

ZCICSDRF.TXT

zCICS Diagnosis Reference

=====

Index

Control Blocks...Client

- Dynamic Storage Area (DSA)
- EXEC Interface Block (EIB)
- HANDLE ABEND Block
- HANDLE AID Block
- HANDLE CONDITION Block
- Link-Level Area (LKA)
- Terminal Control Table Terminal Entry (TCTTE)
- Temporary Storage request/reply block (DFHTSBLK)
- File Control request/reply block (DFHFCBLK)
- Interval Control request/reply block (DFHICBLK)
- Task Control request/reply block (DFHKCBLK)
- CWA request/reply block (DFHCWBLK)

Control Blocks...Server

- File Control Table (DFHFCT)
- VSAM Work Area (DFHVSAD)
- Interval Control Element (DFHICEDS)
- Temporary Storage Name Table (DFHTSNDS)
- Thread control (THRDDSCT)
- Queue Element Area (DFHQEADS)

Management descriptions...

- Abend
- COMMAREA
- GETMAIN/FREEMAIN
- HANDLE AID
- HANDLE/IGNORE CONDITION
- Temporary Storage
- File Control
- Interval Control
- ENQ/DEQ
- CWA
- Macros, LCLhhh modules and Z390LCL...the linkage mechanisms.
- Sequential Terminal support

LCL submodules for EXEC CICS functions

Event Tracing

ZCICSDRF.TXT

Z390CICS

Operation
GBL submodules
Internal Abends

-
Control Blocks...Client

Dynamic Storage Area (DSA).

Eye catcher: None
Acquired : DFHEIENT
Released : DFHEIRET
Anchor : R13
DSECT : DFHEISTG (prefix only)
Cleared : Only the prefix, not the user area
Length : Variable

EXEC Interface Block (EIB).

Eye catcher: 'DFHEIBLK'
Acquired : Z390KCP
Released : Close of thread
Anchor : R11 (DFHEIBR)
DSECT : DFHEIBLK
Cleared : Yes
Length : EIBLENG

HANDLE ABEND Block.

Eye catcher: 'DFHABBLK'
Acquired : 1st use
Released : Task end
Anchor : TCTTEABD (Byte after eyecatcher)
DSECT : DFHABBLK
Cleared : Yes
Length : ABDLENG (one table entry)
Entries : 25

HANDLE AID block.

Eye catcher: 'DFHADBLK'
Acquired : HANDLE AID (1st block only)
 PUSH HANDLE
Released : DFHEIRET (all chained AID blocks)
 POP HANDLE (only top-of-chain block)

ZCICSDRF.TXT

XCTL (all chained AID blocks)
Anchor : DFHEIAID (DSA)
Chain : AIDCHAIN
DSECT : DFHADBLK
Cleared : Yes
Length : AIDLENG

HANDLE CONDITION block.

Eye catcher: 'DFHHCBLK'
Acquired : HANDLE CONDITION (1st block only)
PUSH HANDLE
Released : DFHEIRET (all chained HANDLE CONDITION blocks)
POP HANDLE (only top-of-chain block)
XCTL (all chained HANDLE CONDITION blocks)
Anchor : DFHEIHCN (DSA)
Chain : HCNCHAIN
DSECT : DFHHCBLK
Cleared : Yes
Length : HCNLENG

Link-Level Area (LKA).

Eye catcher: None
Acquired : Z390KCP
Released : Close of thread
Anchor : TCTTELKA
Cleared : Yes
Length : 4 (R13 value for this link-level)
Entries : 25

Terminal Control Table Terminal Entry (TCTTE).

Eye catcher: 'DFHTCTTE'
Acquired : Z390KCP
Released : Close of thread
Anchor : R10 (TCTTEAR)
DSECT : DFHTCTTE
Cleared : Yes
Length : TCTTELEN

Temporary Storage request/reply block (DFHTSBLK).

Data sent/received follows the block.

Eye catcher: None
Acquired : GETMAINd by EXEC CICS TS command
Released : FREEMAINd by EXEC CICS TS command
Anchor : None
DSECT : DFHTSBLK

ZCICSDRF.TXT

Cleared : No
Length : TSPREFIX

File Control request/reply block (DFHFCBLK).
Data sent/received follows the block.

Eye catcher: None
Acquired : GETMAINd by an EXEC CICS file control command
Released : FREEMAINd by an EXEC CICS file control command
Anchor : None
DSECT : DFHFCBLK
Cleared : Yes
Length : FCPREFIX

Interval Control request/reply block (DFHICBLK).

Eye catcher: None
Acquired : GETMAINd by an EXEC CICS START/CANCEL command
Released : At task end
Anchor : None
DSECT : DFHICBLK
Cleared : Yes
Length : ICPREFIX

Task Control request/reply block (DFHKCBLK).

Eye catcher: None
Acquired : GETMAINd by an EXEC CICS ENQ/DEQ command
Released : At task end
Anchor : None
DSECT : DFHKCBLK
Cleared : Yes
Length : KCPREFIX

CWA request/reply block (DFHCWBLK).

Eye catcher: None
Acquired : Embedded in LCL0202
Released : N/A
Anchor : None
DSECT : DFHCWBLK
Cleared : No
Length : CWPREFIX

Control Blocks...Server

File Control Table (DFHFCT)
Contains the ACB for each file operation.

ZCICSDRF.TXT

Eye catcher: None
Acquired : Z390CICS at start
Released : Never
Anchor : DFHFCTAD
DSECT : DFHFCTDS
Length : FCTABLEN

VSAM Work Area (DFHVSAD)

Also contains the RPL for file operations.

Eye catcher: None
Acquired : Z390CICS at File Control process start
Released : Z390CICS at File Control process end or task abend
Anchor : FCTVSWA
Chain : VSWCHAIN
DSECT : DFHVSAD
Cleared : Yes
Length : VSWLEN

Interval Control Element (DFHICEDS)

Chained in time order

Eye catcher: None
Acquired : GBL1008 when a valid EXEC CICS START is processed
Released : GBL100C when a valid EXEC CICS CANCEL is processed
 : GBL10FF when an ICE scan can start a task
Anchor : ICEANCHR
Chain : ICECHAIN
DSECT : DFHICEDS
Cleared : Yes
Length : ICELEN

Temporary Storage Name Table (DFHTSNDS)

Eye catcher: None
Acquired : GBL0A02 when first EXEC CICS WRITEQ TS is processed
Released : GBL0A06 when EXEC CICS DELETEQ TS is processed
Anchor : TSNANCHR
Chain : TSNCHAIN
DSECT : DFHTSNDS
Cleared : Yes
Length : TSNLEN

Thread Control (THRDDSCT)

One entry for each terminal to a maximum of 10.

When SEQ_TERM=YES, the special 11th terminal is reserved for SQ01.

Eye catcher: None
Acquired : Fixed in Z390CICS

ZCICSDRF.TXT

Released : Never
Anchor : THRDCNTL
DSECT : THRDDSCT
Cleared : Yes
Length : THRDLEN

Queue Element Area (DFHQEADS)

Eye catcher: None
Acquired : GBL1204 when EXEC CICS ENQ is processed
Released : GBL1206 when EXEC CICS DEQ is processed
 : GBL12FC when task end DEQALL is processed
Anchor : QEANCHR
Chain : QEACHNF
DSECT : DFHQEADS
Cleared : Yes
Length : QEALLEN

-
Abend Management

The following types of abend may occur...

Program check
IGNORE CONDITION not permitted (program check)
Condition raised but not handled or ignored
EXEC CICS ABEND

Whether any of these result in a dump and/or termination of the task depends on the HANDLE ABEND status which is discussed later.

All of the above types cause a program check, with the following markers...

X'000000',C'ABEND',A(IGNORE address) -- HANDLE CONDITION
X'0000FE',C'ABEND',C'xxxx' -- ABEND with dump
X'0000FF',C'ABEND',C'????' -- ABEND without dump
Other -- Program check

The program check is trapped by the ESTAE routine APPABEND in Z390KCP. This routine determines the cause of the abend and takes the correct action...

HANDLE CONDITION marker...

The last CONDITION block is located (if any) and the

ZCICSDRF.TXT

condition

slot is tested, followed by the ERROR slot.
If no HANDLE or IGNORE then the terminate handler is invoked
(see later).

ABEND with/without dump and program checks go straight to the
terminate handler.

Terminate handler...

APPTABDN in Z390KCP tests for the existence of a HANDLE ABEND
block (DFHABBLK, see above). If none the task is terminated
abnormally.

Each entry in the HANDLE ABEND block represents a link-level,
so the table is scanned backwards for the highest active
entry.

If there are no active entries, the task is terminated
abnormally.

If an active entry is found it is immediately inactivated.
HANDLE ABEND LABEL will cause a branch to the label.
HANDLE ABEND PROGRAM will cause an XCTL to the program.
If the abending program has received a COMMAREA, then that
will be passed to the abend handler.

If HANDLE ABEND LABEL is at a higher link-level than the
abending program, then a special EXEC CICS RETURN CLEANER is
issued (in Z390KCP) to simulate a RETURN, as all lower levels
are now considered abandoned.

Note that PUSH, POP, HANDLE ABEND CANCEL/RESET only affect
the current link-level.

Dumps...

A program check will always produce an ASRA SNAP dump.
ABEND without NODUMP, will always produce a dump using
ABCODE.

No other dumps will be produced if an abend or condition is
handled.

COMMAREA Management

a) RETURN COMMAREA

ZCICSDRF.TXT

The program issuing the RETURN must be at link-level 1 (ie. about to return to Z390KCP). If this is not the case then INVREQ will be raised. This condition cannot be IGNORED as it is assumed that no valid code follows a RETURN.

The RETURN macro sets TCTTECA (address) and TCTTECAL (length). When the next task is invoked TCTTECAL is used to refresh EIBCALEN.

When Z390KCP regains control after RETURN, the COMMAREA address and length are compared with the last RETURN COMMAREA (holding areas COMMADDR and COMMLLEN).

--- If both are the same, they are passed to the next transid.

--- If either is different, then a new area is GETMAINd, the COMMAREA is copied, and the old one FREEMAINd.

Temporary holding areas COMMSAVA and COMMSAVL are used

during

the FREEMAIN process.

b) LINK COMMAREA

The address is stored in the linkers DSA at DFHEICAP and the length in EIBCALEN. DFHEICAP is passed as a parameter.

Note: A LINK COMMAREA is never specifically FREEMAINd, it is always part of another storage area (DSA, Program,

GETMAIN).

c) XCTL COMMAREA

The current COMMAREA address in the DSA (DFHEICAP) and length in EIBCALEN are compared with the XCTL COMMAREA.

--- If both are the same, the address is passed to the next program.

--- If either is different, then a new area is GETMAINd and the COMMAREA is copied.

The new COMMAREA address is held in the callers DSA (DFHEICAP)

and is passed via a small GETMAINd area. This area address is

stored at DFHEIPRM and its existence flagged by TCTTECND=X'FF'.

See Interval Control Management for further documentation which may

ZCICSDRF.TXT

affect COMMAREA processing.

GETMAIN/FREEMAIN Management

A chain of storage areas is anchored from TCTTESCC.
TCTTESCC has the address of the first GETMAINd area.

Eight bytes are added to each request, and they serve as a prefix...

4-byte address of next GETMAIN or 0, 4-byte total length.

The user is passed the address after the prefix.

FREEMAIN must have the same address as the GETMAIN passed otherwise an INVREQ condition is raised. FREEMAINS may occur in any order, the chain is just 'repaired' at that point.

At task end or task abend all remaining GETMAINS are FREEMAINd.

HANDLE AID Management

A HANDLE AID is owned by a program and is never passed to another program. The AID block (DFHADBLK, see above) is acquired on first use. PUSH and POP will acquire/release additional AID blocks.

At task end, task abend or an XCTL, all AID blocks are FREEMAINd.

HANDLE AID only works for conversational tasks.

HANDLE/IGNORE CONDITION Management

A HANDLE CONDITION is owned by a program and is never passed to another program. The CONDITION block (DFHHCBLK, see above) is acquired on first use. PUSH and POP will acquire/release additional CONDITION blocks.

Each 4-byte entry represents a condition, this may contain...

- 4X'00' -- The condition is not handled (default)
- A(label) -- The condition will be handled at label
- 4X'FF' -- The condition should be ignored

The ERROR condition can be handled as a 'catch-all' for any type of condition that doesn't have a specific HANDLE CONDITION. When both a condition and ERROR are set, only the condition label is used, not both.

At task end, task abend or an XCTL, all CONDITION blocks are

ZCICSDRF.TXT

FREEMAINd.

Temporary Storage Management

The queues are owned by Z390CICS, so all requests for TS services are sent by zCICS tasks to the server.

There are two structures in the server:

The queue name chain ...

A chain of all queue names.

The anchor of the chain is internal (TSNANCHR).

The DSECT for the name table is internal (TSNAMES).

A queue name is created by the first WRITEQ TS for the name and is chained on the end.

DELETEQ TS will delete all the data items and then delete the queue name and repair the chain.

The TS data chain ...

A chain of all items added to the queue.

The anchor of the chain is in the queue name table (TSNITEM1).

The DSECT for the TS data chain prefix is internal (TSDPREFIX),

the data follows the prefix.

WRITEQ TS will add a new item to the chain end.

WRITEQ TS REWRITE will free the old item, create a new one, and repair the chain.

DELETEQ TS will delete all the data items and then delete the queue name and repair the chain.

File Control Management

The files are owned by Z390CICS, so all requests for FC services are sent by zCICS tasks to the server.

The FCT (DSECT DFHFCTDS) defines the status of each file.

Each FCTTE contains the ACB for that file.

See the zCICS VSAM Guide to see how files are created and defined to zCICS.

a) File opening

When Z390CICS starts, all files defined as FILSTAT=OPENED are opened.

ZCICSDRF.TXT

Failure results in the status (CLOSED,DISABLED).

Files defined as (CLOSED,ENABLED) are opened when the first request is received.

Failure results in the status (CLOSED,DISABLED).

b) Request processing

No error conditions are explained here, they are listed in the zCICS Application Programming Guide, and in the IBM Manuals.

Any VSAM feedback codes and errors are converted to RESP/RESP2 values and sent back to the Client.

When a task ends or abends all VSWAs owned by the task are released. An exception to this occurs when a condition is raised after a browse command is issued and there is a HANDLE CONDITION.

In this case, the VSWA is not released unless the transaction is abended and will require an ENDBR.

A READ will always release the VSWA regardless of HANDLE CONDITION.

i) READ (ESDS)

A VSWA is acquired.

RPL OPTCD is set to (ADR) or (ADR,XRBA).

RPLARG is set to the address of FCP(X)RBA.

Area of the maximum or fixed length is GETMAINd and RPLAREA is set. GET issued and the data is sent to the program.

The VSWA is released.

READ (RRDS)

A VSWA is acquired.

RPL OPTCD is set to (KEY).

RPLARG is set to the address of FCPRRN.

Area of the maximum or fixed length is GETMAINd and RPLAREA is set. GET issued and the data is sent to the program.

The VSWA is released.

READ (KSDS)

A VSWA is acquired.

RPL OPTCD is set to (KEY,FKS,KEQ).

Options KGE and/or GEN are also set if specified.

KEYLENGTH(0) is a special case and forces GEN and KGE.

ZCICSDRF.TXT

RPLARG is set to the address of FCPRID.
Area of the maximum or fixed length is GETMAINd and RPLAREA is set.

If GEN or KGE is specified, then POINT is issued.
GET issued and the data is sent to the program.
The VSWA is released.

ii) STARTBR (ESDS)

A VSWA is acquired.
The REQID is set (default is zero).
RPL OPTCD is set to (ADR,SEQ) or (ADR,SEQ,XRBA).
RPLARG is set to the address of FCP(X)RBA.
A POINT is issued.
The current XRBA is saved in the VSWA.

STARTBR (RRDS)

A VSWA is acquired.
The REQID is set (default is zero).
RPL OPTCD is set to (KEY,SEQ).
RPLARG is set to the address of FCPRRN.
A POINT is issued.
The current RRN is saved in the VSWA.

STARTBR (KSDS)

A VSWA is acquired.
The REQID is set (default is zero).
RPL OPTCD is set to (KEY,SEQ,FKS,KEQ).
Options KGE and/or GEN are also set if specified.
KEYLENGTH(0) is a special case and forces GEN and KGE.
RPLARG is set to the address of FCPRID.
A POINT is issued.
The current (generic) FCPRID is saved in the VSWA.
The current KEYLENGTH is saved in the VSWA.

iii) READNEXT (ESDS)

The VSWA created by the STARTBR is located.
RPL OPTCD is set to (ADR,SEQ,FWD) or (ADR,SEQ,FWD,XRBA).
RPLARG is set to the address of FCP(X)RBA.
A check is made to see if the XRBA supplied differs from the current XRBA, if it does then a POINT is issued. This allows skip-sequential processing to occur.
Area of the maximum or fixed length is GETMAINd and RPLAREA is set.
The current XRBA is saved in the VSWA, a GET is issued, and the current (X)RBA and the data are sent to the program.

ZCICSDRF.TXT

READNEXT (RRDS)

The VSWA created by the STARTBR is located.

RPL OPTCD is set to (KEY,SEQ,FWD).

RPLARG is set to the address of FCPRRN.

A check is made to see if the RRN supplied differs from the current RRN, if it does then a POINT is issued. This allows skip-sequential processing to occur.

Area of the maximum or fixed length is GETMAINd and RPLAREA is set.

The current RRN is saved in the VSWA, a GET is issued, and the current RRN and the data are sent to the program.

READNEXT (KSDS)

The VSWA created by the STARTBR is located.

RPL OPTCD is set to (KEY,SEQ,FKS,KEQ,FWD).

Options KGE and/or GEN are also set if specified in the STARTBR,

RPLARG is set to the address of FCPRID.

POINT is issued for skip-sequential:

If the keylength has changed and/or the (generic) key in FCPRID has changed.

Area of the maximum or fixed length is GETMAINd and RPLAREA is set.

The current (generic) FCPRID is saved in the VSWA, a GET is issued, and the current full key and the data are sent to the program.

iv) READPREV (ESDS)

The VSWA created by the STARTBR is located.

RPL OPTCD is set to (ADR,SEQ,BWD) or (ADR,SEQ,BWD,XRBA).

RPLARG is set to the address of FCP(X)RBA.

A check is made to see if the XRBA supplied differs from the current XRBA, if it does then a POINT is issued. This allows skip-sequential processing to occur.

Area of the maximum or fixed length is GETMAINd and RPLAREA is set.

The current XRBA is saved in the VSWA, a GET is issued, and the current (X)RBA and the data are sent to the program.

READPREV (RRDS)

The VSWA created by the STARTBR is located.

RPL OPTCD is set to (KEY,SEQ,BWD).

RPLARG is set to the address of FCPRRN.

A check is made to see if the RRN supplied differs from the

ZCICSDRF.TXT

current RRN, if it does then a POINT is issued. This allows skip-sequential processing to occur.

Area of the maximum or fixed length is GETMAINd and RPLAREA is set.

The current RRN is saved in the VSWA, a GET is issued, and the current RRN and the data are sent to the program.

READPREV (KSDS)

The VSWA created by the STARTBR is located.

RPL OPTCD is set to (KEY,SEQ,FKS,KEQ,BWD).

Note: GEN is invalid and KGE is ignored.

RPLARG is set to the address of FCPRID.

POINT is issued for skip-sequential if the keylength has changed.

Area of the maximum or fixed length is GETMAINd and RPLAREA is set.

The current FCPRID is saved in the VSWA, a GET is issued, and the current full key and the data are sent to the program.

v) RESETBR (ESDS)

The VSWA created by the STARTBR is located.

RPL OPTCD is set to (ADR,SEQ) or (ADR,SEQ,XRBA).

RPLARG is set to the address of FCP(X)RBA.

A POINT is issued.

The current XRBA is saved in the VSWA.

RESETBR (RRDS)

The VSWA created by the STARTBR is located.

RPL OPTCD is set to (KEY,SEQ).

RPLARG is set to the address of FCPRRN.

A POINT is issued.

The current RRN is saved in the VSWA.

RESETBR (KSDS)

The VSWA created by the STARTBR is located.

RPL OPTCD is set to (KEY,SEQ,FKS,KEQ).

Options KGE and/or GEN are also set if specified.

KEYLENGTH(0) is a special case and forces GEN and KGE.

RPLARG is set to the address of FCPRID.

A POINT is issued.

The current (generic) FCPRID is saved in the VSWA.

The current KEYLENGTH is saved in the VSWA.

vi) ENDBR (ESDS, RRDS, KSDS)

The VSWA created by the STARTBR is located.

ZCICSDRF.TXT

The VSWA is released.

Interval Control Management

- a) DELAY and ASKTIME are handled by the Client.
- b) START

The module LCL1008 validates the time parameters and also the TRANSID.

However the time is specified, it is converted to a STCK time that has been reduced to units of 0.01 seconds.

If REQID has not been specified, one is created by multiplying TRANSID and TERMID and then overlaying the first two bytes with C'DF'. If TERMID is omitted then blanks are assumed as zeros

would

produce a zero result. The REQID is then stored in EIBREQID.

If the START command contains any of the parameters FROM, QUEUE, RTRANSID or RTERMID then a TS Q will be built.

The TS Q record will have a 16-byte prefix to hold any of the parameters QUEUE, RTRANSID or RTERMID followed by the FROM data (if any). In the prefix, any parameters not specified will be X'00'.

The WRITEQ TS has two special parameters (internal use only), ICTRAN and ICTERM. These are the TRANSID and TERMID parameters from the START command. When the WRITEQ TS is shipped to the Server for processing, these parameters are stored in the TS name table. The REQID (TS Q name) must be owned by the START TRANSID/TERMID combination. Any violation of that rule will raise the IOERR condition. Although the WRITEQ TS detects the IOERR condition, it is passed on to the START command.

The ICTRAN/ICTERM parameters also serve another purpose, in that specifying them allows the writing of a DF-prefix TS record

which

would normally raise the INVREQ condition.

The START command parameters are then shipped to the Server.

Server processing...GBL1008

The TERMID is validated, if invalid the TERMIDERR condition is returned to the Client.

ZCICSDRF.TXT

An Interval Control Element (ICE) is then created and chained in expiration time order.

c) Invocation

This mechanism is unique to each terminal.

The IC scan mechanism is in Z390KCP.

On every attempt to receive data from a terminal a request is sent to the Server to do an ICE scan for an expired one.

Server processing...GBL10FF

Assuming an an expired ICE is found for our terminal, checks are made to see if the terminal is available.

Terminal not available...

If the ICE REQID and TRANSID are the same then this is a repeated START request which has written more records to the TS Q to be read by the initiated task. The ICE is deleted.

This situation will occur if multiple tasks issue START requests all with a very close expiry time.

Terminal available...

The Client is sent a zero return and will initiate the transaction and set EIBREQID. The ICE is then deleted.

An ICE rescan is then done, and if any expired ICEs match the REQID and TRANSID, they are deleted.

This situation will occur if one task issues multiple START requests all with a very close expiry time.

d) RETRIEVE

An IC invoked transaction is passed the REQID in EIBREQID. The module LCL100A issues a READQ TS QUEUE(EIBREQID) ...

If conditions ITEMERR or QIDERR occur these are converted to condition ENDDATA and this condition is raised.

Each TS Q record has a 16-byte prefix and the fields QUEUE, RTRANSID and RTERMID are moved into the labels provided.

If a field is requested but is X'00', then the ENVDEFERR

condition

is raised.

Note: If one of the three fields is supplied but not requested, then no error is raised.

ZCICSDRF.TXT

The relationship between SET/INTO and LENGTH is complex, rather than repeat the logic here, I refer the reader to the text following label PFXDUN in LCL100A.

INTO/SET will receive the data after the prefix.

Deletion of the TS Q...

The condition ENDDATA will attempt a DELETEQ TS, no error occurs if this fails.

A successful READQ TS will do a DELETEQ TS if it returns NUMITEMS=1.

The condition LENGERR will attempt a DELETEQ TS if NUMITEMS=1.

No error occurs if this fails.

If EIBREQID is not null at task end or abend, a DELETEQ TS is done. No error occurs if this fails.

Mysteries...

It is not known how the condition NOTFND can be raised.

According to Hursley, it can't be raised.

An RCF has been raised.

e) CANCEL

The module LCL100C sends the REQID to the server.

The server module GBL100C scans the ICE chain for a matching REQID. If found the ICE is deleted and a rescan is done.

If no ICEs are deleted then the NOTFND condition is raised.

Any TS Q records associated with the deleted ICEs are not deleted.

f) IC tasks initiated while a pseudo-conversational task is in progress.

Note: It is unwise to change the screen in a way that would affect

a mapping operation. eg. don't add any unprotected fields
or
overlay any existing attributes.

ZCICSDRF.TXT

- i) IC tasks that end with EXEC CICS RETURN

eg. message broadcasters

The pseudo-conversational task should continue to operate correctly.

- ii) IC tasks that end with EXEC CICS RETURN TRANSID()

The original pseudo-conversational task will no longer operate. Any COMMAREA built by that task will be freed.

The next task initiated will be the TRANSID specified

- iii) IC tasks that end with EXEC CICS RETURN TRANSID() COMMAREA()

The original pseudo-conversational task will no longer operate. Any COMMAREA built by that task will be freed.

The next task initiated will be the TRANSID specified and

will

be passed the COMMAREA specified.

ENQ/DEQ Management

- a) ENQ

The module LCL1204 validates the length.

Although an omitted LENGTH is accepted, it may not work in zCICS. Please consult the author for advice.

The ENQ command parameters are then shipped to the Server.

Server processing...GBL1204

The QEA chain is scanned.

If a QEA is found with the same resource, length and originating termid, then the use count (QEAUCT) is

incremented.

If no QEA is found with the same resource and length, then a new QEA is built and chained.

If a QEA is found with the same resource and length but a different originating termid, then a potential suspend has occurred.

Please refer to the matrix in GBL1204 at label KCENQBSY for the complete logic of the ENQBUSY condition.

The Manual omits one description, when HANDLE CONDITION (ENQBUSY) is not specified, NOSUSPEND is specified and there is no NOHANDLE, then NOHANDLE must be forced as the task can never abend ENQBUSY (there is no abend code).

ZCICSDRF.TXT

When a task is suspended the THRDCNTL entry for this terminal is marked SUS-ENQ and the QEA is flagged with the termid suffix (QEAWAIT).

A task is suspended because we don't respond to the shipped ENQ request, this is done by the DEQ process.

b) DEQ

The module LCL1206 validates the length.

The DEQ command parameters are then shipped to the Server.

Server processing...GBL1206

The QEA chain is scanned.

If a QEA is found with the same resource, length and originating termid...

The use count is decremented if not zero.

Ok response is returned.

If the use count is zero, and no tasks are waiting for

this

resource, then the QEA is released and the chain repaired. Ok response is returned.

If the use count is zero, and tasks are waiting for this resource, then ok response is sent to each waiting task.

The terminal status is reset to RUNNING.

Then the QEA is released and the chain repaired.

Ok response is returned.

c) Task end or abend

Z390KCP will invoke a special DEQALL server request.

Server processing...GBL12FC

The QEA chain is scanned.

If a QEA is found with the same originating termid...

If no tasks are waiting for this resource, then the QEA is released and the chain repaired.

Ok response is returned.

If tasks are waiting for this resource, then ok response

ZCICSDRF.TXT

is

sent to each waiting task.
The terminal status is reset to RUNNING.
Then the QEA is released and the chain repaired.
Ok response is returned.

CWA Management

The CWA is considered a Global Resource, and is therefore owned and handled by the Global Manager, Z390CICS.

CWA size is determined by the Z390CICS.INI parameter CWASIZE= and the original 3.5K limit has been extended to 9,999,999 bytes.

The only access to the CWA is via EXEC CICS ADDRESS CWA(). This obtains a copy of the CWA and ENQ is used to lock any further CWA access for the duration of the task. See later for exceptions.

When the task ends or abends the CWA is sent back to the Global Manager for a refresh, and the natural clean-up process will ensure that the local copy is freed and DEQ takes place. There is no programmable access to the refresh/unlock mechanism.

Exceptions...

EXEC CICS RECEIVE (conversational) and EXEC CICS DELAY are considered to be long running and therefore each of those processes have a refresh/unlock and re-acquire mechanism wrapped around them.

Macros, LCLhxxx modules and Z390LCL...the linkage mechanisms

This is the operation at the time of shipping.

Experimental work is being done on CEDF and this will require a radical reworking of this mechanism.

Before the terminal is created, the module Z390LCL is loaded and the address placed in TCTTELCL.

I am going to use EXEC CICS READ as an example.

When this macro is issued the parameter list is set up and these instructions are issued:

L	R15,TCTTELCL	R15=LCL MODULE INDEXER
LARL	R1,=A(P0602)	R1=LCL MODULE PARAMETER LIST
BAKR	0,R15	STACK REGS AND GO

ZCICSDRF.TXT

Z390LCL has almost all (see later for exceptions) the macro processors linked in. The processor is located and a direct branch, BALR R14,R15 is used to invoke it.

Each LCLhhhh module has standard SUBENTRY/SUBEXIT linkage.

An LCLhhhh module can itself issue an EXEC CICS command, even though it's not really a CICS program. This will just create another stack entry.

When a processor completes, the SUBEXIT will return to Z390LCL which

issues a PR. This will unstack (only GR2-GR14) and return to the invoking macro for error processing or return to the invoking application.

Exceptions...

There are three processes that don't return immediately to the invoking application after completion...LINK, XCTL and RETURN.

These LCL modules are not part of Z390LCL and are linked in to each application that uses them.

When these macros are issued the parameter list is set up and these instructions are issued:

```
LARL R1,P0E08          R1=PARAMETER LIST
LARL R14,P0E08RTN_&SYSNDX RETURN ADDRESS
LRL  R15,=V(LCL0E08)   R15=EXTERNAL RETURN MODULE ADDRESS
BR   R15              GO TO IT
P0E08RTN_&SYSNDX EQU *
```

The processor decides where to go next.

Sequential Terminal support

Please read the Doc for this feature in zCICS Sequential Terminal Support. This section does not cover the batch programs Z390SEQ or Z390COMP.

When the INI parameter SEQ_TERM=YES is specified then Z390CICS will start a special CMDPROC terminal with a terminal id. of SQ01. This will reduce the total terminals that may be started to nine.

When Z390KCP is invoked for SQ01, two QSAM files are opened, one to process the input streams, and the other to write the output streams

ZCICSDRF.TXT

to SEQ00001. The TCTTE contains the DCB addresses and other supporting fields.

The internal EXEC CICS RECEIVE in Z390KCP and any in user programs read the next data stream from the input QSAM file. The internal EXEC CICS SEND in Z390KCP and any in user programs writes all data streams to the output QSAM file and displays them on the SQ01 terminal.

When an input file reaches the end, the file number is incremented and the file is closed and re-opened.

When all of the input streams are exhausted, the terminal is closed via an emulated CEMT S TER OUT unless the last input data stream was CEMT P SHU (recommended), in which case zCICS is shut down.

-

Event Tracing

Tracing is currently limited to those events that the server knows about. These events appear on the log as WTO messages.

The level of tracing is controlled by the INI parm TRACE_Z390CICS=

There is an intent to provide full application tracing, but each event would have to be sent to the server and may be too great an overhead in this environment.

Submodules for EXEC CICS processing

-FN-	Process	
LCL0202	ADDRESS	Embedded EXEC CICS
LCL0204	HANDLE CONDITION	
LCL0206	HANDLE AID	
LCL020A	IGNORE CONDITION	
LCL020C	PUSH HANDLE	
LCL020E	POP HANDLE	
LCL0402	RECEIVE	Embedded EXEC CICS
LCL0404	SEND	
LCL0602	READ	Embedded EXEC CICS
LCL060C	STARTBR	Embedded EXEC CICS

ZCICSDRF.TXT

LCL060E	READNEXT	Embedded EXEC CICS
LCL0610	READPREV	Embedded EXEC CICS
LCL0612	ENDBR	Embedded EXEC CICS
LCL0614	RESETBR	Embedded EXEC CICS
LCL0A02	WRITEQ TS	Embedded EXEC CICS
LCL0A04	READQ TS	Embedded EXEC CICS
LCL0A06	DELETEQ TS	Embedded EXEC CICS
LCL0C02	GETMAIN	
LCL0C04	FREEMAIN	
LCL0E02	LINK	Direct linkage
LCL0E04	XCTL	Direct linkage
LCL0E06	LOAD	
LCL0E08	RETURN	Direct linkage
LCL0E0A	RELEASE	
LCL0E0C	ABEND	
LCL0E0E	HANDLE ABEND	
LCL1204	ENQ	
LCL1206	DEQ	
LCL1002	ASKTIME	
LCL1004	DELAY	Embedded EXEC CICS
LCL1008	START	Embedded EXEC CICS
LCL100A	RETRIEVE	Embedded EXEC CICS
LCL100C	CANCEL	
LCL1802	RECEIVE MAP	
LCL1804	SEND MAP	
LCL1812	SEND CONTROL	
LCL1C02	DUMP	
LCL4A02	ASKTIME ABSTIME	
LCL4A04	FORMATTIME	

-
Z390CICS

Operation

- a) The file Z390CICS.INI is opened and the parameters analysed and used to set fixed fields in the program.

ZCICSDRF.TXT

Note: CEMT I SYS can be used to display them.

- b) A Command Prompt is started with the correct directory and parameters for each local terminal requested and Z390KCP is invoked in each one.
- c) A 32K receive area is acquired
- d) The Server port is opened.
- e) The FCT is loaded and any files eligible to be opened immediately are opened.
- f) At READLOOP a TCPIO RECEIVE is done (the process is slightly different when in shutdown mode (see later)).

Z390CICS will wait here for a request sent by any client. When a request is received the identity of the Client is return in GR2. Occasionally more than one request is received at the same time (batched), these are identified and the messages are split and individually processed.

REQTABLE is then scanned for the requested process and the routine is invoked.

CALL
Some routines are handled within Z390CICS and others will
a GBL submodule (listed below).

return
After the process is complete a TCPIO SEND is done with
codes and data (if requested) to the Client and the program
returns to READLOOP.

- g) Shutdown processing
After a Client has issued CEMT P SHU (IMM) a flag (SHUTIND) is set. The alternate code at READLOOP is then invoked which does a TCPIO RECEIVE,NOWAIT and checks every second if all Clients have closed.

Then all open VSAM files are closed and the Server (Z390CICS) shuts down.

Submodules

-FN-	Description	Program-
------	-------------	----------

ZCICSDRF.TXT

GBL0602	READ	Any
GBL060C	STARTBR	Any
GBL060E	READNEXT	Any
GBL0610	READPREV	Any
GBL0612	ENDBR	Any
GBL0614	RESETBR	Any
GBL0A02	WRITEQ TS	Any
GBL0A04	READQ TS	Any
GBL0A06	DELETEQ TS	Any
GBL0AFF	CEBR Request Qnames	Z390CEBR
GBL1008	START	Any
GBL100C	CANCEL	Any
GBL10FF	ICE SCAN	Z390KCP
GBL1204	ENQ	Any
GBL1206	DEQ	Any
GBL12FC	DEQALL	Z390KCP
GBLFE00	CEMT I TERM	Z390CEMT
GBLFE01	CEMT I SYSTEM	Z390CEMT
GBLFE05	CEMT I FILE	Z390CEMT
GBLFE06	CEMT I ENQueue	Z390CEMT
GBLFE07	CWA GET/PUT	Any

Copy Books

FILEERTB VSAM Error Code Table Z390CICS

Internal Abends

These are mostly caused by programming errors or situations I had not anticipated. Please report all of these to Don.

555	Unknown request sent to Server
666	CMDPROC failed
777	TCPIO OPEN/CLOSE Server failed
778	TCPIO RECEIVE failed
780	TCPIO SEND failed
790	VSAM feedback code was not expected

Change Summary

February 21, 2009

Added LCL0202, GBLFE07

ZCICSDRF.TXT

Added DFHCWBLK and CWA Management
Added Macro and LCL processor Management

November 24, 2008

Added section on LCL submodules
Added control blocks:
 DFHICBLK, DFHICEDS, DFHTSNDS, DFHKCBLK, THRDDSCT, DFHQEADS
Added Interval Control Management
Added ENQ/DEQ Management
Completed Z390CICS operation
GBLFE06 added (CEMT I ENQ)

June 27, 2008

Z390CICS operation extensively expanded

January 18, 2008

Extensive updates to File Control

Trademarks

IBM, CICS and VSAM are registered trademarks of International Business
Machines Corporation.

Author : Melvyn Maltz
Shipping Date: February 21, 2009
Z390 version : V1.5.00
zCICS version: V7

→