z/TPFDF Update

2023 TPF Users Group Conference
April 24-26, Dallas, TX
Database/TPFDF Subcommittee

—

Chris Filachek

IBM

z/TPFDF System Error Enhancements

Problem Statement

Some z/TPFDF system errors do not include any message text on the console or only provide a z/TPFDF file ID and file addresses.

Several of these system errors are the result of application or I/O errors, which means developers have minimal information available when they start analyzing dumps.

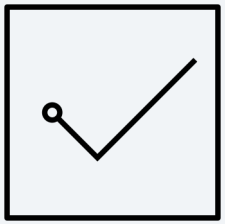
Pain Points

Zach
application developer

When testing my application, I received a DB0101 system error indicating an invalid LREC size.

Without reviewing the dump and having some knowledge of z/TPFDF internals, I can’t easily determine if the invalid LREC size is for a new or existing LREC or where it is in the subfile.

Value Statement



Select z/TPFDF system errors related to application and I/O errors have been updated with additional diagnostics, making it faster and easier to determine the cause of those errors.

- Updated 19 z/TPFDF system errors to include additional diagnostic information in the message text
- Created 2 new z/TPFDF system errors for existing conditions previously included with other system errors

Delivered in APAR PH48876 (Dec. 2022)

Faster Problem Determination

Zach
application developer

When testing my application, I received a DB0101 system error indicating an invalid LREC size. The message text states that a new LREC has an invalid size and R15 points to the new LREC in the dump.

With this information, I can quickly diagnose the problem in my application and start working on a fix.

```
CPSE0162E 09.33.48 IS-0001 SS-BSS SSU-BSS  SE-000794 OPR-IDB0101
010000A TRC-QXG6        OWNER-drvrDFTDfrmCVZZ-ECBAdr:11A24000.
UTDF    OBJ-ub90            00002B5C  LOADSET-BASE
BD90 PFA 0000000068280357 00 CFA 0000000068280357
 FAILING FILE ADDRESS 0000000068280357
THE LREC SIZE OF 1F40 IN THE NEW LREC EXCEEDS THE MAXIMUM SIZE OR IS NOT VALID.
```

Additional z/TPFDF System Error Examples

- DB017E: Message text added with basic z/TPFDF information (file ID and file addresses) and description of the error

```
CPSE0162E 09.51.24 IS-0001 SS-BSS SSU-BSS  SE-NODUMP OPR-IDB017E
010000A TRC-QXG4        OWNER-drvrDFTDfrmCVZZ-ECBAdr:11A2D000.
UTDF    OBJ-ub90            00002B5C  LOADSET-BASE
BDDD PFA 0000000000000000 00 CFA 0000000000000000
A CURRENT LREC MUST BE LOCATED BEFORE THAT LREC CAN BE RETAINED.
```

- DB011B: Additional text added describing why a FIND error occurred along with the failing file address

```
CPSE0162E 09.48.17 IS-0001 SS-BSS SSU-BSS  SE-000798 OPR-IDB011B
010000A TRC-QXEU        OWNER-drvrDFTDfrmCVZZ-ECBAdr:11A24000.
UTDF    OBJ-ub90            00002B5C  LOADSET-BASE
BD14 PFA 00000000811E943F 9B CFA 00000000811E943F
 ID CHECK FAILURE FOR FILE ADDRESS 00000000811E946F
```

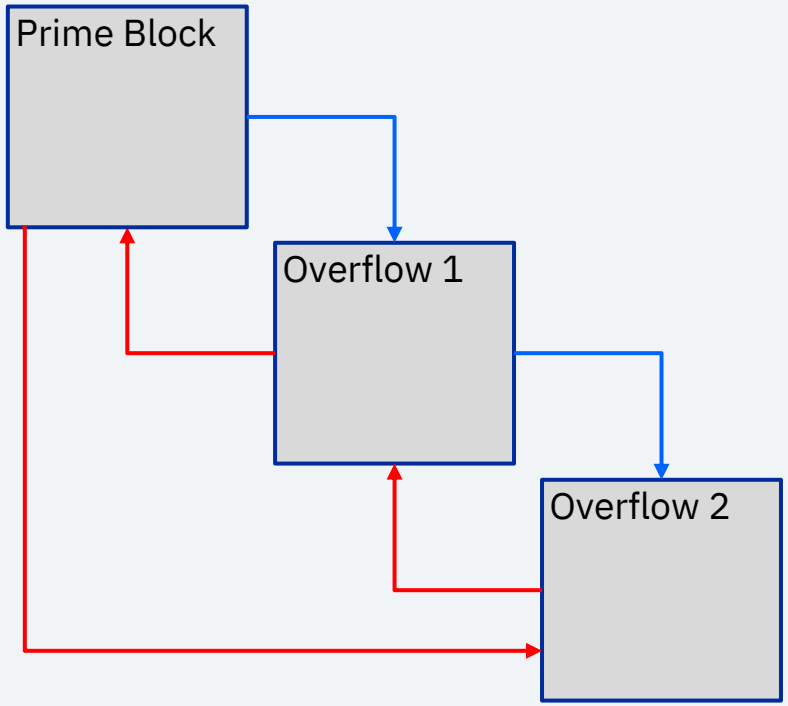
z/TPFDF CRUISE

Validation Enhancements

Problem Statement

- The z/TPFDF CRUISE utility does not use or validate backward chains in z/TPFDF subfiles.

- CRUISE only validates forward chains.



Pain Points

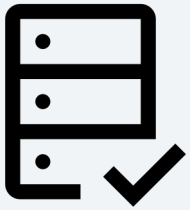
Andres
database admin

Due to a recent issue, some of the backward chains in our z/TPFDF subfiles were corrupted.

We were alerted to the corruption because a z/TPFDF API issued an error when using backward chains, such as reading backwards or deleting empty blocks in a subfile.

Packing a subfile appears to quietly fix the issue, but we don’t have a way to determine the extent of the corruption.

Value Statement



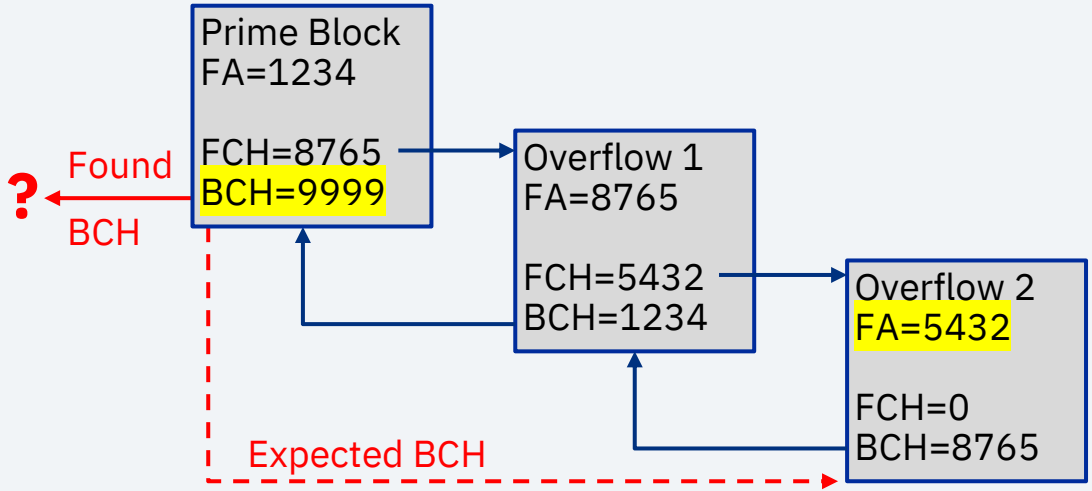
The z/TPFDF CRUISE utility (VERIFY, PACK and CAPTURE functions) validates backward chains in most z/TPFDF R-type files and issues an error message if an incorrect backward chain is found.

- Any pack of the subfile will fix incorrect backward chains (CRUISE, ZUDFM PACK, or API)
- CRUISE also validates the IBM-managed STDAUT bits and issues messages if incorrect

Delivered in APAR PH44215 (July 2022)

Example 1: Incorrect Backward Chain in Prime Block

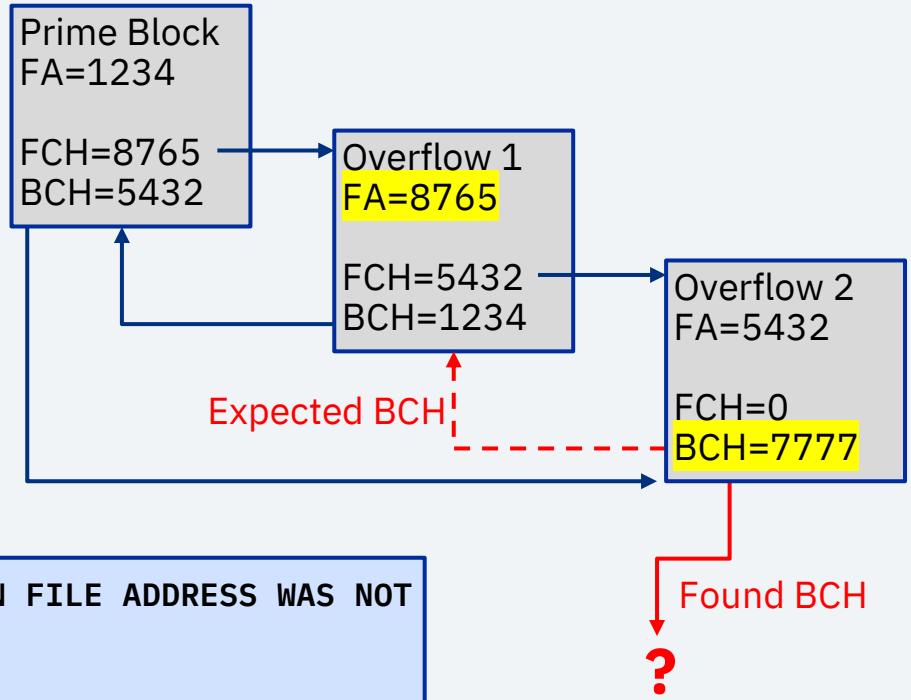
- The backward chain (BCH) in the prime block should contain the file address of the last overflow
- CRUISE issues message FCRU0222E when prime BCH is not as expected




```
FCRU0222E 15.25.16 THE EXPECTED BACKWARD CHAIN FILE ADDRESS WAS NOT
  FOUND IN THE PRIME BLOCK FOR FILE ID B321
  PRIME FA         - 0000000000001234
  EXPECTED BCH FA  - 0000000000005432
  FOUND BCH FA     - 0000000000009999+
```

Example 2: Incorrect Backward Chain in an Overflow

- The backward chain (BCH) in the overflow should contain the file address of the previous block
- CRUISE issues message FCRU0223E when overflow BCH is not as expected

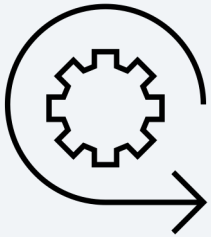



```
FCRU0223E 15.25.16 THE EXPECTED BACKWARD CHAIN FILE ADDRESS WAS NOT
  FOUND IN AN OVERFLOW BLOCK FOR FILE ID B321
  PRIME FA         - 0000000000001234
  OVERFLOW FA      - 0000000000005432
  EXPECTED BCH FA  - 0000000000008765
  FOUND BCH FA     - 0000000000007777+
```

ZUDFM INIT Enhancements

Subsystem User Options

Problem Statement



ZUDFM INIT initializes the z/TPFDF file only for the subsystem user where the command is entered.

The ZIFIL command is not an option because it doesn’t support initializing records for use by z/TPFDF. For example, it doesn’t set up the records for z/TPFDF 8–byte headers (FARF6) or z/TPFDF encryption.

Pain Points

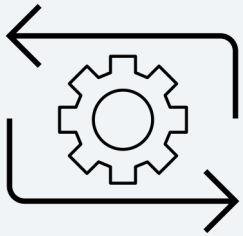
Andres
database admin

I need to initialize a subsystem user unique z/TPFDF file for all subsystem users (SSUs). With dozens of SSUs, I need to enter dozens of commands to initialize this file.

This also requires dozens of files or tapes if I use the FILE parameter with ZUDFM INIT to save the contents of the file before initialization.

Finally, one of the SSUs is dormant and ZUDFM INIT can’t initialize records on a dormant SSU.

Value Statement



An operator can enter a single ZUDFM INIT command to initialize a z/TPFDF file for all subsystem users on the current subsystem or initialize a z/TPFDF file for a dormant subsystem user.

If using the FILE parameter with ZUDFM INIT, prime blocks for all subsystem users can be saved to a single file or tape.

Delivered in APAR PH43813 (June 2022)

Example: Initialize a File for All SSUs

```
==> WP/ZUDFM INIT B231 SSU-ALL
UDFM0384A 11.04.17 THE FOLLOWING FILE WILL BE INITIALIZED WITH HOLD.
                   ID-B231  FVN-00  FACET-#HOTEL  SSU-ALL
                   BOR-0000000000   EOR-VARIOUS

                   TO CONFIRM THE INITIALIZATION,
                   REPEAT THE COMMAND WITHIN 2 MINUTES.+
```

"VARIOUS" indicates different end ordinals for each SSU

```
==> WP/ZUDFM INIT B231 SSU-ALL
UDFM0386I 11.04.19 ZUDFM DATABASE INITIALIZATION COMPLETED ON SSU WP1
                   ID-B231  FVN-00  TOTAL ORDINALS-0000002000+

UDFM0386I 11.04.20 ZUDFM DATABASE INITIALIZATION COMPLETED ON SSU WP2
                   ID-B231  FVN-00  TOTAL ORDINALS-0000001590+

UDFM0386I 11.04.20 ZUDFM DATABASE INITIALIZATION COMPLETED ON SSU WP3
                   ID-B231  FVN-00  TOTAL ORDINALS-0000001180+
```

Completion message for each SSU

```
UDFM0383I 11.04.20 ZUDFM DATABASE INITIALIZATION COMPLETED
                   ID-B231  FVN-00  TOTAL ORDINALS-0000004770+
```

Completion message for the command

Initializing a Dormant SSU

- Dormant SSUs must be initialized from an active SSU
  - ➢ Initialize using SSU-ALL or SSU-<dormantSSU>
- RELEASE parameter is supported only for active SSUs
  - ➢ If RELEASE is specified with SSU-ALL, active SSUs are initialized and dormant SSUs are skipped (UDFM0418W issued for each dormant SSU)

```
==> WP1/ZUDFM INIT B075 SSU-WP2 NOHOLD
UDFM0384A 11.25.02 THE FOLLOWING FILE WILL BE INITIALIZED WITH NOHOLD.
                   ID-B075  FVN-00  FACET-#BTREE  SSU-WP2
                   BOR-0000000052   EOR-0000000077

                   TO CONFIRM THE INITIALIZATION,
                   REPEAT THE COMMAND WITHIN 2 MINUTES.+

==> WP1/ZUDFM INIT B075 SSU-WP2 NOHOLD
UDFM0383I 11.25.04 ZUDFM DATABASE INITIALIZATION COMPLETED
                   ID-B075  FVN-00  TOTAL ORDINALS-0000000026+
```

Indicates HOLD, NOHOLD, or RELEASE was requested

Single completion message when initializing only 1 SSU

ZUDFM INIT Enhancements

Cancel Options

Pain Points

Andres
database admin

I have to enter ZUDFM INIT twice to start the initialization, but there's no clear way to cancel the first ZUDFM INIT.

The first ZUDFM INIT times out in 2 minutes, but the request is still in the database and interferes with future ZUDFM INIT commands. This causes the next ZUDFM INIT command to fail, even if it is entered days later.

Value Statement

An operator can enter a single ZUDFM INIT command to cancel a pending ZUDFM INIT request.

If a pending ZUDFM INIT request times out, the request is automatically removed so future ZUDFM INIT requests are not impacted.

Delivered in APAR PH43813 (June 2022)

Example: Cancelling ZUDFM INIT Requests

```
==> ZUDFM INIT B075
UDFM0384A 10.25.20 THE FOLLOWING FILE WILL BE INITIALIZED WITH HOLD.
                   ID-B075  FVN-00  FACET-#BTREE  SSU-HPN
                   BOR-0000000052   EOR-0000000077

                   TO CONFIRM THE INITIALIZATION,
                   REPEAT THE COMMAND WITHIN 2 MINUTES.+

==> ZUDFM INIT CANCEL
UDFM0389I 10.25.26 THE PENDING INITIALIZATION FOR FILE ID B075 IS CANCELED.+
```

Example 1:

Use the CANCEL parameter to cancel a pending request

```
==> ZUDFM INIT B075
UDFM0384A 11.30.32 THE FOLLOWING FILE WILL BE INITIALIZED WITH HOLD.
                   ID-B075  FVN-00  FACET-#BTREE  SSU-HPN
                   BOR-0000000052   EOR-0000000077

                   TO CONFIRM THE INITIALIZATION,
                   REPEAT THE COMMAND WITHIN 2 MINUTES.+

UDFM0400E 11.32.32 THE PENDING INITIALIZATION FOR FILE ID B075 TIMED OUT.+
```

Example 2:

After 2 minutes, a pending request is automatically cancelled

Application Accessible

SW00SR Fields

New Application Accessible SW00SR Field

- Existing SW00SR fields for use by applications
  - SW00USI (1 byte)
  - SW00USA (4 bytes)
- New application accessible SW00SR field
  - SW00US8 (8 bytes)
  - Mapped as both a long integer and void pointer in C/C++
  - **Delivered in APAR PH47742 (July 2022)**

```
#include <cdf.h>
dft_fil *filePtr;
filePtr = dfopn("GR12AB  ", “AB", 0);

// Existing 1-byte user field
filePtr->sw00us1 = ‘A’;

// Existing 4-byte user field
filePtr->sw00usa = 1000;

// Example 1: Save a timestamp,
// long integer, or other 8-byte value
// in SW00US8
filePtr->sw00us8 = time();

// Example 2: Save a 64-bit memory
// address in SW00US8
filePtr->sw00us8_ptr = malloc64();
free(filePtr->sw00us8_ptr);
```

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

z/TPFDF CRUISE

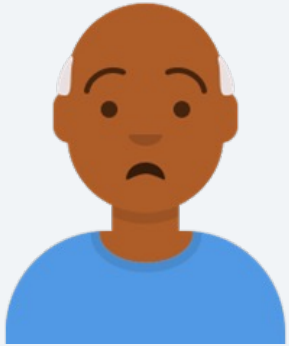
Low Priority & Runtime Enhancements

Problem Statement #1

The z/TPFDF CRUISE utility can be run as low priority. However, the CRUISE commands must be prefixed with **–LP/** for CRUISE to run as low priority.

After defining the CRUISE run in the CRUISE parameter table, it can be easy to forget the **–LP/** prefix when starting CRUISE.

As Is User Story



Derrick
operator

We have predefined CRUISE parameter tables for several z/TPFDF files. The ECB levels in those tables were set assuming low priority.

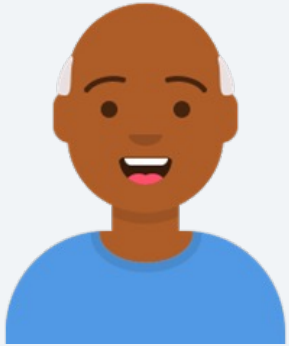
To run CRUISE as low priority, we need to remember the **–LP/** prefix when starting CRUISE with the ZFCRU START command (entered twice) and when restarting CRUISE after it pauses.

If we forget the **–LP/** prefix, CRUISE can impact transactional work.

Value Statement #1

An operator can set a CRUISE option so CRUISE runs as low priority, allowing operators to confidently setup CRUISE to run as low priority without using command prefixes.

To Be User Story

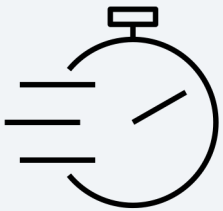


Derrick
operator

We have predefined CRUISE parameter tables for several z/TPFDF files. CRUISE is predefined to run as low priority and the ECB levels are set accordingly.

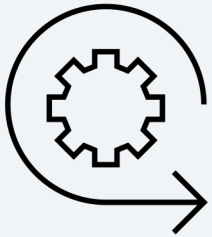
When CRUISE is started, it automatically runs as low priority and doesn’t impact transactional work.

Additional Value when Running as Low Priority



When CRUISE is run as low priority and fenced I-streams are available, CRUISE VERIFY and PACK functions will utilize some of the fenced I-streams to decrease CRUISE runtime without impacting transactional workload.

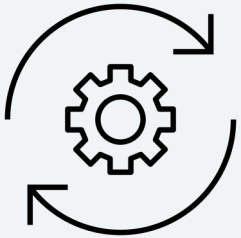
Problem Statement #2



When CRUISE chain chases z/TPFDF databases, an ECB is created to chase each fixed record ordinal (subfile) in the database.

With very wide and shallow databases (zero or very few overflows), there is very little work for each ECB. Most of the CPU processing is creating and exiting ECBs.

Value Statement #2



For wide and shallow z/TPFDF files, database administrators can configure those files so each CRUISE ECB chases up to N ordinals (subfiles), reducing the CPU consumption of CRUISE by reducing the total number of created ECBs.

- The DBDEF RANGE parameter was added to provide the same value for recoup chain chasing. CRUISE would be updated to take advantage of the same RANGE parameter.

We want sponsor users!

Our development cycle is driven by your feedback.

We are looking for sponsor users to assist in design and implementation, targeting the following personas:

- Database administrator
- System programmer

We expect to begin engaging with the sponsor users in 3rd Quarter 2023.

If you are interested in participating as a sponsor user, please contact:

Dan Jacobs (dhjacobs@us.ibm.com)

Rob Dunn (strmbrgr@us.ibm.com)

Thank you