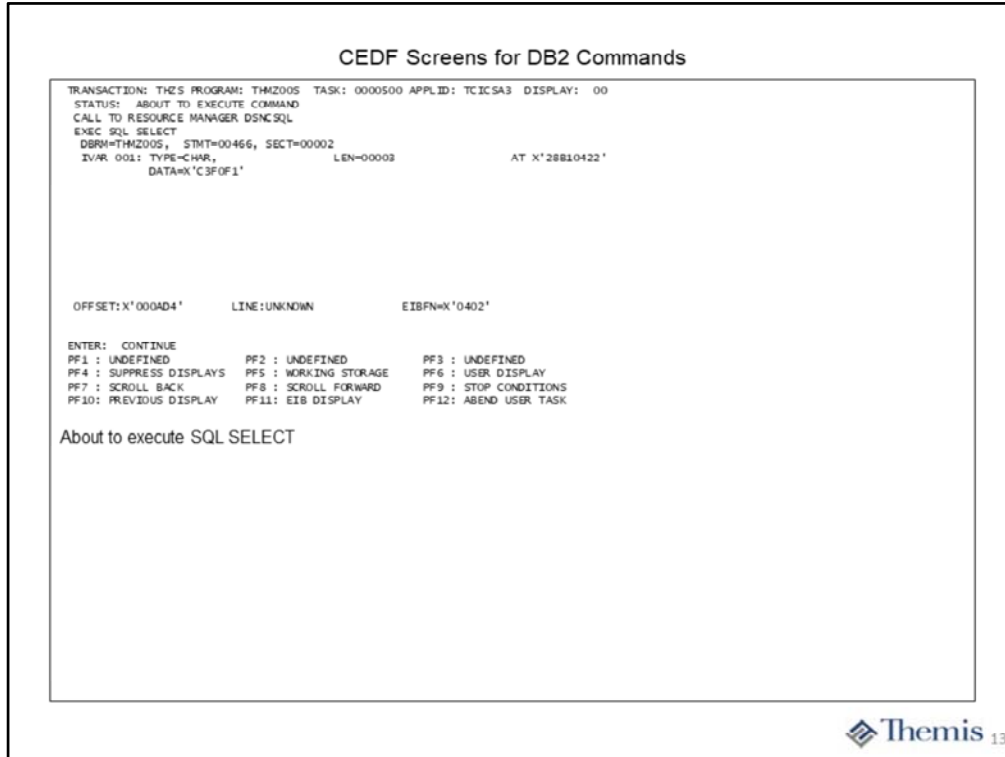
One last time through the CICS DB2 interface as part of the EXEC CICS RETURN command being executed. CICS is the coordinator for committing modified data during execution of the application within the CICS region. There is a lot of activity going on during this process of ending and cleanup of the application. We have carved out the activity associated with the CICS DB2 interface only.

The trace entries represented by connector **A**, represent the total time within the CICS DB2 interface itself. The entries represented by the connector **B**, show the time the interface communicated with Resource Recovery Services (RRS) address space on z/OS. The entries represented by connector **C**, is the actual time spent in the DB2 region itself.

It should be noted that during execution of our application we did not perform any actual update of files or DB2 tables, strictly read-only functions. Given that, you can still see that the time represented by connector **C** is still about 3 ms in the DB2 region.

# **CEDF SCREENS FOR DB2 INTERFACE**





This section of the webinar is simply to show the use of CEDF to stop on SQL statements in a similar fashion to the EXEC CICS commands.

On this slide we see the before execution of the SELECT statement to obtain the average and maximum SALARY. We do not actually see the complete SQL statement, but the STMT= value ties directly back to the COBOL compile listing. We can see the “host data variable” being supplied by the WHERE clause, even if only in hexadecimal form.

### CEDF Screens for DB2 Commands

```

TRANSACTION: THZS PROGRAM: THZ005 TASK: 0000500 APPLID: TCICSA3 DISPLAY: 00
STATUS: COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER DSNCSQL
EXEC SQL SELECT                                P.AUTH=CICUSER, S.AUTH=
PLAN=CICSPL, DBRM=THZ005, STMT=00466, SECT=00002
SQL COMMUNICATION AREA:
SQLCASC = 136                                AT X'28B10274'
SQLCODE = 000                                AT X'28B10278'
SQLERRML = 000                                AT X'28B1027C'
SQLERRMC = ''                                AT X'28B1027E'
SQLRRP = 'DSN'                                AT X'28B102C4'
SQLRRD(1-6) = 000, 000, 00000, -1, 00000, 000 AT X'28B102CC'
SQLWRN(0-A) = '-----'                    AT X'28B102E4'
SQLSTATE = 00000                                AT X'28B102E8'
+
OVAR 001: TYPE=DECIMAL, LEN=09.02            AT X'28B10408'
+
  DATA=X'003015666C'
+
OFFSET:X'00DAD4' LINE:UNKNOWN EIBFN=X'0402'

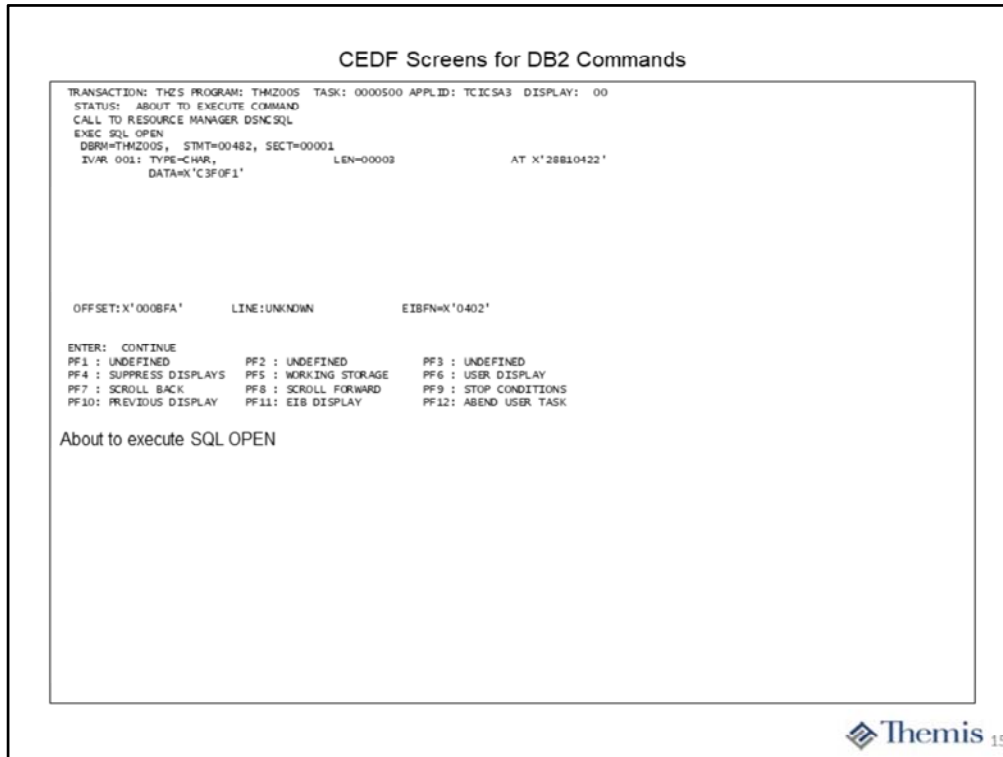
ENTER: CONTINUE
PF1 : UNDEFINED PF2 : UNDEFINED PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK

SQL SELECT execution complete
    
```



On this slide we can see the completion of the SQL SELECT statement execution. Once again we do not see the complete statement but can directly tie back to the compile listing by the STMT= option. We can see the first data variable being returned (OVAR 001) in packed decimal format. If we were to place the cursor on the plus sign below OVAR 001 and hit enter, we could see all of the data variables being returned.

We also see most of the attributes from the SQLCA including the most important ones like SQLCODE and SQLSTATE, which indicate successful completion of the statement.



On this slide we see the before execution of the SQL OPEN statement. Once again we do not see the complete statement, but can tie back to the compile listing by the STMT= option. We can also see the host data variable value being used on the WHERE clause.

### CEDF Screens for DB2 Commands

```
TRANSACTION: THZ5 PROGRAM: THZ005 TASK: 0000500 APPLID: TCIC5A3 DISPLAY: 00
STATUS: COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER DSNCSQL
EXEC SQL OPEN                                P.AUTH=CICUSER, S.AUTH=
PLAN=CIC5PL, DBRM=THZ005, STMT=00482, SECT=00001
SQL COMMUNICATION AREA:
SQLCASC = 136                                AT X'28B10274'
SQLCODE = 000                                AT X'28B10278'
SQLERRML = 000                                AT X'28B1027C'
SQLERRMC = ''                                AT X'28B1027E'
SQLRRP = 'DSN'                                AT X'28B102C4'
SQLRRD(1-6) = 000, 000, 0000, -1, 0000, 000 AT X'28B102CC'
SQLRRN(0-A) = ' N _ _ _ _ 1 _ _ _ _ _ ' AT X'28B102E4'
SQLSTATE = 00000                                AT X'28B102EF'
```

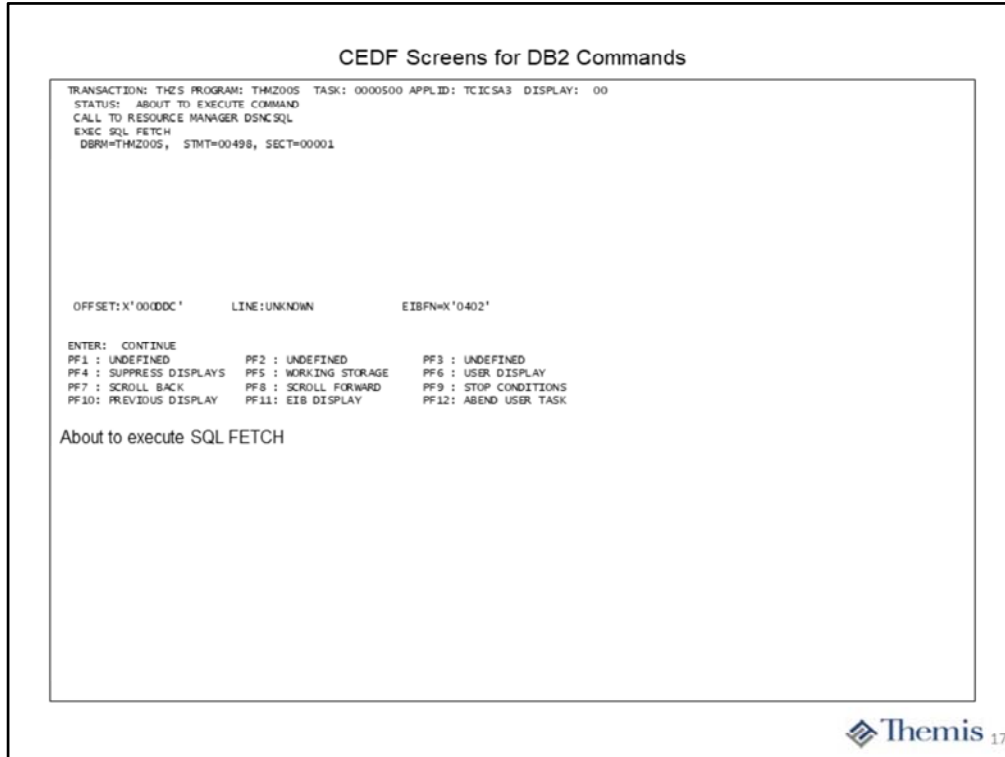
OFFSET:X'000BFA' LINE:UNKNOWN EIBFN=X'0402'

```
ENTER: CONTINUE
PF1 : UNDEFINED PF2 : UNDEFINED PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK
```

SQL OPEN execution complete

This slide shows the completion of the SQL OPEN statement. We can easily see the successful completion by )'s in both the SQLCODE and SQLSTATE.





Here is the before execution of the SQL FETCH statement. We can see the complete statement with all of its attributes by using the STMT= option to tie back to the COBOL compile listing.

### CEDF Screens for DB2 Commands

```
TRANSACTION: THZS PROGRAM: THZ005 TASK: 0000500 APPLID: TCIC5A3 DISPLAY: 00
STATUS: COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER DSNCSQL
EXEC SQL FETCH                                P.AUTH=CICUSER, S.AUTH=
PLAN=CIC5PL, DBRM=THZ005, STMT=00498, SECT=00001
SQL COMMUNICATION AREA:
SQLCASC = 136                                AT X'28B10274'
SQLCODE = 000                                AT X'28B10278'
SQLRRML = 000                                AT X'28B1027C'
SQLRRMC = ''                                 AT X'28B1027E'
SQLRRP = 'DSN'                               AT X'28B102C4'
SQLRRD(1-6) = 000, 000, 00000, -1, 00000, 000 AT X'28B102CC'
SQLWRN(0-A) = '-----'                     AT X'28B102E4'
SQLSTATE = 00000                             AT X'28B102EF'
OVAR 001: TYPE=CHAR, LEN=00006               AT X'28B102F4'
+ DATA=X'F0F0F0F0F0F0F0'
OFFSET:X'000DDC' LINE:UNKNWN EIBFN=X'0402'

ENTER: CONTINUE
PF1 : UNDEFINED PF2 : UNDEFINED PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK

SQL FETCH execution complete
```



Here we see the completion of the SQL FETCH execution. We see the first data attribute (OVAR 001) returned, but can see them all if we expand the list at the plus sign. Again the SQLCODE and SQLSTATE indicate successful completion of the FETCH statement.

### CEDF Screens for DB2 Commands

```
TRANSACTION: THZ5 PROGRAM: THZ2005 TASK: 0000500 APPLID: TCIC5A3 DISPLAY: 00  
STATUS: ABOUT TO EXECUTE COMMAND  
CALL TO RESOURCE MANAGER DSNCSQL  
EXEC SQL FETCH  
DBRM=THZ2005, STMT=00498, SECT=00001
```

```
OFFSET:X'000DDC' LINE:UNKNOWN EIBFN=X'0402'
```

```
ENTER: CONTINUE  
PF1 : UNDEFINED PF2 : UNDEFINED PF3 : UNDEFINED  
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY  
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS  
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK
```

About to execute SQL FTECH

This slide indicated another execution of the same SQL FETCH statement as on the previous slide. This is the before execution screen for the statement.

### CEDF Screens for DB2 Commands

```
TRANSACTION: THZS PROGRAM: THZ005 TASK: 0000500 APPLID: TCIC5A3 DISPLAY: 00
STATUS: COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER DSNCSQL
EXEC SQL FETCH                                P.AUTH=CICUSER, S.AUTH=
PLAN=CICSP1, DBRM=THZ005, STMT=00498, SECT=00001
SQL COMMUNICATION AREA:
SQLCASC = 136                                AT X'28B10274'
SQLCODE = +100                               AT X'28B10278'
SQLRRML = 000                               AT X'28B1027C'
SQLRRMC = ''                                AT X'28B1027E'
SQLRRP = 'DSNKEBR'                          AT X'28B102C4'
SQLRRD(1-6) = -110, 000, 00000, -1, 00000, 000 AT X'28B102CC'
SQLWRN(0-A) = '-----'                    AT X'28B102E4'
SQLSTATE = 02000                             AT X'28B102EF'
OVAR 001: TYPE=CHAR, LEN=00006              AT X'28B102F4'
+ DATA=X'F0F0F0F1F3F0'
OFFSET:X'000DDC' LINE:UNKNOW EIBFN=X'0402'

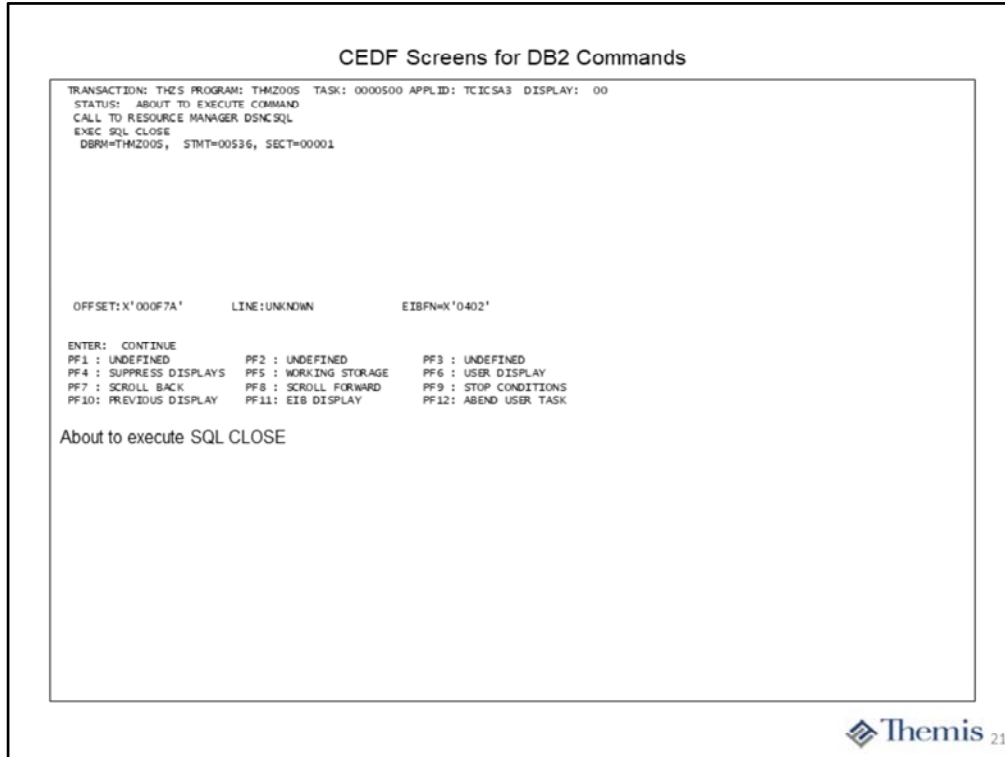
ENTER: CONTINUE
PF1 : UNDEFINED PF2 : UNDEFINED PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK

SQL FETCH execution complete

Notice that the SQLCODE is +100, indicating that we have retrieved the last row
```



This slide is the completion of the SQL FETCH statement. In fact this is the last FETCH statement execution within the loop to process the entire results set. We can tell that we have processed the complete results set by the indication of the +100 in the SQLCODE. We have not shown the few intermediate FETCH statement executions which retrieved the rows from the results set. They would appear similar to the previous slides we covered.



We now see the before execution screen for the SQL CLOSE statement. By using the STMT= option we can get back to the compile listing to see which cursor we are closing.

### CEDF Screens for DB2 Commands

```
TRANSACTION: THZ5 PROGRAM: THZ005 TASK: 0000500 APPLID: TCIC5A3 DISPLAY: 00
STATUS: COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER DSNCSQL
EXEC SQL CLOSE                                P.AUTH=CICUSER, S.AUTH=
PLAN=CIC5PL, DBRM=THZ005, STMT=00536, SECT=00001
SQL COMMUNICATION AREA:
SQLCASC = 136                                AT X'28B10274'
SQLCODE = 000                                AT X'28B10278'
SQLERRML = 000                                AT X'28B1027C'
SQLERRMC = ''                                AT X'28B1027E'
SQLRRP = 'DSN'                                AT X'28B102C4'
SQLRRD(1-6) = 000, 000, 0000, -1, 0000, 000 AT X'28B102CC'
SQLWRN(0-A) = '-----'                    AT X'28B102E4'
SQLSTATE = 00000                                AT X'28B102EF'
```

OFFSET:X'00DF7A'      LINE:UNKNOWN      EIBFN=X'0402'

```
ENTER: CONTINUE
PF1 : UNDEFINED      PF2 : UNDEFINED      PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS      PF5 : WORKING STORAGE      PF6 : USER DISPLAY
PF7 : SCROLL BACK      PF8 : SCROLL FORWARD      PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY      PF11: EIB DISPLAY      PF12: ABEND USER TASK
```

SQL CLOSE execution complete

At last we see the final screen associated with our application for the SQL statements. Here we see the successful execution of the SQL CLOSE statement for the cursor used by our application.

# STATISTICS FOR DB2 INTERFACE


### DB2 Interface Statistics

```

DFHDB2014 10/06/2017 08:25:06 TCICSA3 Statistics report follows for DB1A
accessing DB2 DB1A
---COMMITTS-----
DB2ENTRY PLAN      CALLS  AUTHS  W/P HIGH  ABORTS  1-PHASE  2-PHASE
*COMMAND          0      0      0  0      0        0        0
*POOL DSNACCO      0      0      0  0      0        0        0
THMZENT CICSPL     7      0      0  1      0        1        0
DFHDB2020 10/06/2017 08:25:06 TCICSA3 The display command is complete.

DSNC DISP STAT interface command

```



This slide shows the basic statistics captured by the CICS DB2 interface. These can be viewed at any point in time by using the DSNB transaction with the DISP STAT option. The statistics for the special “command” and “pool” entries are shown, along with all associated DB2ENTRY resource definitions.

For each entry we can see the PLAN name being used in DB2. When the name is all \*’s, the dynamic plan exit is in effect for this entry. The next column, CALLS, indicates the total number of SQL statements executed on behalf of that entry. The column, AUTHS, indicates the times a “sign-on thread” request is issued for the entry. If the number of AUTHS is high relative to the number of CALLS you may want to look at bumping up the number of protected threads.

The W/P column indicates the number of times the request either had to wait or overflow to the POOL due to the fact that all of the defined threads were in use. The HIGH column indicates the high-water mark for the number of threads in use since the entry was first used.

The ABORST column indicates the number of abends or SYNCPOINT ROLLBACK requests performed by the application or caused due to -911 DB2 SQL errors.

The COMMITTS columns indicate the number of SYNCPOINT commands issued or implied by the EXEC CICS RETURN command. The 1-PHASE column is the number of single phase commits due to the fact that no recoverable data resources have been modified. The 2-PHASE column indicates the number of SYNCPOINT commands issued or implied by the EXEC CICS RETURN command due to recoverable data resources being modified.



### DB2 Interface Statistics

Applid TCICSA3	Sysid TCA3	Jobname TCICSA3	Date 10/06/2017	Time 08:35:26	CICS 7.0.0	PAGE 5		
<b>Dispatcher TCB Modes</b>								
Dispatcher Start Time and Date . . . . .			07:38:37.737445 10/05/2017					
Address Space Accumulated CPU Time . . . . .			0000:02:25.366793 (Not Reset)					
Address Space Accumulated SRB Time . . . . .			0000:00:07.572668 (Not Reset)					
Address Space CPU Time (Since Reset) . . . . .			0000:00:02.984917					
Address Space SRB Time (Since Reset) . . . . .			0000:00:00.175095					
TCB Mode	TCBs Attached Current	TCBs Attached Peak	Op. System Waits	Op. System Wait Time	Total TCB Dispatch Time	Total TCB CPU Time	DS TCB CPU Time	TCB CPU/Disp Ratio
QR	1	1	2,696	0000:35:23.545540	0000:00:02.390893	0000:00:02.301001	0000:00:00.751736	96.2%
RO	1	1	20	0001:56:19.458037	0000:00:00.214516	0000:00:00.094449	0000:00:00.003179	
CO	0	0	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
SZ	0	0	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
RP	0	0	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
FO	1	1	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
SL	1	1	1	0000:31:27.434895	0000:00:00.001014	0000:00:00.000953	0000:00:00.000139	
SO	1	1	2	0001:00:00.062669	0000:00:00.000284	0000:00:00.000335	0000:00:00.000194	
SP	1	1	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
EP	2	2	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
TP	2	2	2	0000:55:03.790810	0000:00:00.004954	0000:00:00.000665	0000:00:00.000467	
D2	1	1	71	0000:35:40.379856	0000:00:00.025334	0000:00:00.021246	0000:00:00.011583	
S8	0	0	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
L8	3	3	37	0002:28:42.563016	0000:00:00.019203	0000:00:00.021122	0000:00:00.005438	
L9	0	0	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
X8	0	0	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
X9	0	0	0	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	0000:00:00.000000	
T8	1	2	5	0002:00:03.249781	0000:00:00.040463	0000:00:00.025187	0000:00:00.004962	
Totals	15	2	5			0000:00:02.464960	0000:00:00.777701	
Partial CICS Dispatcher Domain statistics								



On this slide we see a set of CICS DS domain statistics. While not directly associated with the CICS DB2 interface, the timings for the L8 and L9 TCBs do include the amount of CPU and WAIT time consumed by the DB2 interface.

```

                                DB2 Interface Statistics
-----
AppId TCICSA3  Sysid TCA3  Jobname TCICSA3  Date 10/05/2017  Time 08:35:26  CICS 7.0.0  PAGE 43
DB2 Connection
DB2 Connection Name . . . . . : DB1A
DB2 Group Id . . . . . :
DB2 Sysid . . . . . : DB1A
DB2 Release . . . . . : 11.1.0
DB2 Connection Status . . . . . : CONNECTED
DB2 Connection Error . . . . . : SQLCODE
DB2 Standby Mode . . . . . : RECONNECT
DB2 Pool Thread Plan Name . . . . . : CICSPL
DB2 Pool Thread Dynamic Plan Exit Name . . :
Pool Thread Authtype . . . . . : USERID
Pool Thread Authid . . . . . :
Signid for Pool/Entry/Command Threads . . : TCICSA3
Create Thread Error . . . . . : N90D
Protected Thread Purge Cycle . . . . . : 00.30
Deadlock Resolution . . . . . : ROLLBACK
Non-Terminal Intermediate Syncpoint . . . : RELEASE
Pool Thread wait Setting . . . . . : WAIT
Pool Thread Priority . . . . . : EQUAL
Current Connection Limit (TCB Limit) . . . : 8
Current number of Connections with a TCB . . : 0
Peak number of Connections with a TCB . . . : 1
Current number of tasks on Conn Readyq . . . : 0
Peak number of tasks on Conn Readyq . . . . : 0
Resyncmember . . . . . : N/A
DB2 Connect Date and Time . . . : 10/05/2017 07:39:10.00878
Command Thread Authtype . . . . . : N/A
Command Thread Authid . . . . . : CICSUSER
Message TD Queue 1 . . . . . : CDB2
Message TD Queue 2 . . . . . :
Message TD Queue 3 . . . . . :
Statistics TD Queue . . . . . : CDB2
DB2 Accounting records by . . . . . : NONE
Thread ReuseLimit . . . . . : 1,000
Current number of Connections without a TCB . : 1
CICS DB2 interface statistics for Command and Pool Threads

```



Here is the set of CICS statistics directly associated with the CICS DB2 interface. At the top of the page you will see the named of the connected DB2 region, DB1A. Along with the date and time the connection was start, usually when the CICS region is started.


In the middle we see information about the “POOL” threads, including the PLAN name (CICSPL) use by DB2 for these threads. We can also see the authorization information for the POOL and COMMAND threads. This is normally the default CICS region user ID. We can also see some of the options chosen for the resource definition, which includes the TD queue for messages, priority of pool and command threads, and purge cycle time for protected threads.

Finally at the bottom we see the defined limits for the thread TCBS, number of tasks on connection ready queue, and the limit for thread reuse.

### DB2 Interface Statistics

Pool Thread Limit. . . . .	3	Number of Calls using Pool Threads. . . . .	0
Current number of Pool Threads. . . . .	0	Number of Pool Thread Signons. . . . .	0
Peak number of Pool Threads. . . . .	0	Number of Pool Thread Partial Signons. . . . .	0
Number of Pool Thread waits. . . . .	0	Number of Pool Thread Commits. . . . .	0
Current number of Pool Tasks. . . . .	0	Number of Pool Thread Aborts. . . . .	0
Peak number of Pool Tasks. . . . .	0	Number of Pool Thread Single Phase. . . . .	0
Current Total number of Pool Tasks. . . . .	0	Number of Pool Thread Creates. . . . .	0
Current number of Tasks on Pool Readyq. . . . .	0	Number of Pool Thread Reuses. . . . .	0
Peak number of Tasks on Pool Readyq. . . . .	0	Number of Pool Thread Terminates. . . . .	0
Current number of DSN Command threads. . . . .	0	Times reuse limit hit by a pool thread. . . . .	0
Peak number of DSN Command threads. . . . .	0	Number of DSN Command Calls. . . . .	0
DSNC Command Thread Limit. . . . .	1	Number of DSN Command Signons. . . . .	0
		Number of DSN Command Thread Creates. . . . .	0
		Number of DSN Command Thread Terminates. . . . .	0
		Number of DSN Command Thread Overflows. . . . .	0

CICS DB2 interface statistics for Command and Pool Threads continued



This slide is the continuation of the POOL and CONNECTION statistics. On the left side we see information about pool thread and task usage. We tend to track “Peak number of Pool Threads” and “Number of Pool Thread Waits”, which indicate potential bottle necks in pool thread utilization. In conjunction with the threads we also monitor “Peak number of Pool Tasks” and “Peak Number of Tasks on Pool Readyq”, which can indicate that tasks are waiting for execution due to no available threads.

On the right side we see many statistics related to pool and connection request types issued. Tracking the ones like “Signons”, “Creates” and “Terminates” can provide an indication of thread reuse. This is key since as the ration of these to the number of calls increases, this means that the overhead of interface processing is also increasing.

DB2 Interface Statistics			
Applid TCICSA3	Sysid TCA3	Jobname TCICSA3	Date 10/06/2017 Time 08:35:26 CICS 7.0.0 PAGE 44
<b>DB2 Entries</b>			
DB2Entry Name . . . . .	THMZENT1	DB2Entry Status . . . . .	ENABLED
DB2Entry Static Plan Name . . . . .	CIC3PL	DB2Entry Disabled Action . . . . .	POOL
DB2Entry Dynamic Plan Exit Name . . . . .		DB2Entry Deadlock Resolution . . . . .	ROLLBACK
Dynamic Plan Exit Concurrency Status . . . . .			
DB2Entry Authtype . . . . .	USERID	DB2Entry Accounting records by . . . . .	NONE
DB2Entry Authid . . . . .			
DB2Entry Thread Wait Setting . . . . .	WAIT	Number of calls using DB2Entry . . . . .	7
DB2Entry Thread Priority . . . . .	EQUAL	Number of DB2Entry Signons . . . . .	0
DB2Entry Thread Limit . . . . .	3	Number of DB2Entry Partial Signons . . . . .	0
Current number of DB2Entry Threads . . . . .	0	Number of DB2Entry Commits . . . . .	0
Peak number of DB2Entry Threads . . . . .	1	Number of DB2Entry Aborts . . . . .	0
		Number of DB2Entry Single Phase . . . . .	1
		Number of DB2Entry Thread Creates . . . . .	1
DB2Entry Protected Thread Limit . . . . .	1	Number of DB2Entry Thread Reuses . . . . .	0
Current number of DB2Entry Protected Threads . . . . .	0	Number of DB2Entry Thread Terminates . . . . .	1
Peak number of DB2Entry Protected Threads . . . . .	1	Number of DB2Entry Thread Waits/Overflows . . . . .	0
		Times reuse limit hit by DB2ENTRY thread . . . . .	0
Current number of DB2Entry Tasks . . . . .	0		
Peak number of DB2Entry Tasks . . . . .	1		
Current Total number of DB2Entry Tasks . . . . .	1		
Current number of Tasks on DB2Entry Readyq . . . . .	0		
Peak number of Tasks on DB2Entry Readyq . . . . .	0		
CICS DB2 interface statistics for Entry Threads			



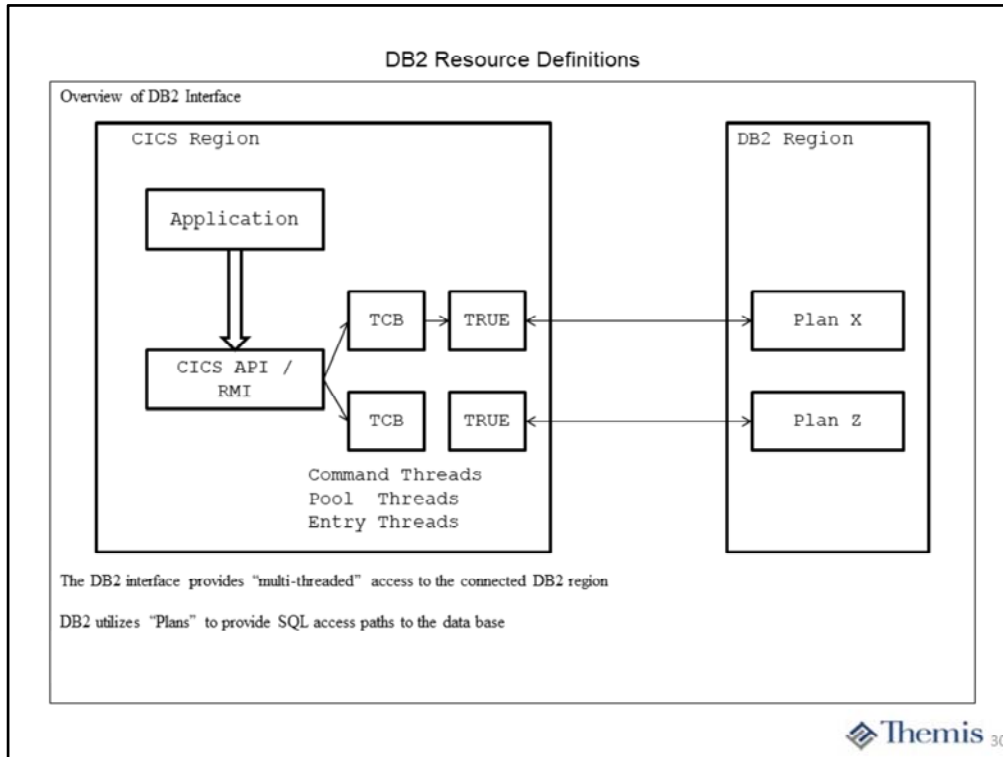
This slide represents on DB2ENTRY resource set of statistics. You will receive a separate set of these statistics for each defined resource. On the top of the page we see the information about the resource definition to include its name and associated DB2 PLAN.

In the middle on the left we see information about thread usage, which includes their priority in relation to CICS TCBS, “Peak number of DB2Entry Threads”, and “Peak number of DB2Entry Protected Threads”. These can be used to indicate the potential for waits on threads for the associated transactions in this entry.

In the middle on the right we find statistics related to the different request types being issued. Here we tend to track the “Signons”, “Created” and “Terminates”, all of which provide an indication of thread reuse.

Finally at the bottom of the slide we find “Peak number of Db2Entry Tasks” and “Peak number of Tasks on Db2Entry Readyq”, which can provide an indication of tasks waiting due to not enough threads allocated.

# DB2 RESOURCE DEFINITIONS



This slide provides an overview of the CICS DB2 interface. It is implemented as a TRUE in CICS, which is provided by the CICS product itself. Any support or enhancements will be provided by the CICS Development groups.

The CICS DB2 interface is designed to provide a "multi-threaded" interface with the connected DB2. CICS can establish a connection with only one DB2 region at a time. This is controlled by the DB2CONN resource described later.

There will be three different style of threads managed by the interface:

- 1) Command threads – on which any DB2 command will be issued
- 2) Pool threads – which will be utilized for applications not associated with a DB2ENTRY resource
- 3) Entry threads – used in support of applications associated with a specific DB2ENTRY resource

Pool threads will consume more processor time since they will have to perform DB2 "sign-on" function with each request. This is due to the fact that the thread may be used by different applications. If many applications are using pool threads this can add as much as 4-7 % overhead in processing these requests.

Entry threads will be defined and dedicated to a given application to ensure better thread reuse and reduce the overhead required to process the DB2 requests.

Only one command thread is required since CICS will typically issue only one DB2 command at a time.

There will be defined "Plans" used to allow DB2 to choose access paths to be used with these requests. This is especially true for Pool and Command threads.

## DB2 Resource Definitions

### Define DB2CONN:

```
DEFINE DB2CONN(DB1A) GROUP (THM200)
DESCRIPTION (CONNECTION TO DB1A - DB2 V11)
TCBLIMIT (8) PURGECYCLE (0,30)
COMAUTHID (CICSUSER) COMTHREADLIM (1)
PLAN (CICSPL) AUTHTYPE (USERID) THREADLIMIT (3)
DROLLBACK (YES) PRIORITY (EQUAL)
```

TCBLIMIT(8) – specifies the maximum number of allowable "threads" between CICS and DB2

PURGECYCLE(0,30) – specifies the number of minutes and seconds to keep "protected threads" active

COMTHREADIM(1) – specifies the number of "command threads"

THREADLIMIT(4) – specifies the number of "pool threads"

PRIORITY(EQUAL) – specifies the priority of DB2 interface threads with respect to other CICS TCBS

Threads are required to perform an DB2 request (SQL or command)



This slide shows the DB2CONN(DB1A) resource definition, maximum 8 characters. Only one of these type of resource definitions can be active in the CICS region at any point in time.

The **TCBLIMIT(8)** specified the maximum number of L8 TCBS associated with the CICS DB2 interface, the value ranges from 4 to 2000 with a default of 12. The value cannot be equal to or greater than SI parameter MAXOPENTCBS. The value specified here impacts the DB2ENTRY resources you define, plus the Pool threads specified on this definition.

The option **PURGECYCLE(0,30)** specifies the number of minutes and seconds which "protected threads" will be allowed to remain inactive before their TCB can be utilized by another thread. By keeping threads protected, you can reduce the processor overhead required by the interface.

The option **COMAUTHID(CICSUSER)** specifies the user ID to be associated with requests issued on the command thread to DB2. This user ID will be use the authorize the use of the command in the DB2 region. You will have to work with the DB2 SYSADM to ensure that this user ID is properly define to DB2 security. Do not be too concerned about using the generic CICSUSER since you will control who can use the DSNC transaction which will issue these commands.

The option **COMTHREADLIM(1)** specifies the number of threads allocated as command threads.

The option **PLAN(CICSPL)** specifies the name of the DB2 plan to be associated with the pool threads.

The option **AUTHTYPE(SUSERID)** specifies that the user ID associated with pool transaction is to be used in conjunction with the pool thread request.. Again you will have to work with the DB2 SYSADM to determine the security requirements for the DB2 region.

## DB2 Resource Definitions

### Define DB2CONN:

```
DEFINE DB2CONN(DB1A) GROUP (THM200)
  DESCRIPTION (CONNECTION TO DB1A - DB2 V11)
  TCBLIMIT (8) PURGECYCLE (0,30)
  COMAUTHID (CICSUSER) COMTHREADLIM (1)
  PLAN (CICSPL) AUTHTYPE (USERID) THREADLIMIT (3)
  DROLLBACK (YES) PRIORITY (EQUAL)
```

TCBLIMIT(12) – specifies the maximum number of allowable "threads" between CICS and DB2

PURGECYCLE(0,30) – specifies the number of minutes and seconds to keep "protected threads" active

COMTHREADIM(1) – specifies the number of "command threads"

THREADLIMIT(4) – specifies the number of "pool threads"

PRIORITY(EQUAL) – specifies the priority of DB2 interface threads with respect to other CICS TCBS

Threads are required to perform an DB2 request (SQL or command)



The option **THREADLIMIT(4)** specifies the number of pool threads which may be in use at any point in time. This value cannot exceed the TCBLIMIT value specified.

The option **DROLLBACK(YES)** specifies that the CICS DB2 interface will issue an EXEC CICS SYNCPOINT ROLLBACK in a DB2 detected deadlock and a -911 SQLCODE will be returned to the application.

The option **PRIORITY(EQUAL)** specifies the scheduling prior of the CICS DB2 thread TCBS in relationship to other CICS TCBS, default is HIGH. The value of EQUAL means that the thread TCBS will have the same dispatching priority as the CICS TCBS. The value of HIGH (default) places the thread TCBS at a higher dispatching priority than the CICS TCBS. This is really not necessary unless you are CPU bound or experience waits for processors in your system. Having the TCBS with HIGH can adversely affect CICS processing if a DB2 application request is working with LOBs or performs a high number of SQL requests.



**DB2 Resource Definitions**

**Define DB2ENTRY:**

```

DEFINE DB2ENTRY(THMZENT1) GROUP(THMZ00)
DESCRIPTION(DB2 ENTRY FOR THZ* TRANSACTIONS)
TRANSID(THZ*) PLAN(CICSPL) AUTHTYPE(USERID)
PROTECTNUM(1) THREADLIMIT(2) THREADWAIT(YES)
DROLLBACK(YES) PRIORITY(EQUAL)

```


PROTECTNUM(1) – specifies the number of “protected threads”

THREADLIMIT(2) – specifies the number of “entry threads”

THREADWAIT(YES) – specifies action to be taken when THREADLIMIT exceeded

PRIORITY(EQUAL) – specifies the priority of DB2 interface threads with respect to other CICS TCBS

Threads are required to perform an DB2 request (SQL or command)

 33

This slide show a DB2ENTRY(THMZENT1) resource definition, maximum 8 characters. You will want to have one or more of these for each application making SQL requests to DB2.

The option **TRANSIN(THZ\*)** specifies the pattern of transaction codes which will be associated with this DB2ENTRY resource. The value you specify may in fact be a specific transaction code or you can use wildcards as in our example. An asterisk (\*) indicates one or more characters may be eliminated from consideration. You may also use a plus sign (+) to eliminate a particular transaction code position from consideration, i.e. THZ+ would mean that any character is acceptable in the fourth position.

The option **PROTECTNUM(1)** specifies the number of “protected threads” to be associated with the entry. Protected threads for high used DB2 entries can improve performance since they do not have to go through the DB2 “sign on” process with each different application. You do not want to have all threads being protected as this can actually affect response times due to the fact that protect thread TCBS are kept in their current state for the amount of time specified in the PURGECYCLE option.

The option **THREADLIMIT(2)** specifies the number of threads which can be in use with this entry at any point in time. You should base this number on the expected number of concurrently executing transactions for these applications. Care should be taken that the total of THREADLIMIT for all define DB2ENTRY resources plus the THREADLIMIT for pool threads does not exceed the TCBLIMIT specified on the DB2CONN resource.

The option **THREADWAIT(YES)** specifies that when the THREADLIMIT is reached any new requests will wait for a thread until one of the current executing applications ends. Another choice is **POOL**, which means the request will transition over to a pool thread to complete. This will add some overhead since the request will have to go through the DB2 “sign on” process as required for all pool thread requests. If you special a value of **NO**, the application will be abended with an abend code of AD2P.

## DB2 Resource Definitions

### Define DB2ENTRY:

```
DEFINE DB2ENTRY (THM2ENT1) GROUP (THM200)
DESCRIPTION (DB2 ENTRY FOR THZ* TRANSACTIONS)
TRANSID (THZ*) PLAN (CICSPL) AUTHTYPE (USERID)
PROTECTNUM (1) THREADLIMIT (2) THREADWAIT (YES)
DROLLBACK (YES) PRIORITY (EQUAL)
```

PROTECTNUM(1) – specifies the number of "protected threads"

THREADLIMIT(2) – specifies the number of "entry threads"

THREADWAIT(YES) – specifies action to be taken when THREADLIMIT exceeded

PRIORITY(EQUAL) – specifies the priority of DB2 interface threads with respect to other CICS TCBs

Threads are required to perform an DB2 request (SQL or command)



The option **DROLLBACK(YES)** specifies that the CICS DB2 interface will issue an EXEC CICS SYNCPOINT ROLLBACK in a DB2 detected deadlock and a -911 SQLCODE will be returned to the application.

The option **PRIORITY(EQUAL)** specifies the scheduling prior of the CICS DB2 thread TCBs in relationship to other CICS TCBs, default is HIGH. The value of EQUAL means that the thread TCBs will have the same dispatching priority as the CICS TCBs. The value of HIGH (default) places the thread TCBs at a higher dispatching priority than the CICS TCBs. This is really not necessary unless you are CPU bound or experience waits for processors in your system. Having the TCBs with HIGH can adversely affect CICS processing if a DB2 application request is working with LOBs or performs a high number of SQL requests.

## DB2 Resource Definitions

### Define DB2TRAN:

```
DEFINE DB2TRAN (THMZTRN) GROUP (THMZ00)  
DESCRIPTION (DB2 TRAN ENTRY FOR ADDITIONAL TRANSACTIONS)  
ENTRY (THMZENT1) TRANSID (THM+)
```

ENTRY(THMZENT1) – specifies the name of the DB2ENTRY with which this DB2TRAN resource should be assigned to

TRANSID(THM+) - specifies the name of the transaction(s) to be associated with this resource

This resource definition can be useful when CICS TRANSIDs do not follow a discernible pattern.

On this slide we see an example for a DB2TRAN(THMZTRN) resource definition, maximum 8 characters. The style of resource is used to associate certain transaction codes with an existing DB2ENTRY resource.

The option **ENTRY(THMZENT1)** specifies the name of the DB2ENTRY to associated this resource with.

The option **TRANSID(THM+)** specifies the transaction code or pattern for the codes to be associated with this resource.

This style of resources definitions allows you to associate several different patterns with the same set of options for thread processing as defined by the DB2ENTRY.

## DB2 Resource Definitions

### Dynamic Plan Exit:

#### Code snippet from SDFHSAMP library, member DFHD2PXT:

```
CONT2 DS OH
*
*
*****
*****
*   INSERT CODE TO UPDATE THE DB2 PLAN (CPRMPLAN) AND/OR   *
*   SPECIAL USER FIELD (CPRMUSER) .                         *
*****
*****
*
*
EXEC CICS RETURN
```

We have chosen to show the member DFHD2PXT from the DFH530.CICS.SDFHSAMP library,

This is due to the fact that the plan exit program itself is defined as THREADSAFE, minimize TCB switching

The supplied module does not set a valid plan, so you must add the appropriate code

The following fields are passed to the exit program via a COMMAREA:

- CPRMPLAN – 8 character DBRM/plan name of the first SQL statement on entry to program, can be modified
- CPRMAUTH – 8 character authorization ID that is passed to DB2 at signon time, cannot be modified
- CPRMUSER – 4 byte area reserved for use by the sample program.
- CPRMAPPL – 8 character name of the application program

If you cannot specify a particular plan for the DB2ENTRY resource or you are using POOL threads for different applications, then you may have to implement a dynamic plan exit program. You specify this fact by using the **PLANEXITNAME()** attribute instead of the PLAN() attribute on the DB2CONN and DB2ENTRY resource definitions. We have chosen to use the DFHD2PXT program supplied by IBM in assembler source since it is defined to CICS with the CONCURRECY(THREADSAFE) attribute.

# **SVC DUMP INFORMATION FOR DB2 INTERFACE**

### DB2 Interface Information in an SVC Dump

```

CICS DUMP: SYSTEM=TCICSA3 CODE=MT0001 ID=3/0013 10 14:54:16 08/13/14
==DB2: CICS/DB2 - SUMMARY

==DB2: GLOBAL STATE SUMMARY

Db2conn name:          DB1A
Connection status:     Connected
In standby mode:       No
DB2 id:                DB1A
DB2 Group id:          1110
DB2 release:           1110
Operating in OpenAPI mode: Yes
Service task started:  Yes
Master subtask started: No - not required
Tcb limit:             8
Currently active connections: 1


Message Queue1:        DB2
Message Queue2:
Message Queue3:
Statistics Queue:     DB2
Standby mode:          Reconnect
Connect error:         sqlcode

==DB2: TRANSACTION SUMMARY

Tran Task  TcaAddr  TieAddr  LotAddr  Rctename  RcteAddr  CsubAddr  Correlation  Uowid          Subtask  Tcb
id  num                                     id                                               id                                     id
-----
THEY 00300 266A4100 29034190 29034210 THIZENT 28243030 27A27030 ENTRHZY0001 D2692A0F6EA9A1C0 N/A      Yes

```

The key portion is the "TRANSACTION SUMMARY"



This slide contains the "CICS DB2 Interface Summary" from an SVC dump. The upper portion simply contains information about the DB2 connection. This includes the name of the DB2 region and its version. Another key piece of information is the TCB limit, plus the current number of tasks connected.

The "TRANSACTION SUMMARY" show all of the tasks in the CICS region which have or are currently accessing DB2 resources. Much of the information is of minimal interest, such as TcaAddr, TieAddr, RctAddr, and Subtask running. However, the Correlation id is useful is you want to cancel the thread. The LotAddr and CsubAddr (shown next) are the control blocks used for the task during execution. The Tcb in DB2 indicates if the task currently has a DB2 request outstanding.

## DB2 Interface Information in an SVC Dump

```

==LOT CONTROL BLOCKS AND CONNECTED CSUBS(ACTIVE THREADS)

DFHD2LOT 29034210 LOT
0000 00DE6EC4 C6C8C4F2 D3D6E340 40404040 E3C8E9E8 266A4100 28243030 27A27030 *...DFHD2LOT  THEY.....S..* 29034210
0020 00000000 282352DC 00000000 282430C4 2A30AC34 00000000 00000000 FF588B00 *.....D.....* 29034230
0040 00000000 00000000 00000000 00000000 00000000 C3C9C3E2 D7D34040 01010101 *.....CICSPL ....* 29034250
0060 8000C000 00000080 00000000 00000000 E3C3C9C3 E2C1F340 D2692A0F 6EA9A1C0 *..{.....TCICSA3 K...>.* 29034270
0080 C3C9C3E2 E4E2C5D9 40404040 40404040 00000000 00000000 E2F0E6F1 40404040 *CICSUSER .....SOWI * 29034290
00A0 E3C9C9C2 E2C1F340 692A0FEE A9A1C6D9 C2400003 00012A30 AC340000 00000000 *TCICSA3 ...>.RRB .....* 290342B0
00C0 00000000 00000018 04050001 04000000 00000000 00000000 00000000 0000 *.....* 290342D0

DFHD2CSB 27A27030 CSUB
0000 03D06EC4 C6C8C4F2 C3E2C740 40404040 D269246C B4B4D800 2829F400 28243030 *...DFHD2CSB  K..N..Q...4.....* 27A27030
0020 29034210 008AC188 00000000 00000000 D2692A0F 6EA9A1C0 00000000 00000000 *.....Ah.....K...>Z.{.....* 27A27050
0040 2824309C 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A27070
0060 00000000 00000000 00000000 C3C9C3E2 D7D34040 C3C9C3E2 E4E2C5D9 40404040 *.....CICSPL CICSUSER * 27A27090
0080 40404040 C5D5E3D9 E3C8E9E8 F0F0F0F1 00000000 D269246C B8E52200 00000000 * ENTRTHZY0001...K..N.V.....* 27A270B0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
0260 6E6EE399 81838540 E2A38199 A3406E6E C900300C C1D7C940 00000000 00000000 *>>Trace Start >>I...API .....* 27A27290
0280 CA00300C C1D7C940 00000000 00000000 C800300C C1D7C940 00000000 00000000 *...API .....API .....* 27A272B0
02A0 CC00300C C1D7C940 00000000 00000000 CD00300C C1D7C940 00000000 00000000 *...API .....API .....* 27A272D0
02C0 CE00300C C1D7C940 00000000 00000000 CF00300C C1D7C940 00000000 00000000 *...API .....API .....* 27A272F0
02E0 0000300C C1D7C940 00000000 00000000 C700300C C1D7C940 00000000 00000000 *)...API .....G...API .....* 27A27310
0300 C800300C C1D7C940 00000000 00000000 4C4CE399 81838540 C5958440 40404C4C *H...API .....<<Trace End <<* 27A27330

The 2 control blocks used while the task is executing

No longer documented, IBM dropped the Supplementary Data Areas manual
    
```



Here we find the Life Of Task (LOT) and Connection Subtask (CSUB) control blocks. These data areas are used while the task is execution to manage the application making requests to DB2. These data area are no longer documented. However, the last Supplemental Data Areas manual was for CICS TS V4.1 and these control blocks seem to have not changed with subsequent releases.

The LOT basic format is as follows:

- +0 Half-word length of data area
- +2 14 character eye-catcher
- +14 Address of TCA
- +18 Address of RCTE (Resource Control Table Entry), DB2ENTRY resource
- +1C Address of CSUB
- +34 ECB used for CICS task wait
- +54 DB2 Plan in use, 8 bytes
- +70 16 byte UR token
- +80 8 character primary authorization ID
- +88 8 character secondary authorization ID
- +98 22 byte accounting token for DB2

The CSUB basic format is as follows:

- +0 Half-word length of data area
- +2 14 character eye-catcher
- +1C Address of RCTE (Resource Control Table Entry), DB2ENTRY resource
- +20 Address of LOT
- +6C DB2 Plan name, 8 bytes
- +74 8 character primary authorization ID
- +7C 8 character secondary authorization ID
- +84 12 character Correlation ID, 4 byte type, 4 byte transid, and 4 byte thread number
- +9C 22 byte accounting token for DB2

### DB2 Interface Information in an SVC Dump

```


DFHD2CSB 27A27030 CSUB
0000 03206EC4 C6C8C4F2 C3E2C240 40404040 D269246C B4B4D800 2829F400 28243030 *...DFHD2CSB K..%.Q...4....* 27A27030
0020 29034210 0084C188 00000000 00000000 D2692A0F 6E9A91C0 00000000 00000000 *.....Ah.....K...z.{.....* 27A27050
0040 2824309C 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A27070
0060 00000000 00000000 00000000 C3C3C3E2 D7D34040 C3C9C3E2 E4E2C5D9 40404040 *.....CICSPL CICSUSER * 27A27090
0080 40404040 C5D5E3D9 E3C8E9E6 F0F0F0F1 00000000 D269246C B8E52200 00000000 * ENRTRHEZY0001...K..%.V.....* 27A270B0
00A0 00000000 00000000 00000000 00000000 00000000 46800000 00000001 00000000 *.....* 27A270D0
00C0 00000000 00000000 00000000 40404040 40404040 40404040 40404040 F5F8B800 *.....* 27A270F0
00E0 C09C240 00030001 2A30AC24 00000000 00000000 00000000 0018040E 00010400 *FRB.....* 27A27110
0100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A27130
0120 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A27150
0140 00000000 00000000 00000000 00000000 00000000 00000000 A7A27110 00000000 *.....XS.....* 27A27170
0160 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A27190
0180 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A271B0
01A0 00000000 00000000 00000000 00000000 40404040 40404040 40404040 40404040 *.....* 27A271D0
01C0 40404040 40404040 40404040 40404040 40404040 40404040 40404040 *.....* 27A271F0
01E0 40404040 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A27210
0200 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A27230
0220 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 27A27250
0240 00000000 00000000 00000000 00000000 00000000 00000000 0001C7D0 27A27310 *.....G.s...* 27A27270
0260 6E6EE399 81838540 E2A38199 A340666E C900300C C1D7C940 00000000 00000000 *>>Trace Start >>I...API.....* 27A27290
0280 CA00300C C1D7C940 00000000 00000000 C800300C C1D7C940 00000000 00000000 *...API.....API.....* 27A272B0
02A0 CC00300C C1D7C940 00000000 00000000 CD00300C C1D7C940 00000000 00000000 *...API.....API.....* 27A272D0
02C0 CE00300C C1D7C940 00000000 00000000 CF00300C C1D7C940 00000000 00000000 *...API.....API.....* 27A272F0
02E0 D000300C C1D7C940 00000000 00000000 C700300C C1D7C940 00000000 00000000 *)...API.....G...API.....* 27A27310
0300 C800300C C1D7C940 00000000 00000000 4C4CE399 81838540 C5958440 40404C4C *H...API.....<<Trace End <<* 27A27330

```

Here is the complete CSUB

Notice the small DB2 trace table at the end of the data area

It is being supplied by the DB2 region to indicate how the requests have completed



The CSUB contains a small trace table being supplied by the DB2 region, which indicates the type of request plus how it has completed. This begins with the constant eye-catcher “>>Trace Start >>” at offset +260. The end is delimited by the constant eye-catcher “<<Trace End <<”. Each entry is 16 bytes in length and contains the following format:

- +0 Full word trace entry number
- +4 4 character type of DB2 request
- +8 reserved, 2 bytes
- +A 2 byte frb return code
- +C 4 byte frb return code

The first byte of the trace entry number (offset +0) is the actual hexadecimal number assigned to the request. The last 3 bytes contain the the packed-decimal task number.

The values for the DB2 request type are as follows:

- ABRT – Abort request
- ASSO – Associate request
- CTHD – Create thread request
- ERRH – Error request handler
- PREP – Prepare request
- SIGN – Full signon request
- TERM – Terminate thread request
- TSGN – Terminate signon request
- API – SQL or IFI request
- COMM – Commit request
- DISS – Dissociate request
- IDEN – Identify request
- PSGN – Partial signon request
- SYNC – Single phase commit
- TIDN – Terminate identify request
- \*REC – Recovery routine entered



**DB2 Interface Information in an SVC Dump**

```

==DS: TASKS SUMMARY
KEY FOR SUMMARY
T = TYPE OF TASK           S=SYSTEM N=NON-SYSTEM
S = STATE OF TASK         D=DISPATCHABLE S=SUSPENDED R=RUNNING A=RUNNING ABTERM YES J=RUNNING IN JVM E=RESUMED EARLY
F = PURGEABILITY FLAG    P=PURGEABLE N=NOT PURGEABLE
P = PURGE STATUS          N=NO PURGE X=PURGED P=PURGE PENDING A=ABTERM PENDING
TT = TIMEOUT TYPE        I=INTERVAL DD=DEADLOCK DELAYED DI=DEADLOCK IMMEDIATE
W = WAIT/SUSPEND TYPE    M=WAIT_MVS S=SUSPEND C=WAIT_DLDC W=WAIT_OLDW
DTA= DISPATCHER TASK AREA
AD = ATTACHING DOMAIN
MD = TASK MODE

DS_TOKEN KE_TASK T S F P TT RESOURCE RESOURCE_NAME W TIME OF TIMEOUT DTA AD ATTACHER MD SUSPAREA XML_TKN_TOKEN
TYPE
00020001 264L7000 S S N N - ENF NOTIFY M 09:34:48.034 - 26543200 DM 26610C00 RD 26610C18
00080003 2642F000 S S N N - TTEXPIRY DS_NUDGE S 10:18:30.845 - 26543680 TI 00630003 QR 26544980
-----
05920005 279FD000 N A 3725BF00 XM 26608300 LB 26608300000300C


CEMT I TA
RESULT - OVRTYPE TO MODIFY
Task(000060)
Trand(CEMT)
Facility(P068)
Runstatus(Suspended)

Htype(ZC10WAIT)
Hvalue(DFHZARQ1)
Htime(000046)
+ Indoubtmfns(000000)

SYSID=TCAS3 APPLID=TCICSA3
TIME: 13.54.51 DATE: 10/18/17
PF 1 HELP 2 HEX 3 END 5 VAR 7 SBH 8 SFH 10 SB 11 SF

Wait indications for CICS DB2 interface, mock up since it is difficult to capture a wait in the interface

```

 41

This slide contains a portion of the DS Domain summary and output of CEMT I TA transaction. We were not able to “catch” our application in a wait state. However, we can at least explain what you might see.

Within the DS Domain summary we are looking at our application performing an SQL request, still out in the DB2 region. If it were in a wait state the two indicators of this would be the values in the “RESOURCE TYPE” and “RESOURCE\_NAME” columns. Also, the “TIME OF SUSPEND” column would show when the application went into the wait state.

As for the CEMT display, we would have to “select” the transaction from the list to show the detailed information portrayed in our mockup. The wait state would be indicated by the **Htype** and **Hvalue** attributes. The **Htime** attribute would show the number of seconds the task has been in the wait state.

The following tables provide a brief explanation of the wait states for applications using the interface:

**Waits issued by the CICS DB2 TRUE:**

Resource type or Htype	Resource_name or Hvalue	Description	Purge or Forcepurge
DB2	LOT_ECB	CICS task is waiting for DB2, that is, waiting for the CICS DB2 task to complete the request, used with DB2 V5 or earlier	Forcepurge
CDB2RDYQ	Pool or name of DB2ENTRY	Task is waiting for a thread to become available	No
CDB2CONN		CICS task has an open TCB but is not waiting for a DB2 connection to become available to use with the TCB. Indicates that TCBLIMIT has been reached.	No



**DB2 Interface Information in an SVC Dump**

```

==DS: TASKS SUMMARY
KEY FOR SUMMARY
T = TYPE OF TASK           S=SYSTEM N=NON-SYSTEM
S = STATE OF TASK         D=DISPATCHABLE S=SUSPENDED R=RUNNING A=RUNNING ABTERM YES J=RUNNING IN JVM E=RESUMED EARLY
F = PURGEABILITY FLAG    P=PURGEABLE N=NOT PURGEABLE
P = PURGE STATUS          N=NO PURGE X=PURGED P=PURGE PENDING A=ABTERM PENDING
TT = TIMEOUT TYPE        I=INTERVAL DD=DEADLOCK DELAYED DI=DEADLOCK IMMEDIATE
W = WAIT/SUSPEND TYPE     M=WAIT_MVS S=SUSPEND C=WAIT_DLDC W=WAIT_OLDW
DTA= DISPATCHER TASK AREA
AD = ATTACHING DOMAIN
MD = TASK MODE

DS_TOKEN KE_TASK T S F P TT RESOURCE RESOURCE_NAME W TIME OF TIMEOUT DTA AD ATTACHER MD SUSPAREA XML_TKN_TOKEN
TYPE
00020001 264L7000 S S N N - ENF NOTIFY M 09:34:48.034 - 26543200 DM 26610C00 RO 26610C18
00080003 2642F000 S S N N - TTEXPIRY DS_NUDGE S 10:18:30.845 - 26543680 TI 00630003 QR 26544980
-----
05920005 279FD000 N A
37258E00 XM 26608300 LA 26608300000300C

CEMT I TA
RESULT - OVRTYPE TO MODIFY
Task(000060)
TranId(CEMT)
Facility(P068)
Runstatus(Suspended)
-----
Htype(ZC10WAIT)
Hvalue(DFHZARQ1)
Htime(000046)
+ Indoubtmfns(000000)

SYSID=TCAS3 APPLID=TCICSAS
TIME: 13.54.51 DATE: 10/18/17
PF 1 HELP 2 HEX 3 END 5 VAR 7 SBH 8 SFH 10 SB 11 SF

Wait indications for CICS DB2 interface, mock up since it is difficult to capture a wait in the interface

```

DB2 wait indicators

The following tables provide a brief explanation of the wait states for applications using the interface:

**EXEC CICS WAIT EVENT issued:**

Resource type or Htype	Resource_name or Hvalue	Description
DFND2EX2	PROTTERM	TERM call has been issued for a protected thread during the thread purge cycle
DFHD2STR	ATCHMSUB	The CICS DB2 startup program DFHD2STR has attached the master subtask program DFHD2MSB and is waiting for it to identify to DB2. (Only applies DB2 V5 or earlier.)
DFHD2STR	DTCHMSUB	The CICS DB2 startup program is waiting for the master subtask program DFHD2MSB to terminate. (Only applies DB2 V5 or earlier.)
DFHD2STP	MSBRETRN	The CICS DB2 shutdown program is waiting for the master subtask program DFHD2MSB to terminate. (Only applies DB2 V5 or earlier.)
DFHD2STP	CEX2TERM	The CICS DB2 shutdown program is waiting for the CICS DB2 service task CEX2 running program DFHD2EX2MSB to terminate all subtasks and terminate itself.

## **MORE DSNC COMMANDS**



## DSNC Commands

```
DSNT360I -BA *****
DSNT361I -BA * DISPLAY DATABASE SUMMARY
          * GLOBAL
DSNT360I -BA *****
DSNT360I -BA DATABASE = DTHM82 STATUS = RW
          DBD LENGTH = 20180
DSNT397I -BA *****
          NAME TYPE PART STATUS PHYERRLO PHYERRHI CATALOG PIECE
          -----
TS00DEPT TS RW
TS00BMP TS 0001 RW AREO*
TS00BMP TS 0002 RW
TS00BMP TS 0003 RW AREO*
TS00BMP TS 0004 RW
TS00BMP TS
TS00PROJ TS RW
XACT01 IX RW
XACT02 IX RW
XDEPT01 IX RW
XDEPT02 IX RW
XDEPT03 IX RW
XEMP01 IX 0001 RW
      -THRU 0004
XEMP01 IX
```

DSNC -DIS DB(DTHM82) command

Key  
P/N to go to next page  
P/P to go to previous page

This slide contains the first page of the DB2 command for displaying a particular database. This is typically a multiple page display. You are able to issue almost any DB2 command via DSNC transaction.

### DSNC Commands

```

XEMP02 IX L0001 RW
      -THRU 0004
XEMP02 IX L*
XEMP03 IX L0001 RW
      -THRU 0004
XEMP03 IX L*
XEMP13A4 IX RW
XEMP16C4 IX RW
XEMP1E45 IX RW
XEMPPROJ IX RW
XPRO1GL2 IX RW
XPROJ01 IX RW
XPROJ02 IX RW
XPROJ03 IX RW
XPROJ04 IX RW
XPROJ05 IX RW
XPROJAC0 IX RW
***** DISPLAY OF DATABASE DTHM82 ENDED *****
DSN8022I -8A DSNTE015 "DISPLAY DATABASE" NORMAL COMPLETION
DFHD82301 10/06/2017 14:08:12 TC:ICSA DSNC DB2 command complete.

```

DSNC-DIS DB(DTHM82) command continued

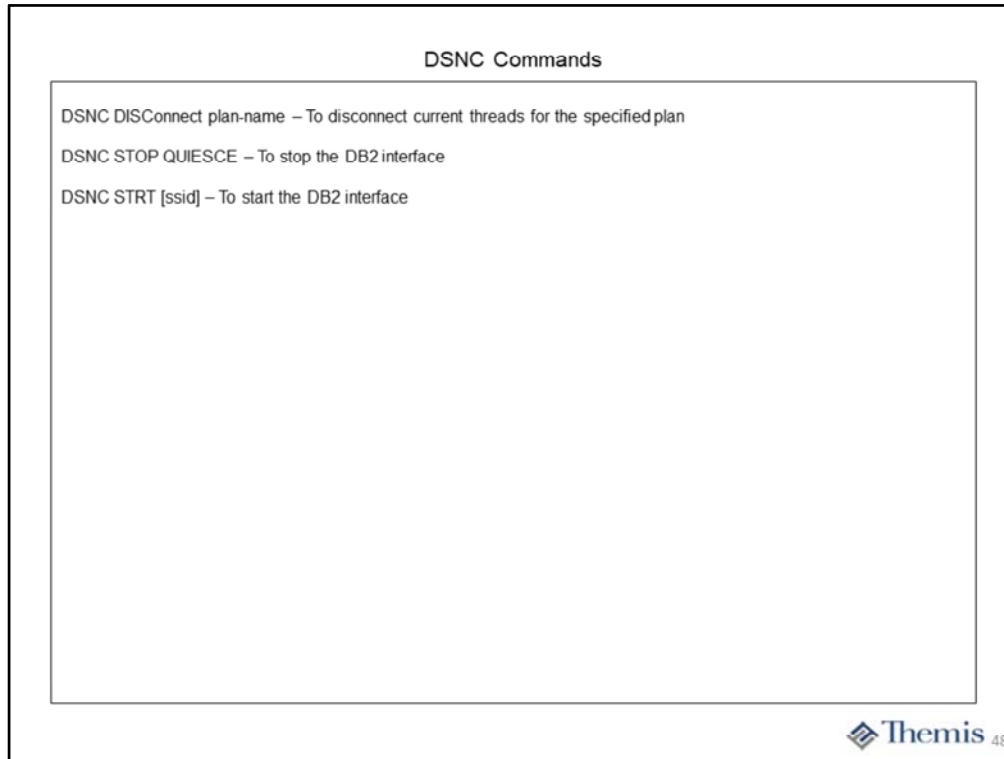
Key

P/N to go to next page

P/P to go to previous page

47

This slide contains the last page of the DB2 command output to display a particular database.



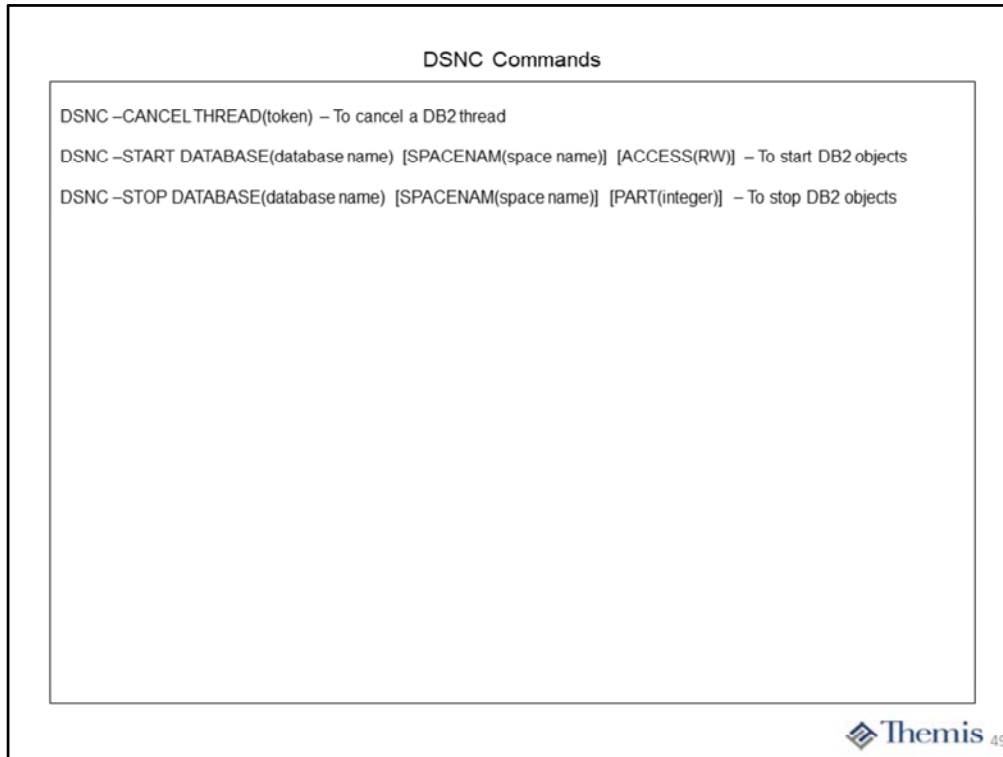
Here are three CICS DB2 interface commands which perform the functions shown.

The **DISConnect** command can be used to release threads for the specified plan. This can be used to allow a utility to execute if it is waiting for applications to end. It will not prevent new threads from being create. To prevent new threads from being created you will have to disable the transaction IDs associated with the DB2ENTRY resource.

The **STOP** command can be used to stop the CICS DB2 attachment facility entirely. The default option of QUIESCE will allow currently executing applications to end and prevent new threads from being created. You may also use the FORCE option to interrupt the currently executing applications and force them to roll back.

The **STRT** command will start up the CICS DB2 attachment facility after it has been stopped. You can optionally specify a different subsystem ID to connect to a different DB2 region than the one specified by the DB2CONN resource.





Here are three DB2 commands which perform the functions mentioned.

The **-CANCEL THREAD** command is used to terminate a particular thread executing within the CICS DB2 interface. The TOKEN is a 1 to 6 digit decimal number assigned to the thread, which can be determined by issuing the DSNC -DIS THD(\*) command.

The **-START DATABASE** command can be used to make a database or tablespace available for use. The data base name can include the asterisk (\*) wild card at the front or end of the specified string, to start all data bases matching the string. You may also specify the name as a range by supplying a pair of names separated by a comma. You can also supply the SPACENAM option to make available certain tablespaces of the database. The rule for specification of the space name are the same as those for the database name. The ACCESS option controls how the database or tablespace is brought online, default is RW for READ/WRITE status.

The **-STOP DATABASE** command can be used to take the specified database offline. The data base name can include the asterisk (\*) wild card at the front or end of the specified string, to stop all data bases matching the string. You may also specify the name as a range by supplying a pair of names separated by a comma. You can also supply the SPACENAM option to take certain tablespaces of the database offline. The rule for specification of the space name are the same as those for the database name. You can also include the PART option to take only certain partitions offline for a partitioned tablespace.

**DB2 RESOURCE DEFINITIONS –  
COMPLETE CEDA SCREENS**



## DB2 Resource Definitions

### Define DB2CONN:

```
OVERTYPE TO MODIFY                                CICS RELEASE = 0700
CEDA Alter DB2Conn( DC1A )
DB2Conn      : DC1A
Group       : TESTGRP
Description ==> TEST CONNECTION TO DC1A, DB2 V12 REGION
CONNECTION ATTRIBUTES
CONNECTerror ==> Sqlcode          Sqlcode | Abend
DB2Groupid  ==>
DB2Id       ==> DC1A
MSGQUEUE1   ==> CDB2
MSGQUEUE2   ==>
MSGQUEUE3   ==>
Nontermrel  ==> Yes              Yes | No
PURgecycle  ==> 00 , 30          0-59
RESyncmember ==> Yes            Yes | No
REUselimit  ==> 01000           0-10000
Signid      ==>
STANdbymode ==> Reconnect       Reconnect | Connect | Noconnect
+ STATsqueue ==> CDB2

                                SYSID=TCA3 APPLID=TCICSA3

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Threads are required to perform an DB2 request (SQL or command)

## DB2 Resource Definitions

### Define DB2CONN:

```
OVERTYPE TO MODIFY                                CICS RELEASE = 0700
CEDA ALTER DB2Conn( DC1A )
+ TCblimit ==> 0008                                4-2000
  THREADError ==> N906D                            N906D | N906 | Abend
POOL THREAD ATTRIBUTES
ACcountrec ==> None                               None | TXid | TAsk | Uow
AUTHId ==>
AUTHType ==> Userid                               Userid | Opid | Group | Sign | TErrm
                                                | TX
DRollback ==> Yes                                 Yes | No
PLAN ==>
PLANExitname ==> DSNCUEXT
Priority ==> Equal                                High | Equal | Low
THREADLimit ==> 0003                              3-2000
THREADWait ==> Yes                                Yes | No
COMMAND THREAD ATTRIBUTES
COMAUTHId ==>
COMAUTHType ==> Userid                            Userid | Opid | Group | Sign | TErrm
                                                | TX
+
                                                SYSID=TCA3 APPLID=TCICSA3

PF 1 HELP 2 COM 3 END                            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Threads are required to perform an DB2 request (SQL or command)

## DB2 Resource Definitions

### Define DB2CONN:

```
OVERTYPE TO MODIFY                                CICS RELEASE = 0700
CEDA ALTER DB2Conn( DC1A      )
+ COMThreadlim ==> 0001                          0-2000
DEFINITION SIGNATURE
DEFinetime   : 10/20/17 09:05:47
CHANGTime    : 10/20/17 09:05:47
CHANGEUsrid  : CICSUSER
CHANGEAGEnt  : CSDApi          CSDApi | CSDBatch
CHANGEAGRel  : 0700
```

SYSID=TCA3 APPLID=TCICSA3

PF 1 HELP 2 COM 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

Threads are required to perform an DB2 request (SQL or command)

## DB2 Resource Definitions

### Define DB2ENTRY:

```
OVERTYPE TO MODIFY OR PRESS ENTER TO EXECUTE          CICS RELEASE = 0700
CEDA Alter DB2Entry( THZENT1 )
DB2Entry       : THZENT1
Group          : TESTGRP
Description    ==> ENTRY FOR THEMIS TEST TRANSACTION
THREAD SELECTION ATTRIBUTES
TRansid        ==> THZ*
THREAD OPERATION ATTRIBUTES
ACcountrec     ==> None           None | TXid | TAsk | Uow
AUTHid         ==>
AUTHType       ==> Userid        Userid | Opid | Group | Sign | TErr
                                     | TX
DRollback      ==> Yes           Yes | No
PLAN           ==>
PLANExitname   ==> T
PRiority       ==> Equal        High | Equal | Low
PROtectnum     ==> 0001         0-2000
THREADLimit    ==> 0003         0-2000
+ THREADWait   ==> Yes         Pool | Yes | No

                               SYSID=TCA3 APPLID=TCICSA3

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Threads are required to perform an DB2 request (SQL or command)

## DB2 Resource Definitions

### Define DB2ENTRY:

```
OVERTYPE TO MODIFY OR PRESS ENTER TO EXECUTE          CICS RELEASE = 0700
CEDA ALTER DB2Entry( THZENT1 )
+ DEFINITION SIGNATURE
  DEFInetime      : 10/20/17 09:07:53
  CHANGETime     : 10/20/17 09:07:53
  CHANGEUsrid    : CICSUSER
  CHANGEAGent    : CSDApi          CSDApi | CSDBatch
  CHANGEAGRel    : 0700
```

SYSID=TCA3 APPLID=TCICSA3

PF 1 HELP 2 COM 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

Threads are required to perform an DB2 request (SQL or command)

## DB2 Resource Definitions

### Define DB2TRAN:

```
OVERTYPE TO MODIFY                                CICS RELEASE = 0700
CEDA Alter DB2Tran( THZTRAN )
DB2Tran      : THZTRAN
Group       : TESTGRP
Description  ==> ADDITIONAL TRANS IDS IN GROUP
Entry       ==> THZENT1
Transid     ==> +THM
DEFINITION SIGNATURE
DEFinetime  : 10/20/17 09:08:58
CHANGTime   : 10/20/17 09:08:58
CHANGEUserid : CICSUSER
CHANGEAGent : CSDApi          CSDApi | CSDBatch
CHANGEAGrel  : 0700
```

SYSID=TCA3 APPLID=TCICSA3

PF 1 HELP 2 COM 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

Threads are required to perform an DB2 request (SQL or command)