z/TPF V1.1

# TPF Users Group – Fall 2012

# z/TPFDF Multiple LREC Buffers

*APAR PM55273*

Chris Filachek
Database/TPFDF Subcommittee

**AIM Enterprise Platform Software**
**IBM z/Transaction Processing Facility Enterprise Edition 1.1.0**

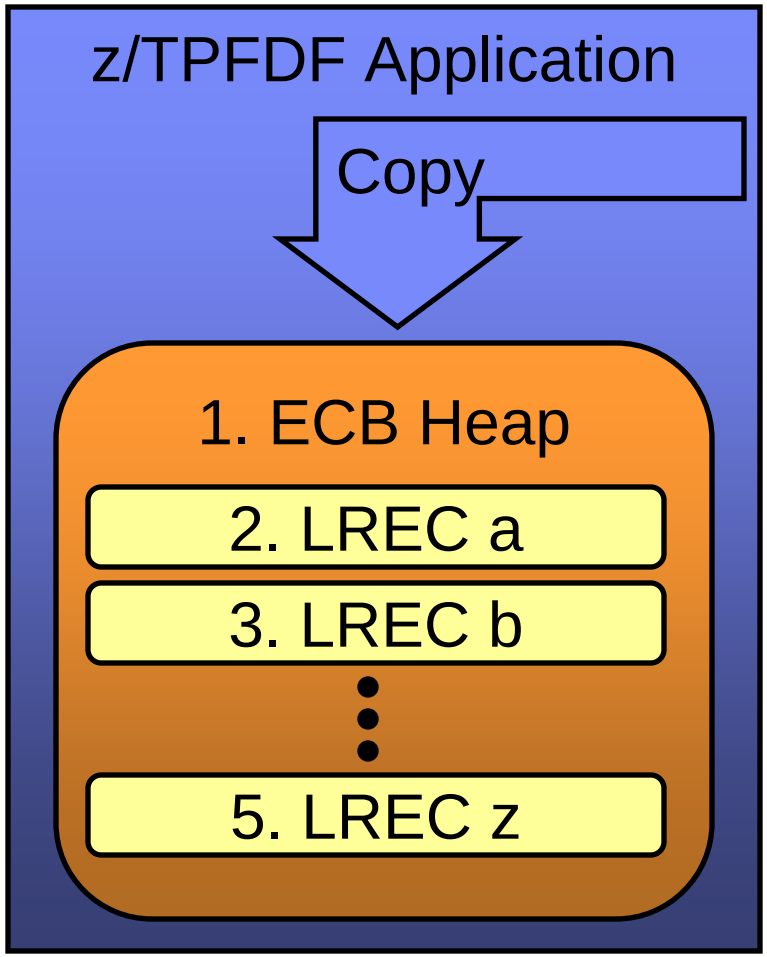
# Agenda

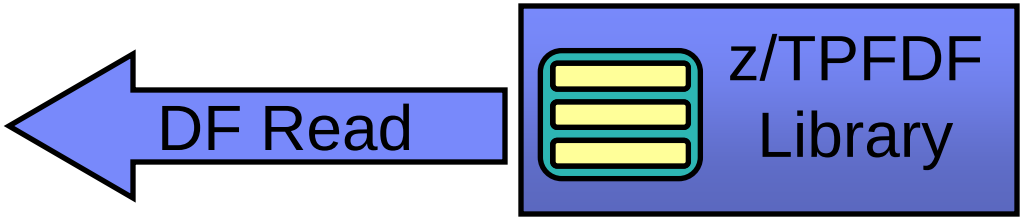
- **Requirement Description**
- **Solution - Multiple LREC Buffers**
- **Coding Examples**

# Requirement Description

- **Create copy of z/TPFDF subfile**
  - Read selected LRECs from z/TPFDF subfile and copy to a buffer
  - Send/save/etc. the buffer outside of z/TPFDF
- **Receive copy of z/TPFDF subfile and enable for local processing**
  - Add LRECs from buffer to subfile
  - Use z/TPFDF APIs to read/update individual LRECs in subfile

# Reading one LREC at a time…




1. Allocate buffer in ECB heap
2. DF read & copy LREC a
3. DF read & copy LREC b
4. …lots of DF reads and copies…
5. DF read & copy LREC z

Result: Inefficient process due to many API calls

# Agenda

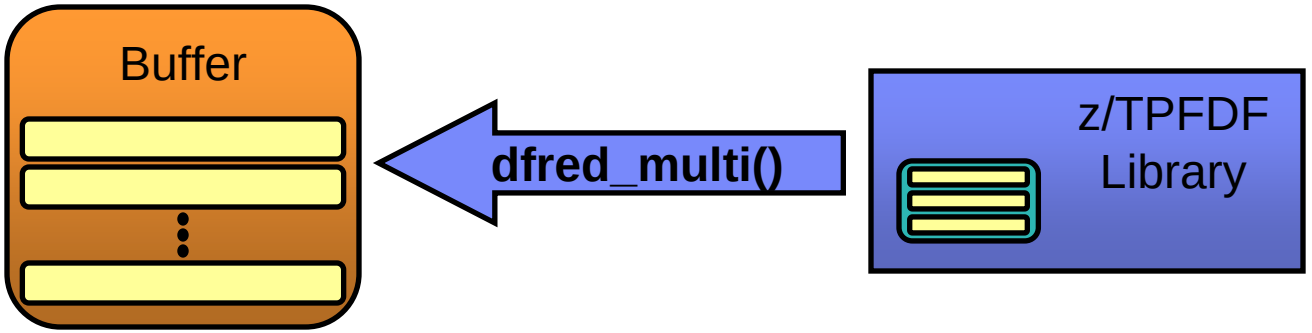
- **Requirement Description**
- **Solution - Multiple LREC Buffers**
- **Coding Examples**

# Solution - Multiple LREC Buffers

- **New buffer structure holds multiple LRECs**
  - Application defines buffer in ECB heap
  - z/TPFDF controls buffer contents
  - May contain fixed or variable size LRECs or LLRs
- **New C and Assembler APIs**
  - Reads multiple LRECs from DF subfile into buffer
  - Adds LRECs in buffer to z/TPFDF subfile
  - Buffer management APIs (initialize, resize, etc.)

# New z/TPFDF Read APIs

- **Read multiple LRECs from subfile to buffer**
  - **`dfred_multi()`** **`DBRED MULTI=`**
  - Read LRECs using standard search keys or no keys
  - Reads until all selected LRECs are read or buffer is full



# New z/TPFDF Add Key and Add LREC APIs

- **Add LRECs in buffer to subfile**
  - **`dfadd_multi()`** **`DBADD MULTI=`**
  - Adds LRECs at appropriate locations using default keys or add keys
  - Unique keys are supported
- **Define add keys prior to add LREC**
  - **`df_setkey_add()`** **`DBSETK #DF_ADD_NEWLREC`**
  - Determines where in the subfile each new LREC is added
  - Can be used with any z/TPFDF add APIs
  - Only type of setkey supported for multiple LREC add

# New Buffer Management APIs

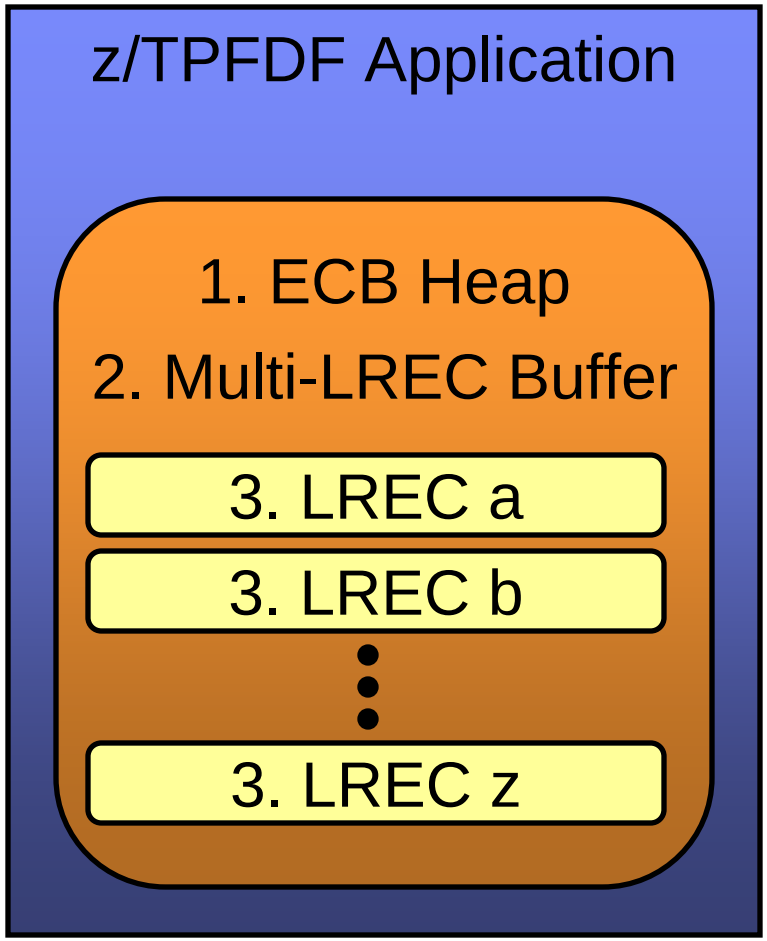
- **Initialize the buffer**
  - **`dfmbuf_init()`** **`DBMBUF INIT`**
  - Initializes a new buffer or reinitializes existing buffer
- **Resize the buffer**
  - **`dfmbuf_resize()`** **`DBMBUF RESIZE`**
  - Allows z/TPFDF to use larger buffer after existing buffer is expanded
- **Get buffer information**
  - **`dfmbuf_getinfo()`** **`DBMBUF GETINFO`**
  - Returns buffer information to application
    - Total buffer size, bytes used, bytes free
    - Number of LRECs

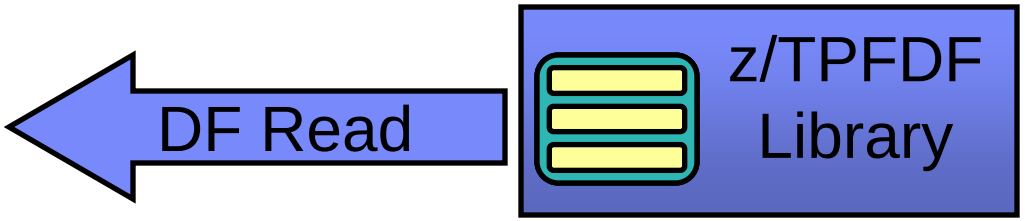
# New SPMs and C Macros for Error Checking

- **DFMULTI_OK**
  - Processing completed successfully
- **DFMULTI_NOTHING_READ**
  - No LRECs were read into the buffer
- **DFMULTI_INCOMPLETE**
  - Buffer full and unable to read all LRECs into buffer
- **DFMULTI_NOTHING_ADDED**
  - No LRECs were added to subfile from buffer

**Note: See z/TPFDF InfoCenter for additional SPMs and C macros for error checking**

# Read LRECs using Multiple LREC Buffers…




1. Allocate buffer in ECB heap
2. Initialize buffer for DF multiple LREC use (inline function)
3. DF read multiple LRECs

Result: Efficient process to read and copy LRECs in a single z/TPFDF API call

# Multiple LREC Buffer is NOT a subfile

- **APIs to read/update/insert/etc. individual LRECs are not provided**
  - Not intended for direct data manipulation
  - Intended for efficient copy of LRECs
  - Copy is intended to be sent or stored outside of z/TPFDF
- **…but I need to read/update/insert/etc. individual LRECs**
  - DBCPY and DBSRT - create a local copy of a subfile
  - Add Multiple LREC buffer to local subfile
  - Temporary subfiles
    - W-type memory resident subfile
    - Short term pools

# Agenda

- **Requirement Description**
- **Solution - Multiple LREC Buffers**
- **Coding Examples***

*** Error checking is left out to keep the coding examples simple for purposes of this presentation.**

# Example 1: Loop to Read all LRECs

```
dft_bufs bufsize = 10000;
dft_buf  *bufptr;

bufptr = (dft_buf *)malloc(bufsize);
dfmbuf_init(buf_ptr, bufsize);

dfred_multi(dffile_ptr, 0, bufptr);

while (DFMULTI_INCOMPLETE(dffile_ptr))
{
   bufsize += 10000;
   bufptr = realloc(bufptr, bufsize);
   dfmbuf_resize(buf_ptr, bufsize);
   dfred_multi(dffile_ptr, 0, bufptr);
}
```

1. Allocate buffer in heap and use dfmbuf_init() to initialize
2. Read as many LRECs as can fit into buffer
3. Check if all LRECs were read
4. If not all LRECs read…
   a. Realloc larger buffer
   b. Tell z/TPFDF buffer is resized
   c. Read more LRECs
   d. Loop to top

# Example 2: Send buffer to remote system

```
int sockdesc;
dft_ubuf  bufinfo;
dft_buf  *bufptr;


dfmbuf_getinfo(bufptr, &bufinfo);


if (bufinfo.DF_MBUFUCNT > 0)
{
  send(sockdesc, (char *)bufptr,
    bufinfo.DF_MBUFUDAT, 0);
}
```

1. **Retrieve data size information**
2. **Check there are 1 or more LRECs in buffer**
3. **Send contents of buffer using favorite comms protocol**

# Example 3: Receive buffer from remote system

```
int sockdesc, bufsize = 10000;
dft_ubuf  bufinfo;
dft_buf  *bufptr = (dft_buf *)malloc(bufsize);

read(sockdesc, (char *)bufptr,
  DF_MBUF_INITIAL_NAB);

dfmbuf_getinfo(bufptr, &bufinfo);

if (bufinfo.DF_MBUFUDAT > bufsize) {
  bufsize = bufinfo.DF_MBUFUDAT;
  bufptr = realloc(bufptr, bufsize);
}
dfmbuf_resize(bufptr, bufsize);

read(sockdesc,
 ((char *)bufptr) + DF_MBUF_INITIAL_NAB,
 bufinfo.DF_MBUFUDAT – DF_MBUF_INITIAL_NAB);

dfadd_multi(dffile_ptr, 0, bufptr);
```

1. Read data into buffer using initial buffer size (header only)
2. Get actual buffer size
3. Realloc buffer as needed. Resize to match actual buffer size
4. Read remaining data into buffer starting after header
5. Add multiple lrecs to local subfile

# Trademarks

- **IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.**

**Notes**

- **Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.**
- **All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.**
- **This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.**
- **All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.**
- **Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.**
- **Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.**
- **This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.**