# z/TPF DEB5 Driver

## User's Guide

© Copyright IBM Corp. 2006

*This page intentionally left blank.*


© Copyright IBM Corp. 2006

# ZTEST DEB5

The DEB5 driver is used to build, maintain and modify a subsystem user unique database. It's purpose is to provide an application, similar to what a TPF customer would use, to simulate database activity. DEB5 is meant only as a test tool to drive system and I/O activity and test TPF functions. Among the functions that DEB5 can be used to test are VFA, Locking (both CF and DASD control unit), I/O macros, POOL functions, and database utilities like recoup and capture/restore.

The DEB5 package can be broken up functionally into a number of categories. First, the display programs display information about the size and configuration of the database and information about database activity. Second, the build and maintenance programs are used to initialize and build a database and to maintain its integrity. Last, the activity programs create activity by accessing and modifying the database -- adding, deleting, and updating records.

### How Does One Use DEB5?

Before one can start to run the DEB5 driver a database must be built. This is done by issuing a ZTEST DEB5 INIT followed by a ZTEST DEB5 BUILD command (a more detailed description of the commands and their use follows later in this user's guide). The INIT command initializes the fixed file records and the DEB5 keypoint. An INIT must be issued on a new system before the BUILD command can be issued. Parameters on the BUILD command determine the size and structure of the database. If there is an existing database then the BUILD command need not be issued (unless you wish to increase the size of the database).

DEB5 can be started only if the system is in NORM state and a database has already been built. Parameters on the START command tailor the amount and type of activity to create. Activity can be started to add, delete, and/or update records in the database.

DEB5 should be started in each subsystem user in which you want to drive database activity. If a database is not already built, DEB5 will *appear* to run, but in actuality the ECBs kicked off will merely exit.

A DEB5 session should begin by displaying information about the current database. It is a good idea, though not necessary, to issue the CHAIN, POOL, and STAT commands before you start DEB5 for the first time. These commands will verify that a database has been built, will tell you the size of the database, and check that there are no errors in the database. If errors exist these should be fixed before DEB5 is started. A DEB5 session should end by issuing the same display commands that the session was started with. This will record the activity created, the changes made to the database and verify that no errors were introduced into the database during the session. The display commands also should be issued periodically while DEB5 is running in order to monitor its activity.

### What does the database look like?

The DEB5 database is a simple tree structure, with the root of the tree being the DEB5 master segment. The first level of the tree structure is a layer of fixed file records referred to as Company level records. Chained below the company records are more fixed file records referred to as Origin (or level two) records. The next level are called Destination or level three records. Destination records are file pool records that are added or deleted to the database as activity is being run.

© Copyright IBM Corp. 2006

### Master Segments (QBB4, QBB5)

The DEB5 master segments define what face IDs are to be used in the DEB5 database for a given subsystem user. QBB4 and QBB5 contain a series of 16 byte entries that indicate which face IDs are defined to be company records and which are origin records. Whenever the database is accessed, the symbolic face ID of a company record is first read from the master segment.

### Fixed File Records

Company level records are the highest level records in the database. Each company record consists of a 96 byte header followed by a variable number of 48 byte company entries - as many as will physically fit in the company record. Each company entry contains a pointer to a unique origin record ordinal. The entry also contains the chained origin record's symbolic face ID and record ID. If the pointer to the origin level record is zero, the entry is considered to be a null company entry. A given company level record may have many entries, each chained to a different origin record. This provides the basis of the tree structured database.

Origin level records have a similar structure as company level records, with a 96 byte header followed by a variable number of 50 byte origin entries. As with company entries, each origin entry may contain a pointer to a destination level record. The destination level records are file pool records with their size and RIAT characteristics determined by the record ID of their origin record anchor. A null origin entry is defined as an origin entry with a zero destination level record pointer.

### File Pool Records

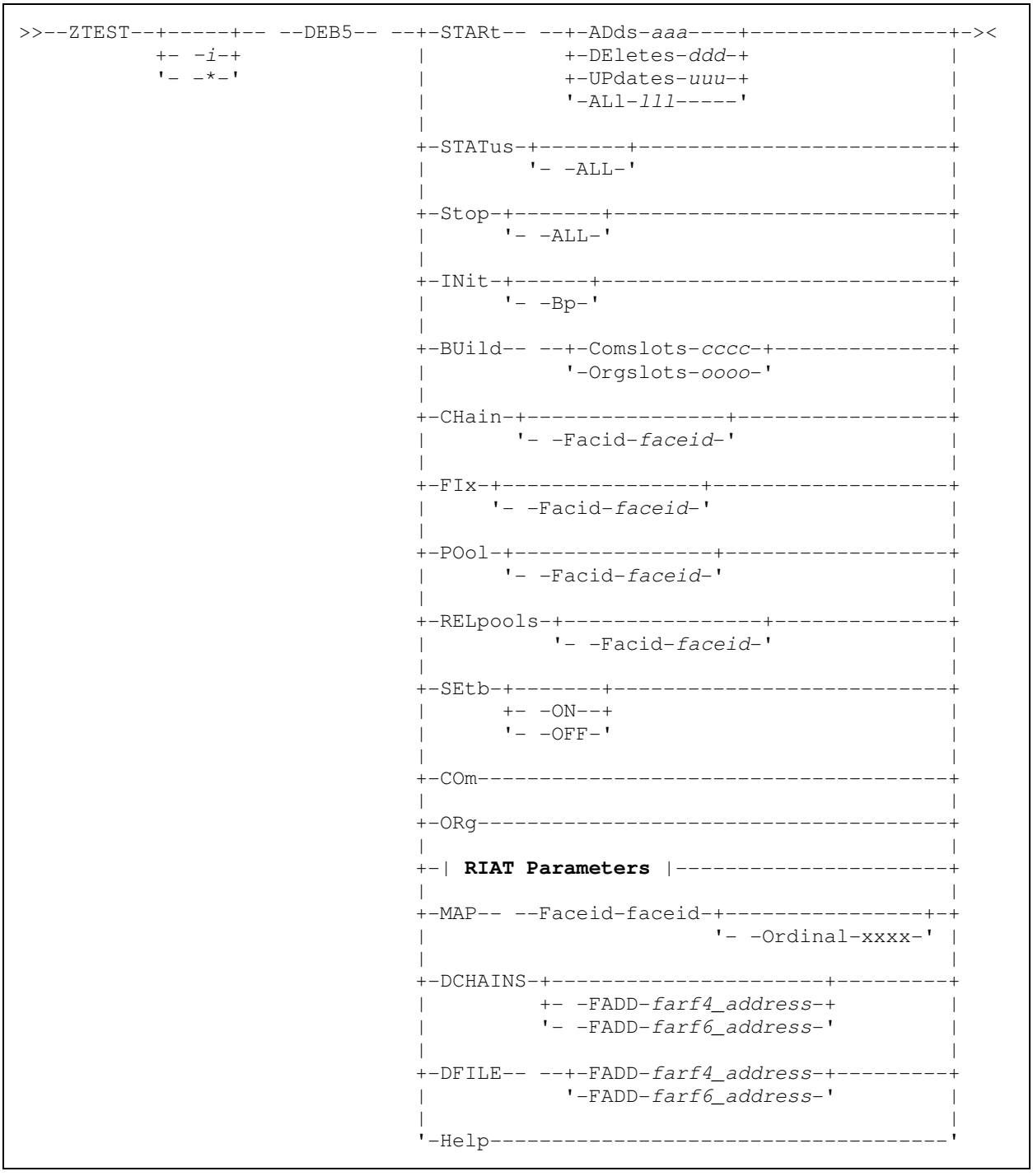
The destination level records are pool records that are chained from origin level entries. They too contain entries, but all destination level entries are null. Destination level records are added/deleted while database activity is being run (DEB5 is started). When a destination level entry is added to the database by an ADD ECB and there is no room in the current destination record for another entry, then a pool is requested and chained (using a pointer in the header) to the *full* record. It is in this manner that chains of pool records are formed. When an entry is deleted, a destination (pool) record can be similarly released. The chains of pool records attached to origin entries vary in length and are constantly changing while DEB5 activity is being run.

## Requirements and restrictions

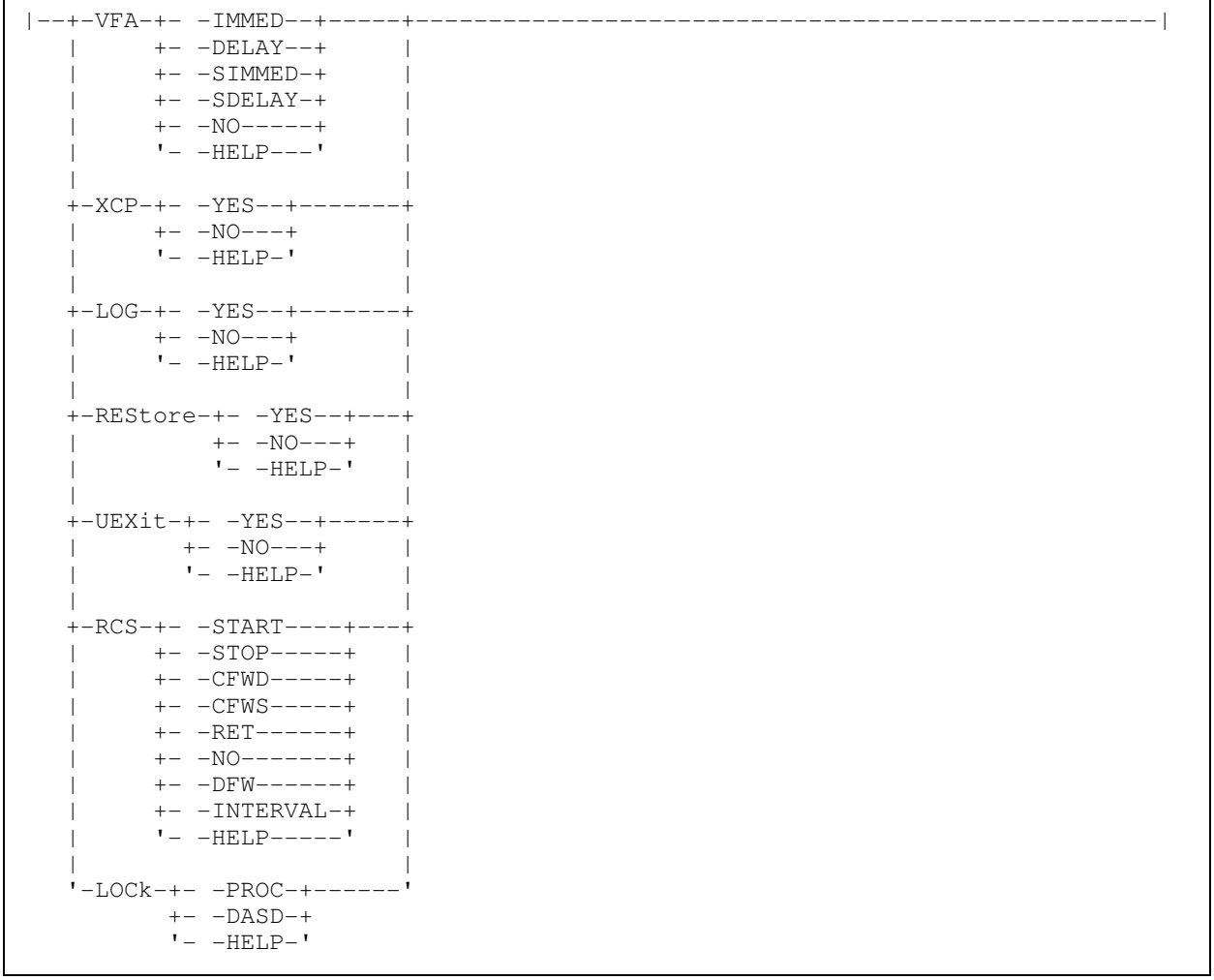
None.

 © Copyright IBM Corp. 2006

## Format


>>--ZTEST--+-----+-- --DEB5-- --+-STARt-- --+-ADds-aaa----+----------------+-><
+- -i-+ | +-DEletes-ddd-+ |
'- -*-' | +-UPdates-uuu-+ |
| '-ALl-lll-----' |
| |
+-STATus-+-------+-------------------------+
| '- -ALL-' |
| |
+-Stop-+-------+-------------------------+
| '- -ALL-' |
| |
+-INit-+------+-------------------------+
| '- -Bp-' |
| |
+-BUild-- --+-Comslots-cccc-+--------------+
| '-Orgslots-oooo-' |
| |
+-CHain-+----------------+-----------------+
| '- -Facid-faceid-' |
| |
+-FIx-+---------------+-------------------+
| '- -Facid-faceid-' |
| |
+-POol-+---------------+------------------+
| '- -Facid-faceid-' |
| |
+-RELpools-+---------------+--------------+
| '- -Facid-faceid-' |
| |
+-SEtb-+-------+-------------------------+
| +- -ON--+ |
| '- -OFF-' |
| |
+-COm-------------------------------------+
| |
+-ORg-------------------------------------+
| |
+-| RIAT Parameters |---------------------+
| |
+-MAP-- --Faceid-faceid-+---------------+-+
| '- -Ordinal-xxxx-' |
| |
+-DCHAINS-+---------------------+---------+
| +- -FADD-farf4_address-+ |
| '- -FADD-farf6_address-' |
| |
+-DFILE-- --+-FADD-farf4_address-+---------+
| '-FADD-farf6_address-' |
| |
'-Help-------------------------------------'

**RIAT Parameters:**




***i***

indicates the specific I-stream in which the driver will be run. If *i* is not specified, the test case(s) will be executed on the I-stream on which the command is entered.

*****

specifies the driver will be invoked on all currently defined and available I-streams.

**START**

initiates continuous database activity on the current subsystem user. It is also used to modify activity levels once the driver is active. It kicks off ADD, DELETE, and UPDATE activity on a periodic basis. The START command may only be issued in NORM state.

**ADDS-*aaa***

specifies the number of ECBs that are to perform ADD activity by appending one entry to a randomly chosen destination chain, where *aaa* is a value in the range of 0 to 200. Additional pool records are chained in as required.

 © Copyright IBM Corp. 2006

**DELETES-*ddd***
specifies the number of ECBs that are to perform DELETE activity by removing one entry from the end of a randomly chosen destination chain, where *ddd* is a value in the range of 0 to 200. Empty pool records are released to the system as required.

**UPDATES-*uuu***
specifies the number of ECBs that are to perform UPDATE activity by updating the timestamp area in the header of every record in a randomly chosen destination chain, where *uuu* is a value in the range of 0 to 200. No modifications to the chain are made.

**ALL-*lll***
specifies the number of ECBs that are to perform ADD, DELETE, and UPDATE activity where *lll* is a value in the range of 0 to 200. This option is a shorthand way of specifying the ADDS, DELETES, and UPDATES parameters substituting the *lll* value on each parameter. The ALL parameter may not be combined with any of the other START command parameters.

**STATUS**
displays information about DEB5 driver activity. It indicates whether DEB5 is active or inactive, the amount of activity that has been started, and DEB5s pool usage. The STATUS command is frequently used in conjunction with the POOL command as a method to verify database integrity.

The STATUS command returns counts of pool records requested and returned to the system for each record size. The total number of records for each size (gotten - released + lost) returned by the STATUS display should match the counts returned by the POOL command. Any discrepancy indicates a database problem and the FIX command should be issued to correct the problem.

The STATUS command may be issued in 1052 state or above. ECB counts are reset at IPL time; pool counts are reset only by the BUILD command.

The ALL option may be used to display "Active" and "Inactive" DEB5 driver status for all subsystem users.

**STOP**
stops database activity in the current subsystem user. The ALL option allows activity to automatically be stopped in all subsystem users.

**INIT**
initializes the DEB5 keypoint and database records. It will do an implied RELPOOLS command onto the database before initializing the database. This ensures that any pool records that may exist in the database are not lost until the next recoup or pool generation. If you want to bypass this implied command, then include the BP parameter, and it will proceed directly to initializing the database.

The INIT command should be issued before attempting to build a database using the BUILD command, or anytime you want to start from scratch. The INIT command may be issued in 1052 state or above; 1052 state is recommended.

© Copyright IBM Corp. 2006

**BUILD**
adds a specified number of COMPANY and/or ORIGIN entry pointers to the database. The maximum number of entries is limited only by the physical space available within the various COMPANY and/or ORIGIN records defined in the DEB5 master segments QBB4 and QBB5.

The BUILD command ultimately determines the shape and composition of the database, and largely controls its maximum size. The BUILD command may be issued in 1052 state or above; 1052 state is recommended.

**CHAIN**
displays counts of Company and Origin entries in the database. Totals for null entries are also displayed.

The FACID option may be used to selectively display Company and Origin entry counts for a specific Company record type.

**FIX**
corrects database chaining errors.

The FACEID option may be used to correct errors in a specific Company record and any Origin or Destination records chained from that Company record type.

**POOL**
displays counts of pool records subtotaled by record size.

The FACEID option may be used to selectively calculate pool counts by Company record type.

**RELPOOLS**
releases all pool records back to the system that are currently chained to the database.

The FACID option may be used to selectively release only pool records chained from a specific Company record type.

**SETB-ON|OFF**
enables and disables continuous Origin entry builds.

**COM**
displays information about the Company level records defined to the subsystem user issuing the command. Included in the display are the record ID, symbolic FACE ID, size, and processor shared/common attributes of each Company record type.

**ORG**
displays information about the Origin level records defined to the subsystem user issuing the command. Included in the display are the record ID, symbolic FACE ID, size, and processor shared/common attributes of each Origin record type.

 © Copyright IBM Corp. 2006

**VFA**
allows the user to modify RIAT attributes for the handling of record IDs used by DEB5. The VFA attributes for FILE and POOL records can be set to IMMED, DELAY, SIMMED, SDELAY, or NO.

**XCP**
allows the user to modify RIAT attributes for Exception Recording handling of record IDs used by DEB5. The XCP attribute can be set to either YES or NO.

**LOG**
allows the user to modify RIAT attributes for File Logging handling of record IDs used by DEB5. The LOG attribute can be set to either YES or NO.

**RESTORE**
allows the user to modify RIAT attributes for Restore handling of record IDs used by DEB5. The RESTORE attribute can be set to either YES or NO.

**UEXIT**
allows the user to modify RIAT attributes for User Exit handling of record IDs used by DEB5. The UEXIT attribute can be set to either YES or NO.

**RCS**
allows the user to modify RIAT attributes for Record Cache Subsystem handling of record IDs used by DEB5. The RCS attributes can be set to START, STOP, CFWD, CFWS, RET, NO, DFW, or INTERVAL.

**LOCK**
allows the user to modify RIAT attributes for designating the Record Lock manager for record IDs used by DEB5. The LOCK attributes can be set to either PROC or DASD.

**MAP**
displays information regarding the structure of the database. Currently only Company record types may be processed by the MAP command.

**FACID-*facid***
specifies the symbolic record type to be mapped. If the ORDINAL option is not specified, then summary information is displayed for each ordinal of the symbolic record type.

**ORDINAL-*xxxxx***
specifies a single ordinal within a symbolic record type for a detail display. Each entry within the record is displayed, listing chain information if present.

**DCHAINS**
allows the user to display the chain stucture of the database starting from any file address. Both FARF4 and FARF6 formats are entered with the FADD parameter. Not entering the FADD parameter or entering a file address of all zeros will begin the display of chains from the first COM FaceID ordinal 0 and will display the entire database.

© Copyright IBM Corp. 2006

**DFILE**
allows the user to display all the fields of a specified database record. Both FARF4 and FARF6 formats are entered with the FADD parameter.

**FADD-*farf4_address***
specifies an 8-character FARF4 address.

**FADD-*farf6_address***
specifies a 16-character FARF6 address.

**Help**
displays the correct syntax of the command.

## Source code information

The DEB5 driver consists of the following program segments:

**Header Files**

| Header File | Description |
|---|---|
| c_cmm.h | This header file defines the format of the DEB5 master list entries. |
| c_de0b4.h | This header file defines the format of the DEB5 keypoint record. |
| c_debeq.h | This header file defines equate values used by DEB5. |

**Macros**

| Macro | Description |
|---|---|
| CM0MM | Macro call for DEB5 master record. |
| CM1MM | Macro call for 06-XXXX. |
| CM2MM | Macro call for 06-XXXX. |
| CM3MM | Macro call for 06-XXXX. |
| DCTCOM | DSECT record for DEB5 COMPANY records. |
| DCTMSTR | Macro call for DEB5 master record. |
| DCTORG | ORG record. |
| DEBEQ | EBX fields for several DEB5 parameters. |
| DE0B4 | DSECTs the DEB5 keypoint record. |
| QWEQU | Common macro file. |
| sd0rv.mac | This macro file contains the DSECT for non-keypointable I-stream shared global (@ISSDRV)used by various drivers |

**BSOs**

| Module | Makefile | Segment | Description |
|---|---|---|---|
| BKZ0 | app_drvs.mak | bkz0.asm | GROUP macros for DEB5 records. |
| BKZ1 | app_drvs.mak | bkz1.asm | GROUP macros for DEB5 records. |
| BKZ2 | app_drvs.mak | bkz2.asm | GROUP macros for DEB5 records. |
| QBB4 | app_drvs.mak | qbb4.asm | DEB5 master segment part 1 |
| QBB5 | app_drvs.mak | qbb5.asm | DEB5 master segment part 2 |
| QDE0 | app_drvs.mak | qde0.asm | Command line parser |
| QDE1 | app_drvs.mak | qde1.asm | STATUS display generation |
| QDE2 | app_drvs.mak | qde2.asm | COM and ORG display generation |
| QDE3 | app_drvs.mak | qde3.asm | INIT processing |

 © Copyright IBM Corp. 2006

| Module | Makefile | Segment | Description |
|---|---|---|---|
| QDE4 | app_drvs.mak | qde4.asm | BUILD processing |
| QDE5 | app_drvs.mak | qde5.asm | POOL count display generation |
| QDE6 | app_drvs.mak | qde6.asm | CHAIN display generation |
| QDE8 | app_drvs.mak | qde8.asm | FIX processing |
| QDE9 | app_drvs.mak | qde9.asm | RELPOOLS processing |
| QDEA | app_drvs.mak | qdea.asm | START processing and activity generation |
| QDEB | app_drvs.mak | qdeb.asm | ADD processing |
| QDEC | app_drvs.mak | qdec.asm | DELETE processing |
| QDED | app_drvs.mak | qded.asm | UPDATE processing |
| QDEE | app_drvs.mak | qdee.asm | DEB5 utility functions |
| QDEF | app_drvs.mak | qdef.asm | RCS processing |
| QDEG | app_drvs.mak | qdeg.asm | RCS processing |
| QDEH | app_drvs.mak | qdeh.asm | RCS processing |
| QDEI | app_drvs.mak | qdei.asm | RCS processing |
| QDEJ | app_drvs.mak | qdej.asm | Loosely coupled configuration check |
| QDEK | app_drvs.mak | qdek.asm | Error message generation |
| QDEQ | app_drvs.mak | qdeq.asm | FARF 3/4/5 address comparison |
| QDER | app_drvs.mak | qder.asm | MAP processing - Company records |
| QDES | app_drvs.mak | qdes.asm | MAP processing - Origin records |
| QDET | app_drvs.mak | qdet.asm | DCHAINS display generation for COM and DEST records |
| QDEU | app_drvs.mak | qdeu.asm | DCHAINS display generation for ORIG records |
| QDEV | app_drvs.mak | qdev.asm | DFILE display generation for COM and ORG records |
| QDEW | app_drvs.mak | qdew.asm | DFILE display generation for DEST records |
| QRN1 | app_drvs.mak | qrn1.asm | Compute random number. |

**CSOs**

| Module | Makefile | Segment | Description |
|---|---|---|---|
| QDC0 | qdc0.mak | qdc0.c | To bring the DEB5 master segments into core, and read in the DEB5 keypoint. |

## Additional information

- Mapping of Destination records (DEB5 MAP POOL-) is not yet implemented.
- The CRETC (DEB5 CRETC) function is not yet implemented.

## Examples

The following command will initialize the DEB5 database after releasing any pool records that were chained to the database:

```
User:   ZTEST DEB5 INIT

System: QDE90000I 07.52.16 0 POOLS RELEASED FROM COMPANY: #QB1LN.+
        QDE90000I 07.52.16 0 POOLS RELEASED FROM COMPANY: #QB1LD.+
        QDE30001I  Fixed File record initialization starting+
        QDE30002I  Fixed file initialization complete, now initializing the keypoint+
        QDE30003I  DEB5 Initialization now complete+
```

© Copyright IBM Corp. 2006

 © Copyright IBM Corp. 2006

The following command will initialize the DEB5 database without releasing any pools records still chained to the database:

```
User:   ZTEST DEB5 INIT BP

System: QDE30001I  Fixed File record initialization starting+
        QDE30002I  Fixed file initialization complete, now initializing the keypoint+
        QDE30003I  DEB5 Initialization now complete+
```

The following command will initialize WP2's DEB5 database:

```
User:   WP2/ZTEST DEB5 INIT

System: CSMP0097I 10.51.52 CPU-B SS-WP   SSU-WP2  IS-04
        QDE30001I  Fixed File record initialization starting+
        CSMP0097I 10.51.52 CPU-B SS-WP   SSU-WP2  IS-04
        QDE30002I  Fixed file initialization complete, now initializing the keypoint+
        CSMP0097I 10.51.53 CPU-B SS-WP   SSU-WP2  IS-04
        QDE30003I  DEB5 Initialization now complete+
```

The following command will start DEB5 activity at 5 ADD ECBs and 2 DELETE ECBs per cycle:

```
User:   ZTEST DEB5 START ADD-5 DEL-2

System: DEA0002I  DEB5 started at the following levels:
        Adds:    5   Deletes:    2   Updates:    0.+
```

The following command will start DEB5 with ADDS-8, DELETES-8, and UPDATES-8:

```
User:   ZTEST DEB5 START ALL-8

System: QDEA0002I  DEB5 started at the following levels:
        Adds:    8   Deletes:    8   Updates:    8.+
```

The following command will stop DEB5 activity on all subsystem users in the system:

```
User:   ZTEST DEB5 STOP ALL

System: QDEA0007I  DEB5 is now beginning STOP processing.+
        QDEA0009W  Warning!  The STOP flag is being set with   24 ECBs
        still unaccounted for
        QDEA0008I  DEB5 is now STOPPED.+
```

## Messages

Below is a list of the DEB5 driver messages:

---

### ZTEST DEB5 INIT

```
QDE90000I 13.00.22 283 POOLS RELEASED FROM COMPANY: #QB24D.
QDE30001I  Fixed File record initialization starting
QDE30002I  Fixed file initialization complete, now initializing the keypoint
QDE30003I  DEB5 Initialization now complete
```

© Copyright IBM Corp. 2006

### ZTEST DEB5 INIT BP

```
QDE30001I  Fixed File record initialization starting
QDE30002I  Fixed file initialization complete, now initializing the keypoint
QDE30003I  DEB5 Initialization now complete
```

### ZTEST DEB5 BUILD COM-150 ORG-3000

**Note:** The following warning message is issued only when the number of COM slots requested is less than the number of COM slots available. The build process continues normally. The number of slots built can be displayed via the entry ZTEST DEB5 CHAIN.

```
QDE40002I 17.54.26 Warning: COM Build continues with less slots than requested.
QDE40002I 17.54.26 BUILD FOR LEVEL ONE RECS HAS COMPLETED.
QDE40003I 17.54.26 BUILD FOR LEVEL TWO RECS HAS COMPLETED.
```

### ZTEST DEB5 SETB ON

```
QDE40001I 17.54.26 CONTINUOUS BUILD FOR ORIGIN ENTRIES SET ON
```

### ZTEST DEB5 SETB OFF

```
QDE40001I 17.54.26 CONTINUOUS BUILD FOR ORIGIN ENTRIES SET OFF
```

### ZTEST DEB5 CHAIN

```
QDE60000I 17.54.26 COMPANY NAME IS: #QB24D
                   COMPANY ENTRIES:      150,  NULL COM ENTRIES:        2
                   ORIGIN  ENTRIES:     2359,  NULL ORG ENTRIES:     2359
```

### ZTEST DEB5 START ADD-8 DEL-2 UPDATE-4

```
QDEA0001E  You MUST be in NORM to run DEB5, START rejected.
```

**Explanation:** DEB5 database activity can only be started in NORM state

### ZTEST DEB5 START ADD-8 DEL-2 UPDATE-4

```
QDEA0002I  DEB5 started at the following levels:
Adds:    8   Deletes:    2   Updates:    4.+
```

### ZTEST DEB5 STOP

```
QDEA0007I  DEB5 is now beginning STOP processing.
QDEA0008I  DEB5 is now STOPPED.+
```

 © Copyright IBM Corp. 2006

## ZTEST DEB5 STOP

```
QDEA0006E  DEB5 is not running, STOP rejected.
```

**Explanation:** DEB5 database activity can only be stopped while it is running

## ZTEST DEB5 STATUS

```
QDE10002I  DEB5 is INACTIVE on processor B

Pool Usage Complex wide:
     Pool Size:            SM       LG       4K
     Pools Gotten:         42       53       39
     Pools Released:        1        0        0
     Pools Lost:            0        0        0

ECBs exited by type:      ADD      DEL      UPD
(Reset by IPL)            136       34       68

END OF DISPLAY
```

## ZTEST DEB5 STATUS

```
QDE10001I  DEB5 is ACTIVE on processor B

ECBs from last start:      ADDs:   8  DELs:   2  UPDs:   4

Pool Usage Complex wide:
     Pool Size:            SM       LG       4K
     Pools Gotten:         45       55       42
     Pools Released:        2        0        0
     Pools Lost:            0        0        0

ECBs exited by type:      ADD      DEL      UPD
(Reset by IPL)            144       36       72

END OF DISPLAY
```

## ZTEST DEB5 STAT ALL

```
QDE40002I 17.54.26 BUILD FOR LEVEL ONE RECS HAS COMPLETED.
QDE10003I  DEB5 status is as follows:
 SS     SSU      DRIVER
NAME    NAME     STATUS
BSS     HPN      ACTIVE
        HPN2     INACTIVE
WP      WP1      ACTIVE
        WP2      INACTIVE
        WP3      INACTIVE
END OF DISPLAY
```

## ZTEST DEB5 POOL

```
QDE50000I 12.50.11 COMP NAME: #QB24D   POOL COUNT:
                   SMALL:       91  LARGE:      121  4K:       71.
QDE50001I 12.50.11 CHAIN ERROR(S) ENCOUNTERED:        0.
```

© Copyright IBM Corp. 2006

### ZTEST DEB5 RELPOOLS

```
QDE90000I 13.00.22 283 POOLS RELEASED FROM COMPANY: #QB24D.
```

### ZTEST DEB5 COM

```
QDE20002I  COMPANY records defined for CPU  B / SSU HPN :
#QB24D   is CPU SHARED, has recid: A3, flags: 98, and is 4K-LD
```

### WP2/ZTEST DEB5 COM

```
QDE20002I  COMPANY records defined for CPU  B / SSU WP2 :
#QW1LD   is CPU SHARED, has recid: YA, flags: C8, and is SM-LD
#QW1LN   is CPU SHARED, has recid: YC, flags: C8, and is 4K-ST
```

### ZTEST DEB5 ORG

```
QDE20002I  ORIGIN  records defined for CPU  B / SSU HPN :
#QB1SN   is CPU SHARED, has recid: YA, flags: A0, and is SM-LD
#RB1SN   is CPU SHARED, has recid: YB, flags: A0, and is SM-LD
#QB1LD   is CPU SHARED, has recid: Y1, flags: C0, and is LG-LD
#RB1LD   is CPU SHARED, has recid: Y2, flags: C0, and is LG-LD
#QB1LN   is CPU SHARED, has recid: Y4, flags: C0, and is LG-LD
#RB1LN   is CPU SHARED, has recid: Y5, flags: C0, and is LG-LD
#QB1SD   is CPU SHARED, has recid: Y7, flags: A0, and is SM-LD
#RB1SD   is CPU SHARED, has recid: Y8, flags: A0, and is SM-LD
#QB14D   is CPU SHARED, has recid: A1, flags: 90, and is 4K-LD
#RB14D   is CPU SHARED, has recid: A2, flags: 90, and is 4K-LD
#RB24D   is CPU SHARED, has recid: A4, flags: 90, and is 4K-LD
#QB34D   is CPU SHARED, has recid: A5, flags: 90, and is 4K-LD
End of Display
```

### WP3/ZTEST DEB5 ORG

```
QDE20002I  ORIGIN  records defined for CPU  B / SSU WP3 :
#QW14D   is CPU SHARED, has recid: A1, flags: 90, and is 4K-LD
#RW14D   is CPU SHARED, has recid: A2, flags: 90, and is 4K-LD
#QW14N   is CPU SHARED, has recid: A3, flags: 90, and is 4K-LD
#RW14N   is CPU SHARED, has recid: A4, flags: 90, and is 4K-LD
#RW1LD   is CPU SHARED, has recid: YB, flags: C0, and is SM-LD
#RW1LN   is CPU SHARED, has recid: YD, flags: C0, and is 4K-ST
#QW1SD   is CPU SHARED, has recid: Y1, flags: A0, and is LG-LD
#RW1SD   is CPU SHARED, has recid: Y2, flags: A0, and is LG-LD
#QW1SN   is CPU SHARED, has recid: Y3, flags: A0, and is LG-LD
#RW1SN   is CPU SHARED, has recid: Y4, flags: A0, and is LG-LD
End of Display
```

### ZTEST DEB5 MAP FACID-QB24D

```
QDER0005I  Company record display for #QB24D   which uses record ID-A3
   Ordinal 0000 at file address F4034801 contains 000A entries
   Ordinal 0001 at file address F4034805 contains 0008 entries
   Ordinal 0002 at file address F4034809 contains 0007 entries
   Ordinal 0003 at file address F403480D contains 0002 entries
   Ordinal 0004 at file address F4034811 contains 000B entries
```

 © Copyright IBM Corp. 2006

```
   Ordinal 0005 at file address F4034815 contains 0008 entries
   Ordinal 0006 at file address F4034819 contains 000B entries
   Ordinal 0007 at file address F403481D contains 0006 entries
   Ordinal 0008 at file address F4034821 contains 000C entries
   Ordinal 0009 at file address F4034825 contains 0007 entries
   Ordinal 000A at file address F4034829 contains 0004 entries
   Ordinal 000B at file address F403482D contains 0005 entries
   Ordinal 000C at file address F4034831 contains 0004 entries
   Ordinal 000D at file address F4034835 contains 0005 entries
   Ordinal 000E at file address F4034839 contains 000E entries
   Ordinal 000F at file address F403483D contains 0007 entries
   Ordinal 0010 at file address F4034841 contains 0008 entries
   Ordinal 0011 at file address F4034845 contains 0006 entries
   Ordinal 0012 at file address F4034849 contains 0008 entries
   Ordinal 0013 at file address F403484D contains 0007 entries
```

---

### ZTEST DEB5 MAP FACID-QB24D ORDINAL-D

```
QDER0005I  Company record display for #QB24D   which uses record ID-A3
   Ordinal 000D at file address F4034835 contains 0005 entries
Slot 0000 has #QB1SN   ordinal 0008 ID-YA at file address F402B020
Slot 0001 has #QB1LN   ordinal 0007 ID-Y4 at file address F402881D
Slot 0002 has #QB34D   ordinal 0001 ID-A5 at file address F4035005
Slot 0003 has #QB14D   ordinal 0007 ID-A1 at file address F405801D
Slot 0004 has #RB1SD   ordinal 0003 ID-Y8 at file address F402B80C
End of display for company record #QB24D
```

---

### ZTEST DEB5 MAP FACID-QB1SN

```
QDES0005I  Origin record display for #QB1SN   which uses record ID-YA
   Ordinal 0000 at file address F402B000 contains 0005 entries
   Forward chain = F402B024   Backward chain = 00000000
   Ordinal 0001 at file address F402B004 contains 0005 entries
   Forward chain = F402B028   Backward chain = 00000000
   Ordinal 0002 at file address F402B008 contains 0005 entries
   Forward chain = F402B030   Backward chain = 00000000
   Ordinal 0003 at file address F402B00C contains 0005 entries
   Forward chain = F402B02C   Backward chain = 00000000 _
   Ordinal 0004 at file address F402B010 contains 0005 entries
   Forward chain = 00000000   Backward chain = 00000000
   Ordinal 0005 at file address F402B014 contains 0005 entries
   Forward chain = 00000000   Backward chain = 00000000
   Ordinal 0006 at file address F402B018 contains 0005 entries
   Forward chain = F402B034   Backward chain = 00000000
   Ordinal 0007 at file address F402B01C contains 0005 entries
   Forward chain = F402B038   Backward chain = 00000000
   Ordinal 0008 at file address F402B020 contains 0005 entries
   Forward chain = F402B03C   Backward chain = 00000000
   Ordinal 0009 at file address F402B024 contains 0005 entries
   Forward chain = F402B040   Backward chain = F402B000
   Ordinal 000A at file address F402B028 contains 0005 entries
   Forward chain = 00000000   Backward chain = F402B004
   Ordinal 000B at file address F402B02C contains 0005 entries
   Forward chain = F402B048   Backward chain = F402B00C
   Ordinal 000C at file address F402B030 contains 0005 entries
   Forward chain = 00000000   Backward chain = F402B008
   Ordinal 000D at file address F402B034 contains 0005 entries
   Forward chain = 00000000   Backward chain = F402B018
   Ordinal 000E at file address F402B038 contains 0005 entries
```

© Copyright IBM Corp. 2006

```
   Forward chain = 00000000   Backward chain = F402B01C
   Ordinal 000F at file address F402B03C contains 0005 entries
   Forward chain = F402B044   Backward chain = F402B020
   Ordinal 0010 at file address F402B040 contains 0005 entries
   Forward chain = F402B04C   Backward chain = F402B024
   Ordinal 0011 at file address F402B044 contains 0005 entries
   Forward chain = 00000000   Backward chain = F402B03C
   Ordinal 0012 at file address F402B048 contains 0005 entries
   Forward chain = 00000000   Backward chain = F402B02C
   Ordinal 0013 at file address F402B04C contains 0005 entries
   Forward chain = 00000000   Backward chain = F402B040
End of display for origin record #QB1SN
```

---

### ZTEST DEB5 MAP FACID-QB1SN ORD-6

```
QDES0005I  Origin record display for #QB1SN   which uses record ID-YA
   Ordinal 0006 at file address F402B018 contains 0005 entries
   Forward chain = F402B034   Backward chain = 00000000
Slot 0000 references record with ID-YA at file address 0000000000084F48
Slot 0001 references record with ID-YA at file address 0000000000084F88
Slot 0002 references record with ID-YA at file address 0000000000000000
Slot 0003 references record with ID-YA at file address 0000000000084ED8
Slot 0004 references record with ID-YA at file address 0000000000000000
End of display for origin record #QB1SN
```

---

### ZTEST DEB5 MAP FACID-QJUNK

```
QDER0001E  FACID #QJUNK   is not defined for this SSU or processor
```

**Explanation:** Only record types defined in the DEB5 master segment can be processed

---

### ZTEST DEB5 FIX

```
QDE80000I 13.42.18 FIX COMPLETE.  POOL COUNT:      252 ,COMP NAME: #QB24D
                   CHAIN ERRORS ENCOUNTERED:        0.
```

---

### ZTEST DEB5 POOL

```
QDEK0001E 13.42.47 BAD FILE ADDR FF058011 IN CM1CHN OF
                   COM ENTRY (FA: F4034805  OFFSET: 060)_
QDE50000I 13.42.47 COMP NAME: #QB24D   POOL COUNT:
                   SMALL:       68  LARGE:      101  4K:       81.
QDE50001I 13.42.47 CHAIN ERROR(S) ENCOUNTERED:        1.+
```

**Explanation:** An invalid file address has been found in the specified COM record.

---

### ZTEST DEB5 POOL

```
QDEK0001E 13.42.47 BAD FILE ADDR FF02A848 IN CM2FCN OF ORIGIN  REC (F402A814)
QDE50000I 13.42.47 COMP NAME: #QB24D   POOL COUNT:
                   SMALL:       68  LARGE:      101  4K:       83.
QDE50001I 13.42.47 CHAIN ERROR(S) ENCOUNTERED:        1.+
```

**Explanation:** An invalid file address has been found in the specified ORG record's forward chain.

 © Copyright IBM Corp. 2006

### ZTEST DEB5 POOL

```
QDEK0001E 13.42.47 BAD FILE ADDR 00000000FF084F50 IN CM2CHN OF
                   ORG ENTRY (FA: F402A814  OFFSET: 120)_
QDE50000I 13.42.47 COMP NAME: #QB24D   POOL COUNT:
                   SMALL:       67  LARGE:      101  4K:       83.
QDE50001I 13.42.47 CHAIN ERROR(S) ENCOUNTERED:        1.+
```

**Explanation:** An invalid file address has been found in the specified ORG record's DEST pointer.

### ZTEST DEB5 POOL

```
QDEK0001E 13.42.47 DEST    REC (0000000000084F50) HAS BAD RECID/RCC (E8F777C0)

                   CHAINED FROM ORIGIN  ENTRY (FA: F402A814  OFFSET: 120)_
QDE50000I 13.42.47 COMP NAME: #QB24D   POOL COUNT:
                   SMALL:       67  LARGE:      101  4K:       83.
QDE50001I 13.42.47 CHAIN ERROR(S) ENCOUNTERED:        1.+
```

**Explanation:** An invalid record ID has been found in the specified ORG record's DEST pointer.

### ZTEST DEB5 POOL

```
QDEK0001E 13.42.47 ORIGIN  REC (F402883D) HAS BAD RECID/RCC (E8F477C0)
                   CHAINED FROM COMPANY ENTRY (FA: F4034815  OFFSET: 060)
QDE50000I 13.42.47 COMP NAME: #QB24D   POOL COUNT:
                   SMALL:       68  LARGE:      101  4K:       83.
QDE50001I 13.42.47 CHAIN ERROR(S) ENCOUNTERED:        1.+
```

**Explanation:** An invalid record ID has been found in the specified COM record's ORG pointer.

### ZTEST DEB5 POOL

```
QDEK0001E 13.42.47 COMPANY REC (F4034815) HAS BAD RECID/RCC (C1C177C0)
                   CHAINED FROM COMPANY REC (F4034811)_
QDE50000I 13.42.47 COMP NAME: #QB24D   POOL COUNT:
                   SMALL:       66  LARGE:       98  4K:       81.
QDE50001I 13.42.47 CHAIN ERROR(S) ENCOUNTERED:        1.+
```

**Explanation:** An invalid record ID has been found in the specified COM record forward chain.

© Copyright IBM Corp. 2006

## ZTEST DEB5 DCHAINS FADD-F4034801

```
QDET0005I  COM: #QB24D   ID: A3 Ord: 0000 FA: F4034801
QDEU0005I    ORG: #RB24D   ID: A4 Ord 0000 FA: F4036001
QDEU0009I      Slot 0000 DEST: #RB24D   ID-A4 FA: 000000000016E725
QDEU0014I      DEST FA: 000000000016E725 FCH 0000000000000000  _
QDEU0005I    ORG: #RB1LN   ID: Y5 Ord 0005 FA: F4029815
QDEU0009I      Slot 0000 DEST: #RB1LN   ID-Y5 FA: 0000000004022FA9
QDEU0014I      DEST FA: 0000000004022FA9 FCH 0000000000000000
QDEU0005I    ORG: #RB1LN   ID: Y5 Ord 0001 FA: F4029805
QDEU0009I      Slot 0000 DEST: #RB1LN   ID-Y5 FA: 0000000004022FC9
QDEU0014I      DEST FA: 0000000004022FC9 FCH 0000000000000000
QDET0013I  COM: #QB24D   FCH Ord: 01 FA: F4034805
QDET0005I  COM: #QB24D   ID: A3 Ord: 0001 FA: F4034805
QDEU0005I    ORG: #RB24D   ID: A4 Ord 0002 FA: F4036009
QDEU0009I      Slot 0000 DEST: #RB24D   ID-A4 FA: 000000000016E6F5
QDEU0014I      DEST FA: 000000000016E6F5 FCH 0000000000000000
QDEU0009I      Slot 0001 DEST: #RB24D   ID-A4 FA: 000000000016E6E9
QDEU0014I      DEST FA: 000000000016E6E9 FCH 0000000000000000
End of display
```

## ZTEST DEB5 DFILE FADD-000000000016E725

```
QDEW0001E  Display of DEST  record 000000000016E725
  Record ID: A4
  Record Code Check: 77
  Record Control Byte: C0
  Last PGM to file: QDEB
  Access Cnt: 0001
  DEST Fch: 0000000000000000
  DEST Bch: 0000000000000000

  Max Item Cnt: 0047  _
  Record Ord Number: 0000
  FADD of Parent COM Rec: F4034801
  FADD of Parent ORG Rec: F4036001

  FADD of this Rec: 000000000016E725
  Inuse Entry Count:    1
  First File Time Stamp: 02JUL 13.52.29
  Last File Time Stamp:

Slot: 0000
    Company Name: 0000
    FADD to chained ORG Record: 00000000  _
    Sub-System User Name: HPN
    Indicator Byte 1: 80
    Indicator Byte 2: 00

    RECID of ORG Record: A4
    FACID of ORG Record: #RB24D
    Max Entries: 0000
    Access Count: 00000000
    Max Ordinal Number for FACID: 0000

End of display for DEST+
```

 © Copyright IBM Corp. 2006

# System Errors

---

### DEB000

**Explanation:** Error on find or file during an INIT.

**User Action:** Find the reason for the bad find or file and correct. DEB5 cannot be started because the initialization was ended.

**Message Appended:** ERROR ON FIND/HOLD, INITIALIZATION ABORTED

---

### DEB002

**Explanation:** Error on find or file of DEB5 keypoint.

**User Action:** Find the reason for the bad find or file and correct. The DEB5 function that caused this dump ended abnormally and has to be run again when the problem is fixed. DEB5 cannot be started with the database in this state.

**Message Appended:** ERROR ON FIND/HOLD OF DEB5 KEYPOINT

---

### DEB003

**Explanation:** Hardware I/O error was encountered during a find or file of a DEB5 record.

**User Action:** Issue a FIX to attempt to correct the database and investigate the source of error. DEB5 need not be stopped.

**Message Appended:** HARDWARE I/O ERROR

---

### DEB004

**Explanation:** An error other than hardware, Recid/Rcc failure, or bad file address was encountered during a find or file of a DEB5 record.

**User Action:** Issue a FIX to attempt to correct the database and investigate the source of error. DEB5 need not be stopped.

**Message Appended:** Unknown Error

---

### DEB005

**Explanation:** A forward chain error has been encountered. There is a bad file address in a chain, or a bad record ID or record code check in one of the records.

**User Action:** Issue a FIX. DEB5 needs to be stopped and in 1052. Determine the cause of the error.

**Messages Appended:** FORWARD CHAIN ERROR ON COM REOCRD - SEE DUMP
FORWARD CHAIN ERROR ON ORG RECORD - SEE DUMP
FORWARD CHAIN ERROR ON DEST RECORD - FIX DATABASE

© Copyright IBM Corp. 2006

### DEB006

**Explanation:** Invalid value(s) have been found in the DEB5 keypoint. The keypoint has been corrupted or was never initialized.

**User Action:** Stop DEB5 if active. Issue a RELPOOLS, then an INIT to restore the keypoint. **Note:** The INIT will destroy the current database.

**Message Appended:** KEYPOINT CORRUPTION

### DEB007

**Explanation:** Possible corruption of DEB5 master segment (QBB4). Invalid data was found there.

**User Action:** Correct the problem before DEB5 can be started or the database is built.

**Message Appended:** CHECK DEB5 MASTER SEGMENT

### DEB008

**Explanation:** Error on file during UPDATE processing. The update was not successfully completed.

**User Action:** Correct the problem if it persists.

**Message Appended:** ERROR ON FILE DURING UPDATE - ECB EXITED

### DEB009

**Explanation:** Error on file during ADD processing. The entry was not added to the database.

**User Action:** Correct the problem if it persists.

**Message Appended:** ERROR ON FILE DURING ADD PROCESSING

### DEB010

**Explanation:** Error on file during DELETE processing. The entry was not deleted from the database.

**User Action:** Correct the problem if it persists.

**Message Appended:** ERROR ON FILE DURING DELETE PROCESSING

### DEB011

**Explanation:** Error on find of DEB5 master segment QBB4.

**User Action:** See if QBB4 was loaded to system. Possible allocation problem.

**Message Appended:** ERROR ON FIND OF MASTER SEGMENT

### DEB013

**Explanation:** There is a loop in a destination chain.

 © Copyright IBM Corp. 2006

**User Action:** Stop DEB5 and correct by zeroing the forward chain pointer of the head record in the problem chain.

**Message Appended:** DESTINATION CHAIN IN LOOP - DATABASE NEEDS FIXING

---

### DEB020

**Explanation:** Prime and dupe copies of a record are not equal.

**User Action:** None. This is issued by fix processing once per COM record, regardless of the number of actual bad records.

**Message Appended:** PRIME/DUPE OUT OF SYNCH - FIXED - POOLS MAY BE LOST

---

### DEB021

**Explanation:** Unable to perform a FINSC/FILSC during a fix.

**User Action:** None. This is issued by fix processing once per COM record, regardless of the number of actual bad records. You could make sure both prime and dupe are online however.

**Message Appended:** UNABLE TO SYNCH PRIME/DUPE

---

### DEB022

**Explanation:** A previously non-zero record was found to be zeroed during delete processing. It will be tossed out.

**User Action:** Examine the dump to determine why. The prime and dupe copies of the record are attached to the ECB on D4 and D6.

**Message Appended:** BAD RECORD FOUND - FIXING - ONE POOL LOST

---

### DEB023

**Explanation:** Unable to get the prime and dupe copies attached for the previous dump. Informational only.

**User Action:** Examine the dump to determine why. The prime and dupe copies should be accessible unless a mod is down.

**Message Appended:** UNABLE TO GET PRIME/DUPE RECORD FOR DEB022

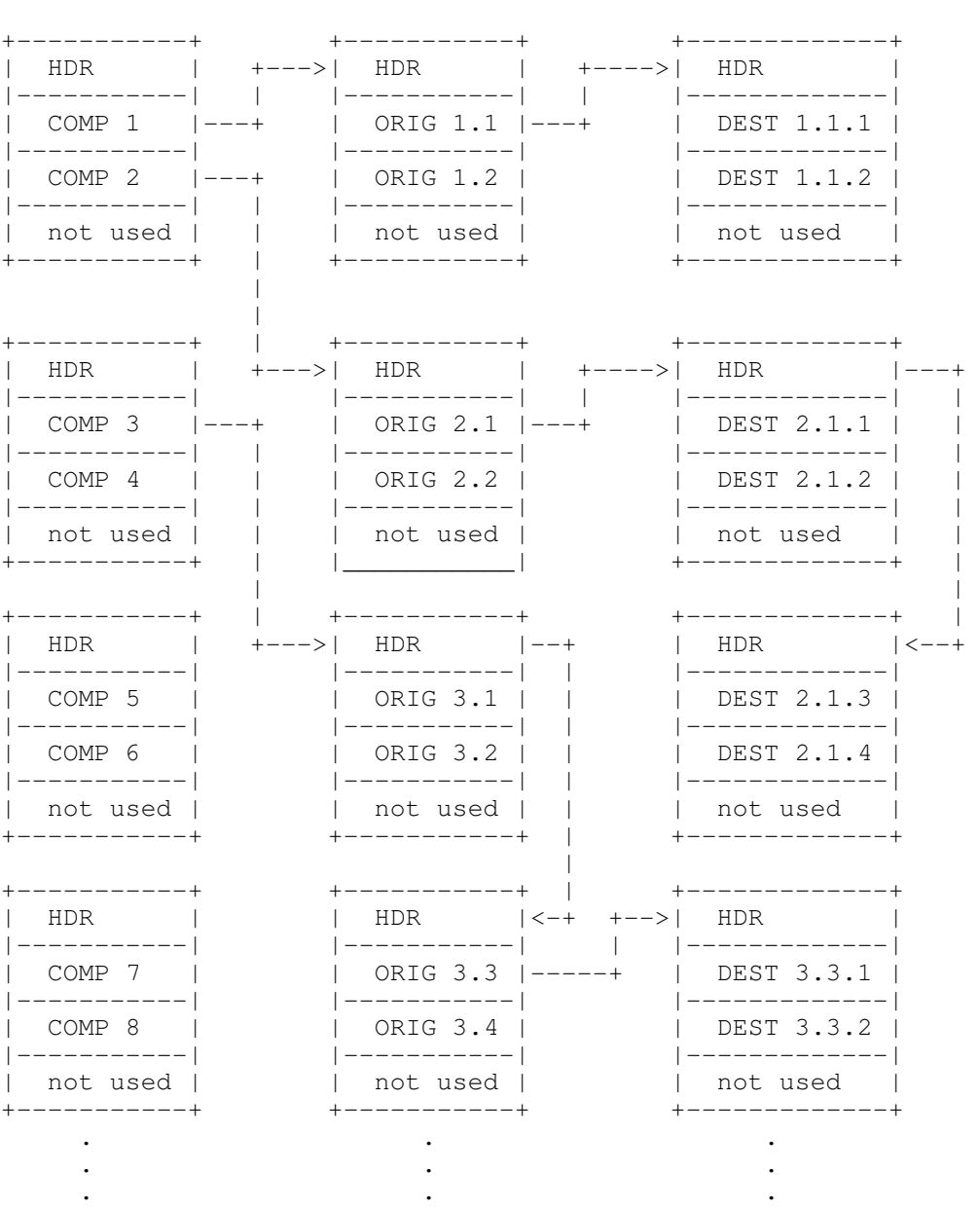
## References

For more information about reading syntax diagrams, also referred to as railroad diagrams, see *Accessibility information* in the TPF Product Information Center.

© Copyright IBM Corp. 2006

# Appendix A: Database structure & organization

The following diagram shows the structure of the database:




 © Copyright IBM Corp. 2006

## Database BUILD Considerations

Understanding the DEB5 BUILD process requires knowledge of the following structural limitations:

- **Company (COM) entries:**
  - The total number of COM entries is calculated using the following formula and adding the results for each Company record type:

    *Allocated Ordinals * Size Factor*

    Where:
    *Allocated Ordinals* = Number of ordinals allocated for the Company record type
    *Size Factor* = Number of COM entries that will fit into each ordinal:

| Small (381) | Large (1055) | 4K (4095) |
|---|---|---|
| 5 | 19 | 83 |

    **Example:** The current BSS database defines one COM record type (#QB24D, 20 ordinals):

```
Max COM entries = (20 ordinals * 83 entries / 4K ordinal) = 1660 COM entries
```

  - Company (COM) entries contain a pointer to an Origin (ORG) record chain. This places a practical limit on the number of usable COM entries determined by the total number of ORG record ordinals.

    **Example:** The current BSS database defines twenty ordinals each of twelve ORG record types:

```
Max usable COM entries = (20 ordinals * 12 record types) = 240 usable COM entries
```

- **Origin (ORG) entries:**
  The total number of ORG entries is calculated using the following formula and adding the results for each Origin record type:

  *Allocated Ordinals * Size Factor*

  Where:
  *Allocated Ordinals* = Number of ordinals allocated for the Origin record type
  *Size Factor* = Number of ORG entries that will fit into each ordinal:

| Small (381) | Large (1055) | 4K (4095) |
|---|---|---|
| 5 | 19 | 83 |

  **Example:** The current BSS database defines twenty ordinals each of twelve ORG record types (four record types in each of three sizes):

```
Small ORG ordinals =  5 entries / ordinal * 20 ordinals * 4  record types = 400
Large ORG ordinals = 19 entries / ordinal * 20 ordinals * 4  record types = 1520
4K ORG ordinals    = 83 entries / ordinal * 20 ordinals * 4  record types = 6640
```

© Copyright IBM Corp. 2006

```
Max ORG entries    = Small + Large + 4K ordinals = 8560 entries
```

- **Destination (DEST) entries:**
  Destination entries are added and removed as needed during normal DEB5 database activity. A randomly determined maximum number of DEST entries per record is set based on the record size. As each DEST record is filled, another POOL record is chained in to a maximum length of 99.

## DEB5 Build Parameters

- The number of COM entries on the BUILD command should not exceed the total number of allocated ORG record ordinals. Even when the number of COM entries is within this guideline, some COM entries may remain permanently NULL due to ORG record chaining.

- The number of ORG entries that can be created by the BUILD command is limited only by the ORG record ordinal allocation. The actual number of ORG entries built, however, will not typically equal the number requested. This is normal, and results when the BUILD process attempts to add to a full ORG record chain.

- Within the constraints listed above, the ratio of COM to ORG entries will determine the "shape" of the database.

  A large COM:ORG ratio will result in a fairly wide tree structure. As the ratio gets smaller, the tree becomes more narrow. A "wide" tree has short ORG chains spread across many COM entries. A "narrow" tree has much longer ORG chains spread across relatively few COM entries.

## MDBF and LC Considerations

- The database that DEB5 builds is self-contained within a subsystem user.

- The DEB5 segments must be loaded to each subsystem and fixed file records to be used as DEB5 records must be defined in the master segment for every SSU in the subsystem.

- If DEB5 and VFA are both active in a LC environment, and the DEB5 records are defined as VFA candidates, then the DEB5 records must also be defined as processor unique in the MASTER RECORD, or all the processor common records must be set in the RIAT to be either VFA synchronized candidates or non-VFA candidates.

- If the records defined in the master segment for a SSU are not defined as processor unique and VFA is active then DEB5 can only be run in that subsystem user on one processor in a LC complex.

## VFA Considerations

- In order to exercise VFA, the programmer has the option of defining DEB5 records as VFA candidates. This is accomplished manually via the ZRTDM command, or automatically by use of the ZTEST DEB5 VFA command. The master records, QBB4 and QBB5, contain the information as to whether or not a particular DEB5 record is CPU common or unique. This information can be altered according to the programmers requirements.

- If the records defined in the master segment for a SSU are not defined as processor unique and VFA is active then DEB5 cannot be run in the SAME subsystem user on DIFFERENT

 © Copyright IBM Corp. 2006

processors unless all the processor shared records are not VFA candidates or they are VFA synchronized. This restriction also applies to starting DEB5 in one processor, stopping it, and then starting DEB5 again in another processor. This cannot be done unless VFA is stopped before DEB5 is started again.

## Capture Considerations

- The DEB5 database may be used to ensure that capture is working correctly.

- Before running capture, the appropriate RIAT attributes must be set. This can be done manually through the ZRTDM command, or automatically by the ZTEST DEB5 XCP command.

© Copyright IBM Corp. 2006