8-Byte File Address Support (APAR PJ28097)

by Michele Dalbo and Stephanie Daniels, IBM TPF ID Core Team; and Rick Matela, IBM TPF Development

What Is 8-Byte File Address Support?

Before 8-byte file address support, file address reference format 5 (FARF5) limited the address capacity of a TPF 4.1 system to a maximum of 2^{32} or 4.2 billion records. The fixed record limit prevented you from addressing the maximum amount of DASD that could be attached to a TPF subsystem. FARF6 is the exploitation of 7 of the 8 bytes in the file address field, which expands addressing capacity to a maximum of 64 petabytes, or PB (64 PB equals 72 057 594 037 927 936 records or 2^{56}).

8-byte file address support incorporates two modes of file addressing:

- 4x4 format 4x4 format became available with TPF data event control block (DECB) support (APAR PJ27393). 4x4 format is the migration mode that allows your 4-byte file addresses to become 8-byte file addresses. 4x4 format provides for standard 4-byte file addresses (FARF3, FARF4, and FARF5) to be stored in an 8-byte field. A 4-byte file address in 4x4 format resides in the low-order 4 bytes of the 8-byte field. The high-order 4 bytes of the 8-byte field contain an indicator (a fullword of zeros) that classifies it as a valid 4x4 format address.

- FARF6 FARF6 file addresses are only available for 4-K long-term duplicated pools. Fixed records, short-term pools, and 381- and 1055-byte long-term pools will not have FARF6 file addresses. The high-order 4 bytes of a FARF6 file address is a nonzero value. No special mode for FARF6 will exist; FARF6 addresses can be used when the TPF 4.1 system is in either stage FARF3/4 or stage FARF4/5. A record can only be referenced by FARF3 and FARF4, or FARF4 and FARF5, or FARF6. For example, a record can be referenced by FARF3 and FARF4, but not FARF6. Likewise, a record can be referenced by FARF6, but not by FARF3 and FARF4. FARF6 coexists with FARF3, FARF4, and FARF5; it does not replace them.

Figure 1 shows the FARF6 address format:

SPARE	UFT	UFT	FTI	FTI/ORD	FTI/ORD	ORD	ORD
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Figure 1. FARF6 Format

The FARF6 address format has a spare byte reserved for use by IBM in byte 0, which ***must*** be 0, and a fixed 2-byte universal format type (UFT) (bytes 1 and 2). The smallest format type indicator (FTI) size allowed is 8 bits, which allows a 4-byte ordinal number. The largest FTI size is 24 bits with a 2-byte ordinal number. These restrictions are necessary to allow code that manages the file address to use 32-bit instructions.

The following figure show a comparison of all the file address formats:

<table>
<tr><th>Bit</th><th>FARF3 Fixed</th><th>FARF3 Pool</th><th>FARF4</th><th>FARF5</th><th>FARF6</th><th>General File RRN</th></tr>
<tr><td>00</td><td>0=fixed</td><td>1=pool</td><td rowspan="2">Universal Format Type</td><td rowspan="2">Universal Format Type</td><td rowspan="3">Space Reserved for IBM Use Only</td><td rowspan="3">General File Symbolic Module Number</td></tr>
<tr><td>01</td><td rowspan="5">Band Number</td><td>0=LT, 1=ST</td></tr>
<tr><td>02
03
04
05
06
07</td><td rowspan="11">Ordinal Number</td><td rowspan="4">Variable-Size Format Type Indicator</td><td rowspan="4">Variable-Size Format Type Indicator</td></tr>
<tr><td>08
09
10
11
12</td><td rowspan="6">Universal Format Type</td><td rowspan="6">Relative Record Number</td></tr>
<tr><td>13
14
15
16
17
18
19
20
21</td><td rowspan="6">Ordinal Number</td></tr>
<tr><td>22
23
24
25
26
27</td><td rowspan="3">Variable-Size Ordinal Number</td><td rowspan="3">Variable-Size Ordinal Number</td></tr>
<tr><td>28</td><td>1</td></tr>
<tr><td>29</td><td>0 = Simplex
1 = Duplex</td><td>0 = Simplex
1 = Duplex</td><td rowspan="3">Variable-Size Format Type Indicator (8-24 bits in size)</td><td>1</td></tr>
<tr><td>30</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>31</td><td>0 = Small
1 = Large</td><td>0 = Small
1 = Large</td><td>0 = Small
1 = Large</td><td>0 = Small
1 = Large</td></tr>
<tr><td>•
•
•
•
63</td><td colspan="4"></td><td>Variable-Size Ordinal Number (16-32 bits in size)</td><td></td></tr>
</table>

Figure 2. File Address Formats

Does This APAR Provide Any Other Enhancements?
Yes, the following table summarizes all the enhancements provided with 8-byte file address support:

Support	Comments
8-byte standard header (c$std8.h/istd8.mac)	The c$std8.h standard header file has been created for databases that use 8-byte file addresses. The 8-byte standard header reserves space for 8-byte forward and backward chain file address fields. The ISTD8 structure defines the 8-byte standard header.
Application programming interface (API) support	Updates were made to the following C functions: attac_id creec_CREEC cretc_level detac_id rlcha sonic swisc_create TO2_getDirectoryForRnn TO2_recoupCollection tpf_lemic
Macro Support	Updates were made to the following macros: General macros System macros SIP Macros See the following for more information: TPF Migration Guide: Program Update Tapes TPF General Macros TPF System Macros TPF System Generation.
A new 4-K duplicated long-term FARF6 (4D6) pool type has been defined.	4D6 can be coded on a RAMFIL statement. Dispensing 4D6 pools is similar to dispensing existing pool types. A record ID for 4-K duplicated long-term FARF6 pools can be defined in the record ID attribute table (RIAT).
Pool ordinal numbers (PSONs) and counts of available pools have been increased to 8 bytes.	Pool directory fields CY3BON and CY3ORD have been combined into an 8-byte field (CY3XORD). Some commands that accept a record type ordinal number as input now require the input to be a hexadecimal number.

	Some commands that display a record type ordinal number will display it as a hexadecimal number. The display of the count of available pools will continue to be decimal. Because the decimal number can be very large, the format of the display for the ZDFPC command has been changed so that every third digit will have a space as a separator.
Commands that handle file addresses and record type ordinals have been changed.	Both 4-byte and 8-byte file addresses will be accepted as input for commands. Most commands will display only 8-byte file addresses. If the file address is originally 4-bytes, it will be converted to a 4x4 format 8-byte file address and displayed.
Recoup has been updated to handle databases that use either 4- or 8-byte file addresses.	The recoup descriptors have indicators that recognize whether chains are using the 4- or 8-byte standard header and whether embedded file addresses are 4 or 8 bytes. The GROUP and INDEX macros now include a FAT (file address type) parameter to specify either a 4- or 8-byte file address.
TPF collection support (TPFCS) has been updated.	You can now create new collections using the 8-byte file address format. You must enter ZMODE 6 to switch on 8-byte file addressing. See "How Will This Affect TPF Collection Support (TPFCS) Databases?" on page 7 for more information.
FACE table generator (FCTBG) changes	The FCTBG now supports FACE tables (FCTBs) larger than 16 MB. Generalized object file format (GOFF) is required to support FCTBs larger than 16 MB. Support has been added to allow the job control language (JCL) console as well as the report listing to appear in the same file. An optional input card (Path card) can now be included in the load deck portion of the offline loader JCL that is used to run the general file loader (ALDR) and the auxiliary loader (TLDR). This has been done so that an FCTB that has been bound (by using the Program Object Binder) can be loaded. This card will specify the hierarchical file system (HFS) location of the FACE table (FCTB) in program object format. Changes have also been made to the Load FCTB card to specify the HFS location of the FCTB in program object format. See "Are There Any Loading Process Changes?" on page 6 for more information.

Changes to offline procedures	New parameters were added on the RAMFIL and UFTFTI SIP macros that allow you to define FARF6 file addresses.
Fixed file record changes	Several new fixed file records have been added. See TPF Migration Guide: Program Update Tapes for a complete list.
Changes for database reorganization (DBR)	DBR support includes the following: 8-byte file addresses and ordinal numbers during DBR capture and restore The use of DECBs for finding records during the capture phase and filing records during the input phase Capturing the 4D6 pool type 8-byte ordinal numbers in the exception records Ordinal numbers to be handled as hexadecimal values instead of decimal values.
File capture and restore changes	Exception recording and logging: The format of the exception recording and logging tapes will change once SYSTC bit SB8BFAD is set to 1. Before FARF6, the tapes stored the file address at location 4 for 4 bytes and the time-of-day (TOD) and subsystem user (SSU) were located in a trailer following the data block. Once SB8BFAD is set to 1, location 4 in the block for 4 bytes will be set to zero and the trailer is changed. The trailer will be a 32-byte field with the first 8 bytes containing the file address. The traditional TOD and SSU trailer will follow the 32-byte field.
Support for test tools	The following test tools have been updated to process 8-byte file addresses: Program test vehicle (PTV) Real-time trace (RTT) system utility Selective file dump and trace (SFDT).

Do I Have to Install This APAR?

This APAR (PJ28097) must be installed if you want to apply maintenance or small programming enhancements (SPEs) that will be shipped on future PUTs. Use of 8-byte file address support is not required, but we encourage enablement and use of 8-byte file addresses to position your infrastructure to accept new business requirements.

What Benefits Will I See If I Install This APAR?

Here are some benefits that you can expect to see if you install 8-byte file address support:

- 8-byte file address support is necessary if your complex will soon exceed the addressing capacity provided by FARF5 (4 GB in 4 bytes). Because pools use the most file addresses and grow at the fastest rate, a larger file address called FARF6 (8 bytes) will provide you with file address expan sion. A 4-K duplicated FARF6 pool is supported with a full 8-byte file address. This support coexists with the existing FARF3, FARF4, and FARF5 addressing.

- Keeping current with critical maintenance areas that 8-byte file addressing hits will be easier if you install this APAR.

- Consistency with operator commands.

- If you are testing an application that uses 8-byte file addresses, you will no longer need to know if the command input parameter is decimal or hexadecimal.

- The ZRTDM command has been changed to allow you to modify (and display) the pool charac teristics for a specified record ID (RTP0 to RTP9). For example, if a GETFC macro for a specific record ID is defined in the RIAT to be dispensed as a 4DP pool type on DEVA and 4DPs exist on DEVB, the updates to the ZRTDM command will allow easy changes to use 4DPs on DEVB.

- The ZDFAI command has been added to provide information about a specific file address. You can find out the following information for a fixed file address:

  - Record type
  - Record ordinal
  - Record status

  or the following information about a pool address:

  - Pool section
  - Pool ordinal
  - Directory ordinal
  - Directory byte
  - Directory bit
  - Pool status (available, in use, or unknown).

Why Would I Want to Use 8-Byte File Address Support?

8-byte file address support enables programmers to develop applications that require a large number of file addresses. An example of this might be travel agents storing picture IDs in JPEG files. TPFCS collections can now support this example.

Are There Any Impacts to Automated Operations?

Yes, you might need to modify automation. There have been many messages (online and system) that have been added, changed or that are no longer supported. For a complete list of these messages, see *TPF Migration Guide: Program Update Tapes*.

Will This Affect Pool Counts?
Yes, because the display of the count of available pools will continue to be decimal (and that decimal number can be very large), the format of the display for the ZDFPC command has been changed to enhance readability so that every third digit will have a space as a separator.

Are There Any Loading Process Changes?
If you want to load an FCTB that was created in GOFF format, you will need to link it with the Program Object Binder to create a program object that must reside in the hierarchical file system (HFS) under OS/390 UNIX System Services (OS/390 UNIX) on the same system where the offline loader runs. TPFLDR JCL must be updated to include a Path card, and you must specify the path name on the Load FCTB card. The primary reason for creating an FCTB in GOFF format is to support FCTBs larger than 16 MB.

Is There Anything I Need to Know about the APIs That Were Changed?
Applications that call the following functions (using 8-byte file addresses or DECBs instead of ECB data levels) must be compiled with the C++ compiler:

- `attac_id`
- `creec, __CREEC`
- `cretc_level, __CRETCL`
- `detac_id`
- `rlcha`

See the *TPF C/C++ Language Support User's Guide* for more information about these functions.

Were There Any Database Changes? If TPF Internet mail server support is installed in your complex, recoup descriptors BKD1 and BKDY must be moved into their current slots before you run recoup for the first time after installing 8-byte file address support. To move BKD1 and BKDY, enter the ZRBKD command and specify the MOVE parameter.

How Will This Affect TPF Collection Support (TPFCS) Databases? For TPFCS, migration means allowing new collections to use 8-byte file address formats and eventually FARF6 pool addresses. If TPFCS is installed on your TPF 4.1 system, follow these steps to migrate collections from 4-byte file addresses to 8-byte file addresses:

1. Add 8-byte file address support to your TPF 4.1 system by following the steps listed in "Before You Begin Migrating to 8-Byte File Address Support."
2. When the installation is completed, existing collections that use 4-byte file addresses will continue to be processed as normal. New collections will still be created using 4-byte addresses.
3. Create new collections using 8-byte file address format by entering **ZMODE 6** to set on the SB8BFAD SYSTC switch (do not do this unless 8-byte file address support has been loaded to all processors). Once this switch is set on, all new collections will be built with the 8-byte file address format.
4. Do a pool reallocation to add 4D6 pools. See *TPF Operations* for more information about the ZPOOL GENERATION and ZPOOL INIT commands. See *TPF Database Reference* for more information about pool reallocation procedures.

5. Convert to FARF6 pools by using the RIAT characteristics associated with the TPFCS record IDs. 4-byte collections use RIAT RTP=0 and 8-byte collections use RIAT RTP=1. Change to the RIAT RTP=1 definition to indicate the FARF6 pool type.

6. Migrate existing collections that use 4-byte file addresses to use 8-byte file addresses by entering the ZOODB MIGRATE command for system collection data stores and the ZBROW COLLECTION command with the MIGRATE parameter specified for indi vidual collections. For more information about these commands, see *TPF Operations*.

Are There Any SIP Changes?

The following steps will enable you to incorporate the changes that were made to SIP and system generation for 8-byte file address support:

1. Update the SIP Stage 1 deck to include the new RAMEND and UFTEND macros. You must include these macros even if you do not plan to use FARF6 file addresses. The RAMEND macro must be placed after the last RAMFIL macro in the Stage 1 deck. The UFTEND macro must be placed after the UFTFTI macro. These macros allow the assem bler to understand the RAMFIL and UFTFTI macros, which might have coded parameters that exceed 255 characters.

2. Update the JCL deck (for compiling and link-editing the FCTBG program) to compile and link-edit segment FTTD01. See *TPF System Generation* for more information about SIP and system generation changes.

How Do I Begin the Migration to 8-Byte File Address Support?

Well, before you actually begin the migration, you have to complete the following steps:

1. Use the OS/390 C/C^{++} compiler version 2.6 or higher to compile the FCTB compile.

2. Update the recoup descriptors using the GROUP and INDEX macros only if you are converting databases to use 8-byte file addresses. The GROUP and INDEX macros now include a FAT (file address type) parameter that specifies a 4- or 8-byte file address. See *TPF System Macros* for more information.

3. Reassemble all E-type programs that issue the ESFAC macro because changes were made to the DCTSON data macro. Once you have completed these steps, we suggest that you refer to *TPF Migration Guide: Program Update Tapes* for step by step information about migrating to 8-byte file address support.

Can You Provide Some Migration Examples?

Yes, the following examples are intended to help you migrate existing and dynamic databases to 8-byte file address support. However, you might have to modify the steps in the procedures for the databases you are migrating.

Migrating Existing Databases - an Example

Use the following example as a guideline to help you convert existing traditional TPF databases (non-TPFDF, noncollection) to 8-byte file address support:

1. Identify specific databases that you want to migrate to FARF6. Include the following:
   - All records in a specific database
   - All programs that access these records.

2. Update programs to use DECBs for all FIND, FILE, GETFC, and RELFC macros that access the database you are migrating.

3. Migrate the database to use 8-byte file addresses by doing the following: v. Change the database structures (or DSECTs) to reserve space for the 8-byte standard header. Save location X'10' - X'1F' for the 8-byte forward chain and backward chain. This will require you to move back your data by X'10' bytes.
   a. Change any fields in the structure that contain file addresses (embedded file addresses) to 8 bytes in length.
   b. Change the pool records to 4 K if the record was a 381- or 1055-byte pool record; if possible, use all of the 4 K (pack the records).
   c. Determine how to implement changes to the database and the programs that access the database. For example, to control access, you can use a single indicator for static data or multiple indicators for dynamic data (which is more complex). You may also need to use a utility to move the data.
   d. Make changes to application programs by using the new data structure and modified macros such as RLCHA and FAC8C.
   e. Load the program changes.
   f. Move the location of the data to make room for 8-byte file addresses.
   g. Update recoup descriptors to indicate that this database is now using 8-byte file addresses.
   h. Define FARF6 pools and do a pool reallocation.
   i. Modify the RIAT entry for the record IDs in the database to use FARF6 pools.

Migrating a Dynamic Database —An Example

Migrating a dynamic database is complex because each TPF database is unique and must be investigated individually. Use the following example as a guideline to help you convert dynamic databases to 8-byte file address support:

1. Define the record chain level indicator. This could be located in the header of each record, perhaps as a different record code check (RCC). The following are the possible settings:
   - Use the 4-byte standard header (the current database should have this set)
   - Use both the 4- and 8-byte standard header
   - Use the 8-byte standard header.

2. Application programs interrogate the record chain indicator when:
   - Accessing the data:
     - When the indicator states to use 4-byte, the old data structure will be used.
     - When the indicator states to use both or to use 8-byte, the new data struc ture will be used.

   - Setting forward or backward chains:
     - When the indicator states to use 4-byte, fill in only the 4-byte standard header. The file address must only be 4 bytes.
     - When the indicator states to use both, fill in both the 4- and 8-byte stan dard header. The file address must be in 4x4 format.
     - When the indicator states to use 8-byte, fill in only the 8-byte standard header. The file address can be 4x4 or FARF6 format.

   - Release chains:
     - When the indicator states to use 4-byte, call RLCHA with TYPE=4 (this is the default).
     - When the indicator states to use both 4- or 8-byte, call RLCHA with TYPE=8.

3. Develop a move utility to copy data from the current location to the new location in the new format:
   - Code the utility (if possible) so that the move utility can run while the system is in NORM state and taking traffic.
   - Migrate embedded file addresses and their chains first; start at the lowest level and work up.
   - Turn on the indicator in each record that is migrated, specifying both 4- and 8-byte.
   - Run a utility that will turn on the 8-byte indicator without copying the records (once all of the database is converted and verified).

4. Include additional logic in application programs:

   - If the head of chain indicator states to use both or to use 8-byte, new chains will then use the new 8-byte data structure.
   - If the head of chain indicator states to use both and a GETFC macro returns a FARF6 address (not 4x4 format), turn on the 8-byte indicator and do not fill in the 4-byte standard header.

Are There Any Fallback and Coexistence Issues?

When using 8-byte file address support, consider the following fallback and coexistence information:

- TPFCS will continue to support all collections that have been created using the 4-byte file address format. New collections will be created using 8-byte file address support. Do not activate 8-byte file address support until all processors in the

loosely coupled complex have been upgraded with the new code. Once there is no possibility of fallback, the SB8BFAD SYSTC switch is set to 1 so that new collections can be created using 8-byte file address format. Fallback at this point would mean that the newly created collections would be lost because previous PUTs did not support 8-byte file address format.

- Before 8-byte file address support, the pool monitor had one monitor value per pool type, which it used to compare for each of the pool sections. With 8-byte file address support, you can change the monitor value for each pool section instead of allowing just one value for all four sections (DEVA–DEVD). Therefore, LLT DEVA will have a value that the pool monitor will use and LLT DEVB will have a separate value. The ability to update the monitor values for each pool section is provided by the ZGFSP ALTER command. If these values are not updated by the ZGFSP ALTER command, the pool monitor will initialize the pool section values from the previous pool type values. For example, if the minimum available value for LLT pool type is 1 000 000, FARF6 pool monitor logic will set the minimum available value for LLT DEVA to 1 000 000, for LLT DEVB to 1 000 000, for LLT DEVC to 1 000 000, and for LLT DEVD to 1 000 000.

- The format of the exception recording and logging tapes will change once SYSTC bit SB8BFAD is set to 1. Before 8-byte file address support, the tapes stored the file address at location 4 for 4 bytes and the time-of-day (TOD) and subsystem user (SSU) were located in a trailer following the data block. Once SB8BFAD is set to 1, location 4 in the block for 4 bytes will be set to 0 and the trailer is changed. The trailer will be a 32-byte field with the first 8 bytes containing the file address. The traditional TOD and SSU trailer will follow the 32-byte field.

- The offline loader (TPFLDR) is entirely backward compatible. Unchanged JCL that is used to run TPFLDR will run the same way by loading the FCTB as a load module from the data sets specified on the ALDRCPL DD statement. The only difference in output is a new informational message in the output report indicating that TPFLDR could not attempt to load the FCTB as a program object. Whether the FCTB is loaded in program object format or load module format, there is no change to the contents of the output created by TPFLDR. Therefore, there are no changes to online loader processing.

- If your complex has been using continuous data collection (CDC), you will need to migrate the remote database to add additional columns for 4D6 pools. To migrate the remote database, enter the ZCDCO command and specify the MI GRATE parameter. Once the database has been migrated, it will still be compat ible with PUT 14 if fallback is necessary. Any remote database created with CDC after 8-byte file address support (APAR PJ28097) is applied will not be compat ible with any prior PUT. The latest version of the CDC graphical user interface (GUI) will work with any database. For example, it will work with one created

before APAR PJ28097 is applied, with one migrated to APAR PJ28097, and with one created after APAR PJ28097 is applied.

- The #PSTCUR and #PSTNEW fixed file records that are built during the pool generation process and describe the pool configuration on the TPF 4.1 system are being converted to the expanded #PSTXCUR and #PSTXNEW records respec tively. The #PSTCUR and #PSTNEW fixed file records are built during the first IPL after PUT 15 is applied to your TPF 4.1 system. These records are initialized from the FCTB in the same way that they are initialized if you enter ZPOOL GENERATION INIT.

Notes:

1. You do not need to enter ZPOOL GENERATION INIT unless a new pool generation needs to be done.

2. The #PSTXNEW and #PSTXCUR fixed file records must be allocated before migration.

3. You cannot make changes in the pool configuration at the same time as the 8-byte file address support migration. Pool definitions in the FCTB must match the current pool configuration. You must not have pools defined in the FCTB for a future pool reallocation at implementation time.

4. If migration to 8-byte file address support is completed, fallback and a pool reallocation is com pleted. Before migrating again, reset the #PSTXCUR and #PSTXNEW fixed file records to a record ID of X’0000' to ensure they will be initialized correctly.

- When a file fails to both the prime and duplicate copy, routine CJPXEMTR in CJIP gains con-trol. An ECB is created and segment CPSR is invoked. If the SB8BFAD SYSTC bit is switched on, the 8-byte file address will always be stored at EBW000. If the entry in field CE1FMO for 4 bytes is 0, the file address will be at EBW000 for 8 bytes. If the entry in field CE1FMO is not 0, the entry is the file address. You might have to update your EPC1 program to load an 8-byte file address from the ECB work area if you intend to use FARF6 addresses.
- Several TPF and TPFDF databases have been replaced. See *TPF System Generation* for more information.

- The BKDY and BKD1 recoup descriptors must be moved before you run recoup for the first time. For more information about moving BKDY and BKD1, see the ZRBKD command in *TPF Operations.*

- The first recoup run following the installation of 8-byte file address support may report lost addresses for the following record IDs:

FC32 Caused by changing ordinal number #E80E8 to #E88E8 for the erroneously available record.
FC35 Caused by changing ordinal number #L80L8 to #L88L8 for the lost address record.
CD Caused by changing record type #SONSP to #SONSKP.

Where Can I Find More Information?

Publication Title	Type of Information
TPF Application Programming	Conceptual information about 8-byte file address support.
TPF C/C++ Language Support User's Guide	Updated C functions.
TPF Concepts and Structures	Conceptual information about 8-byte file address support.
TPF Database Reference	Updates to the following information: • File address formats • File pool support • Database reorganization • Recoup (broken chain report) • TPF collection support (TPFCS) • FACE driver and offline interface (DFAD) restrictions.
TPF General Macros	Changed macros.
TPF Library Guide	New and changed glossary terms.
TPF Messages, Volume 1 and Volume 2	New, changed, and no longer supported messages.
TPF Migration Guide: Program Update Tapes	Migration considerations and scenarios.
TPF Operations	New and changed commands.
TPF Program Development Support Reference	Changes to the real-time trace (RTT) system utility and selective file dump and trace (SFDT).
TPF System Generation	Changes to SIP for 8-byte file address support.
TPF System Installation Support Reference	Information about the new Path card and updates to the Load FCTB card for ALDR and TLDR.
TPF System Performance and Measurement Reference	Updated System Pools Summary report.
TPF System Macros	New and changed system macros.