# GT.M

## Release Notes

V6.2-000

Empowering
the Financial World

## Contact Information

GT.M Group
Fidelity National Information Services, Inc.
200 Campus Drive
Collegeville, PA 19426
United States of America

GT.M Support for customers: gtmsupport@fisglobal.com
Automated attendant for 24 hour support: +1 (484) 302-3248
Switchboard: +1 (484) 302-3160
Website: http://fis-gtm.com

## Legal Notice

Copyright © 2014, 2019 Fidelity Information Services, Inc. All Rights Reserved

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts.

GT.M™ is a trademark of Fidelity Information Services, Inc. Other trademarks are the property of their respective owners.

This document contains a description of GT.M and the operating instructions pertaining to the various functions that comprise the system. This document does not contain any commitment of FIS. FIS believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. FIS is not responsible for any errors or defects.

## Revision History

| | | |
|---|---|---|
| Revision 1.1 | 5 February 2019 | Updated the Platforms section to add AIX 7.1 TL 4 and AIX 7.2 as supported versions. |
| Revision 1.0 | 24 September 2014 | V6.2-000. First published revision. |

# Table of Contents

- V6.2-000 ... 1
  - Overview ... 1
  - Conventions ... 1
  - Platforms ... 3
    - Platform support lifecycle ... 4
  - Migrating to 64-bit platforms ... 5
    - Call-ins and External Calls ... 5
    - Internationalization (Collation) ... 5
    - Environment Translation ... 6
  - Recompile ... 6
  - Rebuild Shared Libraries or Images ... 6
  - Additional Installation Instructions ... 6
    - UNIX ... 6
    - OpenVMS ... 8
  - Upgrading to GT.M V6.2-000 ... 9
    - Stage 1: Global Directory Upgrade ... 9
    - Stage 2: Database Files Upgrade ... 10
    - Stage 3: Replication Instance File Upgrade ... 11
    - Stage 4: Journal Files Upgrade ... 12
    - Stage 5: Trigger Definitions Upgrade ... 12
    - Downgrading to V5 or V4 ... 12
  - Managing M mode and UTF-8 mode ... 13
    - Compiling ICU ... 14
  - Setting the environment variable TERM ... 14
  - Installing Compression Libraries ... 15
- Change History ... 16
  - V6.2-000 ... 16
- M-Database Access ... 22
- M-Other Than Database Access ... 25
- Utilities-MUPIP ... 37
- Utilities-Other Than MUPIP ... 39
- More Information ... 41
  - Additional Information for GTM-7449 - Encrypted File IO Plugin Reference Implementation ... 41
    - Reference Implementation ... 41
    - Encryption Plugin API ... 41
- Error and Other Messages ... 46
  - ACTIVATEFAIL Δ ... 46
  - AIMGBLKFAIL ⊕ ... 46
  - BLKINVALID ⊕ ... 46
  - CANTBITMAP ⊕ ... 46
  - CHSETALREADY ⊕ ... 46
  - COLTRANSSTR2LONG ⊕ ... 46
  - CONNSOCKREQ ⊕ ... 47
  - CORRUPTNODE ⊕ ... 47
  - CREDNOTPASSED ⊕ ... 47
  - CRYPTBADWRTPOS ⊕ ... 47
  - CRYPTKEYTOOBIG ⊕ ... 47
  - CRYPTNOAPPEND ⊕ ... 48
  - CRYPTNOKEYSPEC ⊕ ... 48
  - CRYPTNOOVERRIDE ⊕ ... 48

CRYPTNOSEEK ⊕ 48

CRYPTNOTRUNC ⊕ 48

DBCOLLREQ Δ 48

DSEMAXBLKSAV ⊕ 49

DSENOTOPEN ⊕ 49

GTMDISTUNVERIF ⊕ 49

INSNOTJOINED Δ 49

INVLINKTMPDIR ⊕ 49

INVOBJFILE ⊕ 50

KEYWRDBAD Δ 50

LABELNOTFND ⊕ 50

LDSPANGLOINCMP Δ 50

LITNONGRAPH Δ 50

LOCALSOCKREQ ⊕ 51

MAXBTLEVEL Δ 51

NOGTCMDB ⊕ 51

NOSOCKHANDLE ⊕ 51

NOTGBL Δ 51

NOUSERDB ⊕ 52

PEERPIDMISMATCH ⊕ 52

RECLOAD ⊕ 52

RELINKCTLERR ⊕ 52

REORGINC Δ 52

REPLINSTNOSHM Δ 52

SDSEEKERR ⊕ 53

SEFCTNEEDSFULLB ⊕ 53

SOCKACCEPT ⊕ 53

SOCKNOTPASSED ⊕ 53

SOCKPASS ⊕ 53

SOCKPASSDATAMIX ⊕ 53

TPRESTART Δ 54

TRIGLOADFAIL ⊕ 54

TRIGZBREAKREM Δ 54

ZSOCKETATTR ⊕ 54

ZSOCKETNOTSOCK ⊕ 54

## V6.2-000

## Overview

V6.2-000 is a major release that introduces auto-relink, a publish+subscribe model that enables GT.M processes to automatically and safely execute the latest published versions of routines; adds support to read and write encrypted sequential disk files, pipes, and FIFOs; adds the ability to pass SOCKET devices from one process to another; and includes many other enhancements and fixes.

* Released as field test grade functionality in a production release, auto-relink provides a way to allow long-running processes to automatically and safely execute the latest versions of routines. It also has uses in development and testing. Each process that wishes to subscribe to updates of object files in a directory appends an asterisk ("*") suffix to that directory name in $ZROUTINES before linking from it. When a process publishes updates to routines (including the presence of new routines, and the removal of existing routines) with the ZRUPDATE command, processes that have subscribed to those updates automatically perform a ZLINK on the next execution of a command or function call that explicitly invokes a routine with a published update. Safe execution means that auto-relink of new versions of routines ensures that prior versions of those routines on the M call stack continue executing correctly through their QUIT commands and are not "clobbered" when control returns to them. Field-test grade functionality in a production release means that FIS considers auto-relink functionality sufficiently robust for use in development and testing, but not robust enough for use in production environments. Except for auto-relink, all other functionality in V6.2-000 is robust enough for use in production. See GTM-7380 below.

* To protect non-database data at rest, and more securely exchange data with other systems and applications, both GT.M as well as non-GT.M processes, GT.M encryption functionality extends to file, pipe, and FIFO IO. As with all GT.M encryption features, GT.M itself includes no cryptographic software; instead it accesses external cryptographic libraries using a plugin architecture. The GT.M distribution includes a reference implementation of the plugin, which FIS tests during GT.M development against a couple of widely available cryptographic packages. FIS neither supports cryptographic libraries with GT.M nor recommends the use of any specific library - refer to the GT.M Administration and Operations Guide UNIX Edition for more details. See GTM-7449 below.

* Functionality to transfer DETACHed TCP or LOCAL sockets between GT.M processes using a LOCAL SOCKET device as the rendezvous is a fundamental enabling technology. Rich user interfaces can be built with multiple bi-directional TCP connections between clients such as browsers and server processes, using a layered protocol such as WebSockets. When a client and server wish to add a TCP connection between them, the client can initiate a new connection to a known port on server. An Internet superserver, such as xinetd, can screen the request and start a GT.M process, which can use an application-defined protocol to obtain the process id of the client's existing server process to whom it transfers the connection. The technique can also be used by a server to pass an existing client socket to a different process, for example, one that publishes information, such as a news feed or weather report. See GTM-7320 below.

V6.2-000 also has numerous smaller enhancements, including:

* $ZSOCKET() provides information about a SOCKET device and its attached sockets (GTM-7735).

* $ZSYSLOG() allows a process to send a message to the syslog without requiring the POSIX plugin (GTM-7999).

* GT.M for 32-bit Linux is compiled using the i586 ("Pentium compatible") instruction set, allowing it to run on recent System on Chip (SoC) based computers (GTM-7950).

* MUPIP SIZE includes an estimate of adjacency (GTM-7991).

* $PRINCIPAL can be a LOCAL socket (GTM-8056).

The online help facilities provided through the ZHELP command in GT.M, and the HELP commands in the utilities, provide hierarchically organized information current with V6.2-000. These, as well as robustness improvements, fixes to issues, and performance enhancements, are described below.

## Conventions

This document uses the following conventions:

| | UNIX | OpenVMS |
|---|---|---|
| **Flag/Qualifiers** | - | / |
| **Program Names or Functions** | upper case. For example, MUPIP BACKUP | |
| **Examples** | lower case. For example: mupip backup -database ACN,HIST / backup | |
| **Reference Number** | A reference number is used to track software enhancements and support requests. It is enclosed between parentheses (). | |
| **Platform Identifier** | [UNIX] | [OpenVMS] |
| | The platform identifier does not appear for those new features or enhancements that apply to all platforms. | |

## Note

The term UNIX refers to the general sense of all platforms on which GT.M uses a POSIX API. As of this date, this includes: AIX, HP-UX, GNU/Linux, and Solaris.

The following table summarizes the new and revised replication terminology and qualifiers.

| Pre V5.5-000 terminology | Pre V5.5-000 qualifier | Current terminology | Current qualifiers |
|---|---|---|---|
| originating instance or primary instance | -rootprimary | originating instance or originating primary instance.<br><br>Within the context of a replication connection between two instances, an originating instance is referred to as source instance or source side. For example, in an B<-A->C replication configuration, A is the source instance for B and C. | -updok (recommended)<br><br>-rootprimary (still accepted) |
| replicating instance (or secondary instance) and propagating instance | N/A for replicating instance or secondary instance.<br><br>-propagateprimary for propagating instance | replicating instance.<br><br>Within the context of a replication connection between two instances, a replicating instance that receives updates from a source instance is referred to as receiving instance or receiver side. For example, in an B<-A->C replication configuration, both B and C can be referred to as a receiving instance. | -updnotok |
| N/A | N/A | supplementary instance.<br><br>For example, in an A->P->Q replication configuration, P is the supplementary instance. Both A and P are originating instances. | -updok |

Effective V6.0-000, GT.M documentation is adopting IEC standard Prefixes for binary multiples. This document therefore uses prefixes Ki, Mi and Ti (e.g., 1MiB for 1,048,576 bytes). All GT.M documentation will over time be updated to this standard.

✓ denotes a new feature that requires updating the manuals.

✓ denotes a new feature or an enhancement that may not be upward compatible and may affect an existing application.

⛔ denotes deprecated messages.

⚠ denotes revised messages.

⊕ denotes added messages.

## Platforms

Over time, computing platforms evolve. Vendors obsolete hardware architectures. New versions of operating systems replace old ones. We at FIS continually evaluate platforms and versions of platforms that should be Supported for GT.M. In the table below, we document not only the ones that are currently Supported for this release, but also alert you to our future plans given the evolution of computing platforms. If you are an FIS customer, and these plans would cause you hardship, please contact your FIS account executive promptly to discuss your needs.

GT.M runs on a wide variety of UNIX/Linux implementations as well as OpenVMS. Consult FIS for currently supported versions. Each GT.M release is extensively tested by FIS on a set of specific versions of operating systems on specific hardware architectures (the combination of operating system and hardware architecture is referred to as a platform). This set of specific versions is considered Supported. There will be other versions of the same operating systems on which a GT.M release may not have been tested, but on which the FIS GT.M support team knows of no reason why GT.M would not work. This larger set of versions is considered Supportable. There is an even larger set of platforms on which GT.M may well run satisfactorily, but where the FIS GT.M team lacks the knowledge to determine whether GT.M is Supportable. These are considered Unsupported. Contact FIS GT.M Support with inquiries about your preferred platform.

As of the publication date, FIS supports this release on the hardware and operating system versions below. Contact FIS for a current list of Supported platforms. The reference implementation of the encryption plugin has its own additional requirements, should you opt to use it as included with GT.M.

| Platform | Supported Versions | Notes |
|---|---|---|
| Hewlett-Packard Integrity IA64<br><br>HP-UX | 11V3 (11.31) | Contact your FIS support channel if you wish to use GT.M on this platform. |
| Hewlett-Packard Alpha/AXP OpenVMS | 7.2-1 / 8.2 / 8.3 | *FIS intends to no longer support this platform after December 31, 2014. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.*<br><br>GT.M supports M mode but not UTF-8 mode on this platform. GT.M does not support several recent enhancements on this platform, including but not limited to database encryption, on-line backup, multi-site replication, PIPE devices and triggers.<br><br>If you need to work with external calls written in C with Version 6.x of the C compiler on Alpha OpenVMS, then you must carefully review all the provided kits for that product and apply them appropriately.<br><br>Although this platform remains at present fully supported with respect to bug fixes, new functionality is supported on this platform only for FIS' convenience. |

| Platform | Supported Versions | Notes |
|---|---|---|
| IBM System p AIX | 6.1, 7.1 TL 4, 7.2 | Since GT.M processes are 64-bit, FIS expects 64-bit AIX configurations to be preferable.<br><br>While GT.M supports both UTF-8 mode and M mode on this platform, there are problems with the AIX ICU utilities that prevent FIS from testing 4-byte UTF-8 characters as comprehensively on this platform as we do on others.<br><br>AIX 7.1 TL 5 is Supportable.<br><br>Running GT.M on AIX 7.1 requires APAR IZ87564, a fix for the POW() function, to be applied. To verify that this fix has been installed, execute **instfix -ik IZ87564.** |
| Oracle (Sun) SPARC Solaris | 10 (Update 6 and above) | Contact your FIS support channel if you wish to use GT.M on this platform. |
| x86_64 GNU/Linux | Red Hat Enterprise Linux 6; Ubuntu 12.04 LTS; SuSE Linux Enterprise Server 11 | To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware.<br><br>GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.5-24 or later) and ncurses (version 5.5 or later).<br><br>To install GT.M with Unicode (UTF-8) support on RHEL 6, in response to the installation question **Should an ICU version other than the default be used? (y or n)** please respond **y** and then specify the ICU version (for example, respond 3.6) to the subsequent prompt **Enter ICU version (ICU version 3.6 or later required. Enter as major-ver.minor-ver):**<br><br>GT.M requires the libtinfo library. If it is not already installed on your system, and is available using the package manager, install it using the package manager. If a libtinfo package is not available (for example on SuSE 11):<br><br>* Find the directory where libncurses.so is installed on your system.<br><br>* Change to that directory and make a symbolic link to libncurses.so.<ver> from libtinfo.so.<ver>. Note that some of the libncurses.so entries may themselves be symbolic links, for example, libncurses.so.5 may itself be a symbolic link to libncurses.so.5.9.<br><br>GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.5-49 or later) and ncurses (version 5.5-24 or later). |
| x86 GNU/Linux | Red Hat Enterprise Linux 6 | This 32-bit version of GT.M runs on either 32- or 64-bit x86 platforms; we expect the X86_64 GNU/Linux version of GT.M to be preferable on 64-bit hardware. The CPU must have an instruction set equivalent to 586 (Pentium) or better. Also, refer to the notes above on the 64-bit version. |

## Platform support lifecycle

FIS usually supports new operating system versions six months or so after stable releases are available and we usually support each version for a two year window. GT.M releases are also normally supported for two years after release. While FIS will attempt to provide support to customers in good standing for any GT.M release and operating system version, our ability to provide support is diminished after the two year window.

GT.M cannot be patched, and bugs are only fixed in new releases of software.

## Migrating to 64-bit platforms

The same application code runs on both 32-bit and 64-bit platforms. Please note that:

* You must compile the application code separately for each platform. Even though the M source code is exactly the same, the generated object modules are different even on the same hardware architecture - the object code differs between x86 and x86_64.

* Parameter-types that interface GT.M with non-M code using C calling conventions must match the data-types on their target platforms. Mostly, these parameters are for call-ins, external calls, internationalization (collation) and environment translation, and are listed in the tables below. Note that most addresses on 64-bit platforms are 8 bytes long and require 8 byte alignment in structures whereas all addresses on 32-bit platforms are 4 bytes long and require 4-byte alignment in structures.

### Call-ins and External Calls

| Parameter type | 32-Bit | 64-bit | Remarks |
|---|---|---|---|
| gtm_long_t | 4-byte (32-bit) | 8-byte (64-bit) | gtm_long_t is much the same as the C language long type. |
| gtm_ulong_t | 4-byte | 8-byte | gtm_ulong_t is much the same as the C language unsigned long type. |
| gtm_int_t | 4-byte | 4-byte | gtm_int_t has 32-bit length on all platforms. |
| gtm_uint_t | 4-byte | 4-byte | gtm_uint_t has 32-bit length on all platforms |

**Caution**

If your interface uses gtm_long_t or gtm_ulong_t types but your interface code uses int or signed int types, failure to revise the types so they match on a 64-bit platform will cause the code to fail in unpleasant, potentially dangerous and hard to diagnose ways.

### Internationalization (Collation)

| Parameter type | 32-Bit | 64-bit | Remarks |
|---|---|---|---|
| gtm_descriptor in gtm_descript.h | 4-byte | 8-byte | Although it is only the address within these types that changes, the structures may grow by up to 8 bytes as a result of compiler padding to meet platform alignment requirements. |

**Important**

Assuming other aspects of code are 64-bit capable, collation routines should require only recompilation.

## Environment Translation

| Parameter type | 32-Bit | 64-bit | Remarks |
|---|---|---|---|
| gtm_string_t type in gtmxc_types.h | 4-byte | 8-byte | Although it is only the address within these types that changes, the structures may grow by up to 8 bytes as a result of compiler padding to meet platform alignment requirements. |

### Important

Assuming other aspects of code are 64-bit capable, environment translation routines should require only recompilation.

## Recompile

* Recompile all M and C source files.

## Rebuild Shared Libraries or Images

* Rebuild all Shared Libraries (UNIX) or Shareable Executable Images (OpenVMS) after recompiling all M and C source files..

## Additional Installation Instructions

To install GT.M, see the "Installing GT.M" section in the GT.M Administration and Operations Guide. For minimal down time, upgrade a current replicating instance and restart replication. Once that replicating instance is current, switch it to become the originating instance. Upgrade the prior originating instance to become a replicating instance, and perform a switchover when you want it again to resume an originating primary role.

### UNIX

* FIS strongly recommends installing each version of GT.M in a separate (new) directory, rather than overwriting a previously installed version. If you have a legitimate need to overwrite an existing GT.M installation with a new version, you must first shut down all processes using the old version. FIS suggests installing GT.M V6.2-000 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.2-000_arch (for example, /usr/lib/fis-gtm/V6.2-000_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.2-000_arch would also be appropriate. Note that the ***arch*** suffix is especially important if you plan to install 32- and 64-bit versions of the same release of GT.M on the same system.

* Use the MUPIP RUNDOWN command of the old GT.M version to ensure all database files are cleanly closed.

* In UNIX editions, make sure gtmsecshr is not running. If gtmsecshr is running, first stop all GT.M processes including the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP *pid_of_gtmsecshr***.

* Starting with V6.2-000, GT.M no longer supports the use of the deprecated $gtm_dbkeys and the master key file it points to for database encryption. To convert master files to the libconfig format, please click ▼ to download the CONVDBKEYS.m program and follow instructions in the comments near the top of the program file. You can also download CONVDBKEYS.m from http://tinco.pair.com/bhaskar/gtm/doc/articles/downloadables/CONVDBKEYS.m. If you are using $gtm_dbkeys for database encryption, please convert master key files to libconfig format immediately after upgrading to V6.2-000. Also, modify your environment scripts to include the use of gtmcrypt_config environment variable.

### Caution

Never replace the binary image on disk of any executable file while it is in use by an active process. It may lead to unpredictable results. Depending on the operating system, these results include but are not limited to denial of service (that is, system lockup) and damage to files that these processes have open (that is, database structural damage).

## Reference Implementation Plugin - Shared Library Dependencies

The database encryption and TLS reference implementation plugin binaries requires libgcrypt11, libgpgme11, libconfig 1.4.x, and libssl1.0.0 runtime libraries. Immediately after installing GT.M, install these libraries using the package manager of your operating system. If these runtime libraries are not available, the plugin may produce errors like the following (for example on Ubuntu x86_64):

```
%GTM-I-TEXT, libxxxxxx.so.<ver>: cannot open shared object file: No such file or directory
```

..where **libxxxxxx** is either libgcrypt, liggpgme, libconfig, libgcrypt, or libssl and **<ver>** is the version number of the libxxxxxx shared library. To locate missing share library dependencies in your reference implementation plugin, execute the following command:

```
$ ldd $gtm_dist/plugin/libgtm* | awk '/^.usr/{libname=$0}/not found/{print libname;print}'
```

This command produces an output like:

```
/usr/lib/fis-gtm/V6.1-000_x86_64/plugin/libgtmcrypt_openssl_AES256CFB.so:
        libcrypto.so.10 => not found
/usr/lib/fis-gtm/V6.1-000_x86_64/plugin/libgtmcrypt_openssl_BLOWFISHCFB.so:
        libcrypto.so.10 => not found
/usr/lib/fis-gtm/V6.1-000_x86_64/plugin/libgtmtls.so:
        libssl.so.10 => not found
```

...which means that libcrypto.so.10 and libssl.so.10 shared libraries are missing from the system and the reference implementation plugin will produce a libxxxxxx.so.<ver>: cannot open shared object file: No such file or directory error during runtime. Note down the missing shared libraries and proceed as follows for each missing library:

* **If libxxxxxx.so.<ver> is already installed:** Execute the following command to create a symlink from **libxxxxxx.so.<ver>** to $gtm_dist/plugin:

  ```
  ln -s /path/to/installed/libxxxxxx.so.<ver> $gtm_dist/plugin
  ```

  ### Note

  The reference implementation plugin libraries are linked using the RPATH directive $ORIGIN which means that these libraries always start looking for library dependencies from the current directory. Therefore, it is safe to create symlinks for the missing libraries in $gtm_dist/plugin directory. Do not create symlinks to shared libraries in directories managed by package managers and never symlink incompatible library versions as it may not always lead to predictable results.

* **If libxxxxxx is not installed**: Install the runtime libxxxxxx library using the package manager; check whether the pre-packaged distributions of your operating system contains the library having version specified with <ver>. On older Linux systems, these libraries may not be available so you have to build the missing libraries from source. After installing/building the missing library, create a symlink from **libxxxxxx.so.<ver>** to $gtm_dist/plugin:

  ```
  ln -s /path/to/installed/libxxxxxx.so.<ver> $gtm_dist/plugin
  ```

* As a special case, some systems may have libssl.so.1.0.x and libcrypto.so.1.0.x instead of libssl.so.10 and libcrypto.so.1.0.x. In this case it is safe to link them using:

  ```
  ln -s /path/to/installed/libxxxxxx.so.1.0.x $gtm_dist/plugin/libxxxxxx.so.10
  ```

* Alternatively, you can recompile the reference implementation plugin. The instructions are as follows:

### Recompiling the Reference Implementation Plugin

1. Install the development libraries for libgcrypt11, libgpgme11, libconfig 1.4.x or 1.3.x, libssl1.0.0. The package name of development libraries usually have the -dev or -devel suffix and are available through the package manager. For example, on Ubuntu_x86_64 a command like the following installs the required development libraries:

   ```
   sudo apt-get install libgcrypt11-dev libgpgme11-dev libconfig-dev libssl-dev
   ```

2. Login as root and set the gtm_dist environment variable to point to the location of the GT.M distribution.
3. Note down the file permissions of the *.so files in $gtm_dist/plugin directory.
4. Unpack $gtm_dist/plugin/gtmcrypt/source.tar to a temporary directory.

   ```
   mkdir /tmp/plugin-build
   cd /tmp/plugin-build
   cp $gtm_dist/plugin/gtmcrypt/source.tar .
   tar -xvf source.tar
   ```

5. Recompile the reference implementation plugin and install it in the $gtm_dist/plugin directory.

   ```
   make uninstall
   make
   make install
   make clean
   ```

6. Assign permissions noted in step 3 to the newly installed .so files in $gtm_dist/plugin.

#### Important

If you are running the V6.1-000 or V6.2-000 reference implementation plugin (database encryption and TLS replication) on a UNIX platform other than RedHat, please create a symlinks (**ln -s /path/to/installed/lib{ssl,crypto}.so.1.0.x $gtm_dist/plugin/lib{ssl,crypto}.so.10**) for the missing libcrypto.so.10 and libssl.so.10 shared libraries. GT.M UNIX distributions are compiled and linked on a RedHat platform where libcrypto.so.10 and libssl.so.10 version libraries are available from package managers. On UNIX platforms other than RedHat or older Linux systems, these library versions may not be available from package managers. Alternatively, you can recompile the reference implementation plugin.

### Additional Information for JFS1 on AIX

If you expect a database file or journal file to exceed 2GB with older versions of the JFS file system, then you must configure its file system to permit files larger than 2GB. Furthermore, should you choose to place journal files on file systems with a 2GB limit, since GT.M journal files can grow to a maximum size of 4GB, you must then set the journal auto switch limit to less than 2 GB.

## OpenVMS

To upgrade from a GT.M version prior to V4.3-001, you must update any customized copy of **GTM$DEFAULTS** to include a definition for **GTM$ZDATE_FORM**.

You can ignore the following section if you choose the standard GT.M configuration or answer yes to the following question:

```
Do you want to define GT.M commands to the system
```

If you define GT.M commands locally with **SET COMMAND GTM$DIST:GTMCOMMANDS.CLD** in **GTMLOGIN.COM** or other command file for each process which uses GT.M, you must execute the same command after installing the new version of GT.M and before using it. If you define the GT.M commands to the system other than during the installation of GT.M, you must update the system DCLTABLES with the new GTMCOMMANDS.CLD provided with this version of GT.M. See the OpenVMS "Command Definition, Librarian, and Message Utilities Manual" section on "Adding a system command." In both cases, all GT.M processes must match the proper **GTMCOMMANDS.CLD** with the version of GT.M they run..

## Upgrading to GT.M V6.2-000

The GT.M database consists of four types of components- database files, journal files, global directories, and replication instance files. The format of some database components is different for 32-bit and 64-bit GT.M releases for the x86 GNU/Linux platform.

GT.M upgrade procedure for V6.2-000 consists of 5 stages:

* Stage 1: Global Directory Upgrade
* Stage 2: Database Files Upgrade
* Stage 3: Replication Instance File Upgrade
* Stage 4: Journal Files Upgrade
* Stage 5: Trigger Definitions Upgrade

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V6.2-000 depends on your GT.M upgrade history and your current version.

## Stage 1: Global Directory Upgrade

FIS strongly recommends you back up your Global Directory before upgrading. There is no single-step method for downgrading a Global Directory to an older format.

**To upgrade from any previous version of GT.M:**

* Open your Global Directory with the GDE utility program of GT.M V6.2-000.
* Execute the EXIT command. This command automatically upgrades the Global Directory.

**To switch between 32- and 64-bit global directories on the x86 GNU/Linux platform:**

1. Open your Global Directory with the GDE utility program on the 32-bit platform.
2. On GT.M versions that support SHOW -COMMAND, execute SHOW -COMMAND -FILE=file-name. This command stores the current Global Directory settings in file-name.
3. On GT.M versions that do not support GDE SHOW -COMMAND, execute the SHOW -ALL command. Use the information from the output to create an appropriate command file or use it as a guide to manually enter commands in GDE.
4. Open GDE on the 64-bit platform. If you have a command file from 2. or 3., execute @file-name and then run the EXIT command. These commands automatically create the Global Directory. Otherwise use the GDE output from the old Global Directory and apply the settings in the new environment.

An analogous procedure applies in the reverse direction.

If you inadvertently open a Global Directory of an old format with no intention of upgrading it, execute the QUIT command rather than the EXIT command.

If you inadvertently upgrade a global directory, perform the following steps to downgrade to an old GT.M release:

* Open the global directory with the GDE utility program of V6.2-000.

* Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

## Stage 2: Database Files Upgrade

**To upgrade from GT.M V5.0*/V5.1*/V5.2*/V5.3*/V5.4*/V5.5:**

A V6 database file is a superset of a V5 database file and has potentially longer keys and records. Therefore, upgrading a database file requires no explicit procedure. After upgrading the Global Directory, opening a V5 database with a V6 process automatically upgrades fields in the database fileheader.

A V6 database supports global variable nodes up to 1 MiB and is not backward compatible. Use MUPIP DOWNGRADE -VERSION=V5 to downgrade a V6 database back to V5 format provided it meets the database downgrade requirements. For more information on downgrading a database, refer to  Downgrading to V5 or V4.

### Important

A V5 database that has been automatically upgraded to V6 can perform all GT.M V6.2-000 operations. However, that database can only grow to the maximum size of the version in which it was originally created. If the database is V5.0-000 through V5.3-003, the maximum size is 128Mi blocks. If the database is V5.4-000 through V5.5-000, the maximum size is 224Mi blocks. Only a database created with V6.0-000 or above (with a V6 MUPIP CREATE) can have a maximum database size of 992Mi blocks.

If your database has any previously used but free blocks from an earlier upgrade cycle (V4 to V5), you may need to execute the MUPIP REORG -UPGRADE command. If you have already executed the MUPIP REORG -UPGRADE command in a version prior to V5.3-003 and if subsequent versions cannot determine whether MUPIP REORG -UPGRADE performed all required actions, it sends warnings to the operator log requesting another run of MUPIP REORG -UPGRADE. In that case, perform any one of the following steps:

* Execute the MUPIP REORG -UPGRADE command again, or

* Execute the DSE CHANGE -FILEHEADER -FULLY_UPGRADED=1 command to stop the warnings.

### Caution

Do not run the DSE CHANGE -FILEHEADER -FULLY_UPGRADED=1 command unless you are absolutely sure of having previously run a MUPIP REORG -UPGRADE from V5.3-003 or later. An inappropriate DSE CHANGE -FILEHEADE -FULLY_UPGRADED=1 may lead to database integrity issues.

You do not need to run MUPIP REORG -UPGRADE on:

* A database that was created by a V5 MUPIP CREATE

* A database that has been completely processed by a MUPIP REORG -UPGRADE from V5.3-003 or later.

For additional upgrade considerations, refer to Database Compatibility Notes.

**To upgrade from a GT.M version prior to V5.000:**

You need to upgrade your database files only when there is a block format upgrade from V4 to V5. However, some versions, for example, database files which have been initially been created with V4 (and subsequently upgraded to a V5 format) may additionally need a MUPIP REORG -UPGRADE operation to upgrade previously used but free blocks that may have been missed by earlier upgrade tools.

* Upgrade your database files using in-place or traditional database upgrade procedure depending on your situation. For more information on in-place/traditional database upgrade, see Database Migration Technical Bulletin.

* Run the MUPIP REORG -UPGRADE command. This command upgrades all V4 blocks to V5 format.

### Note

Databases created with GT.M releases prior to V5.0-000 and upgraded to a V5 format retain the maximum size limit of 64Mi (67,108,864) blocks.

## Database Compatibility Notes

* Changes to the database file header may occur in any release. GT.M automatically upgrades database file headers as needed. Any changes to database file headers are upward and downward compatible within a major database release number, that is, although processes from only one GT.M release can access a database file at any given time, processes running different GT.M releases with the same major release number can access a database file at different times.

* Databases created with V5.3-004 through V5.5-000 can grow to a maximum size of 224Mi (234,881,024) blocks. This means, for example, that with an 8KiB block size, the maximum database file size is 1,792GiB; this is effectively the size of a single global variable that has a region to itself; a database consists of any number of global variables. A database created with GT.M versions V5.0-000 through V5.3-003 can be upgraded with MUPIP UPGRADE to increase the limit on database file size from 128Mi to 224Mi blocks.

* Databases created with V5.0-000 through V5.3-003 have a maximum size of 128Mi (134, 217,728) blocks. GT.M versions V5.0-000 through V5.3-003 can access databases created with V5.3-004 and later as long as they remain within a 128Mi block limit.

* Database created with V6.0-000 or above have a maximum size of 1,040,187,392(992Mi) blocks.

* For information on downgrading a database upgraded from V6 to V5, refer to: Downgrading to V5 or V4.

## Stage 3: Replication Instance File Upgrade

V6.2-000 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.2-000 requires new replication instance files if you are upgrading from any version prior to V5.5-000. Instructions for creating new replication instance files are in the Database Replication chapter of the UNIX Administration and Operations Guide. Shut down all Receiver Servers on other instances that are to receive updates from this instance, shut down this instance Source Server(s), recreate the instance file, restart the Source Server(s) and then restart any Receiver Server for this instance with the -UPDATERESYNC qualifier.

### Note

Without the UPDATERESYNC qualifier, the replicating instance synchronizes with the originating instance using state information from both instances and potentially rolling back information on the replicating instance. The UPDATERESYNC qualifier declares the replicating instance to be in a wholesome state matching some prior (or current) state of the originating instance; it causes MUPIP to update the information in the replication instance file of the originating instance and not modify information currently in the database on the replicating instance. After this command, the replicating instance catches up to the originating instance starting from its own current state. Use UPDATERESYNC only when you are absolutely certain that the replicating instance database was shut down normally with no errors, or appropriately copied from another instance with no errors.

### Important

You must always follow the steps described in the Database Replication chapter of the UNIX Administration and Operations Guide when migrating from a logical dual site (LDS) configuration to an LMS configuration, even if you are not changing GT.M releases.

## Stage 4: Journal Files Upgrade

On every GT.M upgrade:

* Create a fresh backup of your database.

* Generate new journal files (without back-links).

### Important

This is necessary because MUPIP JOURNAL cannot use journal files from a release other than its own for RECOVER, ROLLBACK, or EXTRACT.

## Stage 5: Trigger Definitions Upgrade

If you are upgrading from V5.4-002A/V5.4-002B/V5.5-000 to V6.2-000, you do not need to extract and reload triggers.

You need to extract and reload your trigger definitions only if you are upgrading from V5.4-000/V5.4-000A/V5.4-001 to V6.2-000. This is necessary because multi-line XECUTEs for triggers require a different internal storage format for triggers which makes triggers created in V5.4-000/V5.4-000A/V5.4-001 incompatible with V5.4-002/V5.4-002A/V5.4-002B/V5.5-000/V6.0-000/V6.0-001/V6.2-000.

To extract and reapply the trigger definitions on V6.2-000 using MUPIP TRIGGER:

1. Using the old version, execute a command like **mupip trigger -select="*" trigger_defs.trg**. Now, the output file trigger_defs.trg contains all trigger definitions.

2. Place -* at the beginning of the trigger_defs.trg file to remove the old trigger definitions.

3. Using V6.2-000, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.2-000 replicating instance using $ZTRIGGER():

1. Shut down the instance using the old version of GT.M.

2. Execute a command like **mumps -run %XCMD 'i $ztrigger("select")' > trigger_defs.trg** . Now, the output file trigger_defs.trg contains all trigger definitions.

3. Turn off replication on all regions.

4. Run **mumps -run %XCMD 'i $ztrigger("item","-*")** to remove the old trigger definitions.

5. Perform the upgrade procedure applicable for V6.2-000.

6. Run **mumps -run %XCMD 'if $ztrigger("file","trigger_defs.trg")'** to reapply your trigger definitions.

7. Turn replication on.

8. Connect to the originating instance.

### Note

Reloading triggers renumbers automatically generated trigger names.

## Downgrading to V5 or V4

You can downgrade a GT.M V6 database to V5 or V4 format using MUPIP DOWNGRADE.

Starting with V6.0-000, MUPIP DOWNGRADE supports the -VERSION qualifier with the following format:

```
MUPIP DOWNGRADE -VERSION=[V5|V4]
```

-VERSION specifies the desired version for the database header.

**To qualify for a downgrade from V6 to V5, your database must meet the following requirements:**

1. The database was created with a major version no greater than the target version.
2. The database does not contain any records that exceed the block size (spanning nodes).
3. The sizes of all the keys in database are less than 256 bytes.
4. There are no keys present in database with size greater than the Maximum-Key-Size specification the database header, that is, Maximum-Key-Size is assured.
5. The maximum Record size is small enough to accommodate key, overhead, and value within a block.

If your database meets all the above requirements, MUPIP DOWNGRADE -VERSION=V5 resets the database header to V5 elements which makes it compatible with V5 versions.

To qualify for a downgrade from V6 to V4, your database must meet the same downgrade requirements that are there for downgrading from V6 to V5.

If your database meets the downgrade requirements, perform the following steps to downgrade to V4:

1. In a GT.M V6.2-000 environment:
   a. Execute MUPIP SET -VERSION=v4 so that GT.M writes updates blocks in V4 format.
   b. Execute MUPIP REORG -DOWNGRADE to convert all blocks from V6 format to V4 format.
2. Bring down all V6 GT.M processes and execute MUPIP RUNDOWN -FILE on each database file to ensure that there are no processes accessing the database files.
3. Execute MUPIP DOWNGRADE -VERSION=V4 to change the database file header from V6 to V4.
4. Restore or recreate all the V4 global directory files.
5. Your database is now successfully downgraded to V4.

## Managing M mode and UTF-8 mode

On selected platforms, with International Components for Unicode (ICU) version 3.6 or later installed, GT.M's UTF-8 mode provides support for Unicode (ISO/IEC-10646) character strings. On other platforms, or on a system that does not have ICU 3.6 or later installed, GT.M only supports M mode.

On a system that has ICU installed, GT.M optionally installs support for both M mode and UTF-8 mode, including a utf8 subdirectory of the directory where GT.M is installed. From the same source file, depending upon the value of the environment variable gtm_chset, the GT.M compiler generates an object file either for M mode or UTF-8 mode. GT.M generates a new object file when it finds both a source and an object file, and the object predates the source file and was generated with the same setting of $gtm_chset/$ZCHset. A GT.M process generates an error if it encounters an object file generated with a different setting of $gtm_chset/$ZCHset than that processes' current value.

Always generate an M object module with a value of $gtm_chset/$ZCHset matching the value processes executing that module will have. As the GT.M installation itself contains utility programs written in M, their object files also conform to this rule. In order to use utility programs in both M mode and UTF-8 mode, the GT.M installation ensures that both M and UTF-8 versions of object modules exist, the latter in the utf8 subdirectory. This technique of segregating the object modules by their compilation mode prevents both frequent recompiles and errors in installations where both modes are in use. If your installation uses both modes, consider a similar pattern for structuring application object code repositories.

GT.M is installed in a parent directory and a utf8 subdirectory as follows:

* Actual files for GT.M executable programs (mumps, mupip, dse, lke, and so on) are in the parent directory, that is, the location specified for installation.
* Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for Unicode in the utf8 subdirectory, and one compiled without support for Unicode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install Unicode support.
* The utf8 subdirectory has files called mumps, mupip, dse, lke, and so on, which are relative symbolic links to the executables in the parent directory (for example, mumps is the symbolic link ../mumps).
* When a shell process sources the file gtmprofile, the behavior is as follows:
  * If $gtm_chset is "m", "M" or undefined, there is no change from the previous GT.M versions to the value of the environment variable $gtmroutines.
  * If $gtm_chset is "UTF-8" (the check is case-insensitive),
    * $gtm_dist is set to the utf8 subdirectory (that is, if GT.M is installed in /usr/lib/fis-gtm/gtm_V6.2-000_i686, then gtmprofile and gtmcshrc set $gtm_dist to /usr/lib/fis-gtm/gtm_V6.2-000_i686/utf8).
    * On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

For more information on gtmprofile and gtmcshrc, refer to the Basic Operations chapter of UNIX Administration and Operations Guide.

## Compiling ICU

GT.M versions prior to V5.3-004 require exactly ICU 3.6, however, V5.3-004 (or later) accept ICU 3.6 or later. For sample instructions to download ICU, configure it not to use multi-threading, and compile it for various platforms, refer to Appendix C: Compiling ICU on GT.M supported platforms of the UNIX Administration and Operations Guide.

Although GT.M uses ICU, ICU is not FIS software and FIS does not support ICU. The sample instructions for installing and configuring ICU are merely provided as a convenience to you.

Also, note that download sites, versions of compilers, and milli and micro releases of ICU may have changed since the dates when these instructions were tested rendering them out-of-date. Therefore, these instructions must be considered examples, not a cookbook.

## Setting the environment variable TERM

The environment variable TERM must specify a terminfo entry that accurately matches the terminal (or terminal emulator) settings. Refer to the terminfo man pages for more information on the terminal settings of the platform where GT.M needs to run.

* Some terminfo entries may seem to work properly but fail to recognize function key sequences or fail to position the cursor properly in response to escape sequences from GT.M. GT.M itself does not have any knowledge of specific terminal control characteristics. Therefore, it is important to specify the right terminfo entry to let GT.M communicate correctly with the terminal. You may need to add new terminfo entries depending on your specific platform and implementation. The terminal (emulator) vendor may also be able to help.
* GT.M uses the following terminfo capabilities. The full variable name is followed by the capname in parenthesis:

```
auto_right_margin(am), clr_eos(ed), clr_eol(el), columns(cols), cursor_address(cup), cursor_down(cud1),
 cursor_left(cub1), cursor_right(cuf1), cursor_up(cuu1), eat_newline_glitch(xenl), key_backspace(kbs),
 key_dc(kdch1),key_down(kcud1), key_left(kcub1), key_right(kcuf1), key_up(kcuu1), key_insert(kich1),
 keypad_local(rmkx),keypad_xmit(smkx), lines(lines).
```

GT.M sends keypad_xmit before terminal reads for direct mode and READs (other than READ *) if EDITING is enabled. GT.M sends keypad_local after these terminal reads.

## Installing Compression Libraries

If you plan to use the optional compression facility for replication, you must provide the compression library. The GT.M interface for compression libraries accepts the zlib compression libraries without any need for adaptation. These libraries are included in many UNIX distributions and are downloadable from the zlib home page. If you prefer to use other compression libraries, you need to configure or adapt them to provide the same API provided by zlib. Simple instructions for compiling zlib on a number of platforms follow. Although GT.M can use zlib, zlib is not FIS software and FIS does not support zlib. These instructions are merely provided as a convenience to you.

If a package for zlib is available with your operating system, FIS suggests that you use it rather than building your own.

**Solaris/cc** compiler from Sun Studio:

```
./configure --sharedmake CFLAGS="-KPIC -m64"
```

HP-UX(IA64)/HP C compiler:

```
./configure --sharedmake CFLAGS="+DD64"
```

AIX/XL compiler:

```
./configure --sharedAdd -q64 to the LDFLAGS line of the Makefilemake CFLAGS="-q64"
```

Linux/gcc:

```
./configure --sharedmake CFLAGS="-m64"
```

By default, GT.M searches for the libz.so shared library (libz.sl on HPUX PA-RISC) in the standard system library directories (for example, /usr/lib, /usr/local/lib, /usr/local/lib64). If the shared library is installed in a non-standard location, before starting replication, you must ensure that the environment variable LIBPATH (AIX) or LD_LIBRARY_PATH (other UNIX platforms) includes the directory containing the library. The Source and Receiver Server link the shared library at runtime. If this fails for any reason (such as file not found, or insufficient authorization), the replication logic logs a DLLNOOPEN error and continues with no compression.

# Change History

## V6.2-000

Fixes and enhancements specific to V6.2-000 are:

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-230 | C9605-000067 | Other Utilities | Assorted DSE corrections and enhancements ✓ |
| GTM-3589 | C9B03-001667 | MUMPS | MERGE of an unsubscripted undefined local source is a NOOP rather than an error |
| GTM-4250 | S9C03-002074 | MUPIP | In Unix, MUPIP LOAD error reporting improvements and new ONERROR qualifier ✓ |
| GTM-4274 | C9C04-001977 | DB | An event matching an entry in the Custom Errors does not shut off journaling |
| GTM-5687 | C9F04-002717 | MUMPS | ZSHOW "D" displays information on both the local and remote sides of a TCP socket ✓ |
| GTM-6047 | C9H07-002884 | MUMPS | Protect ZBREAK and ZSTEP against actions that include a NEW or excludive NEW |
| GTM-6197 | C9I06-003003 | Other Utilities | DSE FIND -REGION treats region names as case-insensitive and has some improved error messages |
| GTM-6348 | C9J03-003098 | DB | Use GDSCERT automatically and correctly during recovery from an abnormal process termination that impacts a database |
| GTM-6423 | S9J07-002735 | MUMPS | Protect $ZSEARCH() streams from interference from ZCOMPILE using a wild-card |
| GTM-6449 | S9J08-002739 | MUMPS | In UNIX, adjustments to GT.M handling of terminal Delete key and kdch1 ✓ |
| GTM-7003 | - | MUMPS | Appropriate error handling for GOTO a nonexistent label |
| GTM-7320 | - | MUMPS | In UNIX, socket passing over LOCAL SOCKETs ✓ |
| GTM-7337 | - | DB | Ignore VIEW "JNLWAIT" within a TP transaction ✓ |
| GTM-7380 | - | MUMPS | Field test grade of auto-relink |
| GTM-7390 | - | MUPIP | UNIX MUPIP ENDIANCVT correction to an odd failure handling pointers to global root blocks |
| GTM-7449 | - | MUMPS | UNIX encryption facility for sequential disk files ✓ |
| GTM-7472 | - | MUMPS | In UNIX, fix rare inappropriate behavior at process exit |

**Change History**

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-7509 | - | DB | $ZTRIGGER() and MUPIP TRIGGER replicate trigger definitions as logical actions from an originating/ primary instance to a replicating/secondary instance based on the new LGTRIG journal record ✓ |
| GTM-7547 | - | MUMPS | NEW $ETRAP does not change the $ETRAP value ✓ |
| GTM-7569 | - | DB | Improved error report for exceeding the maximum global size for a configuration |
| GTM-7662 | - | MUMPS | ZGOTO 0:entryref memory management improvement |
| GTM-7735 | - | MUMPS | New $ZSOCKET() function ✓ |
| GTM-7740 | - | DB | Protection against faulty collation transform logic |
| GTM-7808 | - | MUMPS | SEEK to a location for an unencrypted UNIX sequential disk file ✓ |
| GTM-7816 | - | MUPIP | MUPIP REORGINC message is a warning as documented |
| GTM-7859 | - | MUPIP | When returning with an error, MUPIP RUNDOWN removes any UNIX semaphores it has created during its operation |
| GTM-7863 | C9D01-002207 | MUMPS | In UNIX, $PRINCIPAL SOCKET device with no attached sockets acts as /dev/null ✓ |
| GTM-7897 | - | DB | UNIX database access on a read-only Filesystem ✓ |
| GTM-7919 | - | MUMPS | UNIX ZMESSAGE fix for many context arguments |
| GTM-7926 | - | MUMPS | GT.M executables validate that they reside $gtm_dist ✓ |
| GTM-7928 | - | MUMPS | Remove the deprecated $ZCALL external call interface |
| GTM-7929 | - | MUMPS | WRITE /WAIT for SOCKET device correction and improvement |
| GTM-7931 | - | MUMPS | Add context to LITNONGRAPH error ✓ |
| GTM-7934 | - | Other Utilities | Configure scripts support pre-deployment efforts of open source packagers |
| GTM-7936 | - | Other Utilities | DSE CACHE -SHOW displays database shared memory size and encrypted global buffer section information ✓ |
| GTM-7938 | - | MUPIP | In UNIX, prevent casading errors due to incorrect set-up with MUPIP ROLLBACK and RUNDOWN |
| GTM-7940 | - | DB | Restore VIEW "DBFLUSH" and "DBSYNC" and improve VIEW "EPOCH" |

**Change History**

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-7942 | - | MUMPS | In UNIX, replace INVOBJ with INVOBJFILE, which provides the file name ✓ |
| GTM-7948 | - | DB | UNIX correction to handling of last entry in libconfig configuration file used in encrypted replication |
| GTM-7950 | - | MUMPS | 32-bit x86 Linux works with the 586 ("Pentium compatible") instruction set ✓ |
| GTM-7953 | - | MUMPS | Fix interaction between $ORDER(,-1), large nodes and large MAX_KEY_SIZE |
| GTM-7954 | - | Other Utilities | Correction to GDE SHOW -COMMANDS for large MAX-KEY_SIZE or alternate collation |
| GTM-7962 | - | MUMPS | Prevent interaction between SET $ZSTATUS, Indirection and XECUTE |
| GTM-7963 | - | MUMPS | Prevent local variable problems subsequent to an UNDEF error |
| GTM-7971 | - | MUMPS | Prevent indefinite loop with full Boolean compilation and some Boolean espressions using extrinic functions |
| GTM-7972 | - | DB | In UNIX, fix a critical section issue that could degrade performance after many abnormal process terminations |
| GTM-7975 | - | MUMPS | Correction to odd exponentiation case |
| GTM-7976 | - | MUPIP | Improvement to MUPIP SET -JOURNAL maintaining previous journal links when those journals are read-only |
| GTM-7983 | - | Other Utilities | Remove spurious subscript for unsubscripted node from DSE DUMP -GLO and -ZWR |
| GTM-7984 | - | MUMPS | UNIX adjustments for FIFO, PIPE and Sequential Disk (SD) devices behavior controlled by STREAM and WRAP |
| GTM-7988 | - | MUMPS | ZWRITE respects VIEW "NOUNDEF" |
| GTM-7990 | - | MUPIP | In UNIX, MUPIP treats region names as case-insensitive and MUPIP EXTRACT accepts a region list to extract only specified regions only ✓ |
| GTM-7991 | - | MUPIP | Add adjacency to MUPIP SIZE and MUPIP INTEG -FAST ✓ |
| GTM-7993 | - | MUMPS | In UNIX, abandon automatic terminfo filling for missing terminfo entries |
| GTM-7994 | - | MUMPS | UNIX facility to separate direct mode terminators from TERMINATOR= deviceparameter settings ✓ |

**Change History**

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-7995 | - | MUPIP | In UNIX, eliminate interaction between a standalone MUPIP JOURNAL -ROLLBACK and Instance Freeze |
| GTM-7997 | - | MUMPS | In UNIX, maintain $X and $Y appropiately for FIXED READs for Sequential Disk, FIFO, and PIPE devices in M mode |
| GTM-7999 | - | MUMPS | UNIX $ZSYSLOG() function to send a message from the application to the system log ✓ |
| GTM-8001 | - | DB | set^%LCLCOL() returns success if requested state matches current state |
| GTM-8002 | - | DB | Prevent possible database damage subsequent to WRITE /EOF followed by CLOSE of a PIPE device |
| GTM-8003 | C9808-000628 | MUMPS | CTRL-C terminates long-running actions such as ZWRITE and informational messages restricted to direct mode |
| GTM-8006 | - | Other Utilities | GDE SHOW -MAP -REGION command displays output correctly for non-printable subscripts and ++ notation in the "Up to" column |
| GTM-8011 | - | DB | REPLINSTNOSHM message text update |
| GTM-8013 | - | MUMPS | Prevent UNIX SIG-11 from WRITE /WAIT with CONNECTED LOCAL Socket |
| GTM-8015 | - | MUMPS | gtm_boolean and gtm_side_effect supported values enforced and related $VIEW() fixes and improvements ✓ |
| GTM-8018 | - | MUMPS | WRITE to an arbitrary positions in an unencrypted UNIX sequential disk file and improved error messages ✓ |
| GTM-8019 | - | DB | $ZTRIGGER() and MUPIP TRIGGER treat the deletion of a non-existent trigger as a success (returns 0 to the shell and 1 to the GT.M process) ✓ |
| GTM-8021 | - | MUMPS | Allow a WRITE to a zero length file for a UNIX sequential device after a USE REWIND |
| GTM-8023 | - | MUPIP | Replication activation in UNIX when using Instance Freeze and running GT.M processes that have accessed a database |
| GTM-8027 | - | DB | $VIEW("REGION","") returns a NOTGBL error |
| GTM-8030 | - | MUMPS | Improve UNIX handling of certain SOCKET deviceparameters in UTF-8 mode |
| GTM-8036 | - | MUMPS | Improved PATCODE error detection and reporting |
| GTM-8042 | - | MUMPS | UNIX $ZPEEK() function generates an UNDEF error when VIEW UNDEF is not set and the 4th $ZPEEK() |

**Change History**

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| | | | argument (the optional format parameter) is specified but is undefined ✓ |
| GTM-8047 | - | MUMPS | Provide a better ZSHOW dump file under certain unusual out-of-memory conditions |
| GTM-8048 | - | Other Utilities | ^%RANDSTR returns an empty string if its arguments specify a zero-length result |
| GTM-8050 | - | MUMPS | Fix UNIX issue with JOB from within RESTARTing TP transactions |
| GTM-8052 | - | MUMPS | UNIX split $PRINCIPAL Fixes |
| GTM-8055 | - | MUMPS | UNIX LOCAL Socket File Cleanup |
| GTM-8056 | - | MUMPS | UNIX LOCAL SOCKET Device on $PRINCIPAL ✓ |
| GTM-8063 | - | MUMPS | Handle missing file descriptors for UNIX std* devices |
| GTM-8064 | - | MUMPS | Prevent local variable problems involving the interaction of exclusive NEW and TP restart variables in RESTARTing transactions |
| GTM-8065 | - | MUMPS | UNIX JOB command detects undefined local varaiable arguments |
| GTM-8070 | - | DB | In UNIX, provide more fairness in releasing critical resources |
| GTM-8072 | - | MUPIP | MUPIP uses NOREGION message for missing regions ✓ |
| GTM-8074 | - | MUPIP | UNIX MUPIP ROLLBACK shorter on a supplementary instance after a prior incomplete rollback |
| GTM-8075 | - | DB | UNIX replication appropriately process large journal data passing through external replication filters |
| GTM-8083 | - | DB | Correctly handle a TXTSRCMAT message in UNIX during trigger execution. |
| GTM-8084 | - | MUPIP | Correction to UNIX Spanning Node totals in multi-region MUPIP INTEG |
| GTM-8085 | - | MUMPS | UNIX $TEXT() and ZPRINT fixes |
| GTM-8086 | - | DB | Journal File Autoswitch Fixes |
| GTM-8092 | - | DB | Replication fix for an edge case involving very large nodes and a very small replication journal pool |
| GTM-8095 | - | DB | UNIX Replication fix for an edge case involving very large multi-region transactions |
| GTM-8100 | - | Other Utilities | %XCMD honors the SHELL setting for $gtm_etrap |
| GTM-8108 | - | MUMPS | Fix issues with $QUERY() of a local variable and UNDEF detection in certain circumstances |

**Change History**

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-8109 | - | MUMPS | In UNIX, appropriate $TEST maintenance in non-FIXED FOLLOW modes for READ # and READ * |
| GTM-8111 | - | MUMPS | EMBED_SOURCE option |
| GTM-8114 | - | MUMPS | Fix unusual interaction between TP RESTARTs and subscripted local variables |
| GTM-8115 | - | MUMPS | Fix problem with BREAK or exclusive NEW using a postconditional on a line after FOR |
| GTM-8116 | - | MUMPS | Fix for odd issue with $TEXT() of trigger code |
| GTM-8118 | - | MUPIP | UNIX Source Server improvements under certain circumstances |
| GTM-8121 | - | MUPIP | MUPIP RUNDOWN ignores the database file corrupt flag |
| GTM-8123 | - | MUMPS | A UNIX JOB command has no effect on open non-terminal and non-SOCKET devices |
| GTM-8128 | - | MUPIP | MUPIP EXTRACT handles large record counts |
| GTM-8132 | - | MUMPS | REWIND, INREWIND, and OUTREWIND deviceparameters for UNIX TRM devices ✓ |
| GTM-8133 | - | MUMPS | UNIX support for TRUNCATE output on $PRINCIPAL ✓ |
| GTM-8141 | - | MUMPS | In UNIX, handle ".." in a path in $zroutines or $zparse() |
| GTM-8142 | - | DB | Eliminate odd inappropriate error during encrypted database initialization |
| GTM-8145 | - | MUMPS | ZHELP and utility HELP content made current and commands deal better with misconfiguration issues and <CTRL-C> |
| GTM-8147 | - | Other Utilities | Update to the open source README |
| GTM-8154 | - | Other Utilities | DSE FIND -EXHAUSTIVE more capable |
| GTM-8155 | - | MUPIP | In UNIX, prevent inappropriate REPLXENDIANFAIL in cross-endian replication |
| GTM-8156 | - | MUPIP | In UNIX, MUPIP RUNDOWN without arguments avoids inappropriate messages advising operator actions when it runs and also for later actions |
| GTM-8157 | - | DB | In UNIX, prevent GTMSECSHRPERM loop when gtmsecshr is misconfigured |
| GTM-8159 | - | Other Utilities | gtminstall --verbose reports issues encountered while downloading missing GT.M components |

## M-Database Access

* GT.M leaves journaling enabled if the underlying journal error is present in the custom errors file. Previously, GT.M left journaling enabled on any error as long as the custom errors file was configured. Also, JNLOPNERR is included in the custom_errors_sample.txt file. [UNIX] (GTM-4274)

* The VIEW "GDSCERT":1 diagnostic setting works appropriately with logic used to recover global buffers from an abnormal termination of a process in the middle of an update. Previously, the unlikely combination of an abnormal termination combined with a database issue reported by the GDSCERT diagnostic capability could cause additional database problems. In addition, the global buffer recovery logic now uses the GDSCERT diagnostic capability to detect database integrity issues typically caused by kill -9 of processes actively updating a database. Note that the GDSCERT diagnostic setting is off by default, and must be explicitly enabled (normally under the direction of the FIS GT.M support team). (GTM-6348)

* GT.M ignores VIEW "JNLWAIT" issued by a process inside a TP transaction ($TLEVEL > 0). Previously it attempted the action and typically wasted resources to no effect as all the journal activity for a TP transaction occurs at commit time. However, it could have disrupted the transaction flow by waiting for non-TP updates or TRANSACTION="BATCH" updates that preceded the start of the current transaction. The workaround was to place any required VIEW "JNLWAIT" actions prior to any TSTART or avoid using BATCH when transaction hardening is required. (GTM-7337) ✓

* $ZTRIGGER() and MUPIP TRIGGER replicate trigger definitions as logical actions from an originating primary instance to a replicating secondary instance. This allows greater flexibility in configuration in trigger sets and database layouts (e.g. different number of regions, block sizes, maximum-record-size, etc.). Previously these actions replicated the low-level structural representation of trigger definitions (held in the hidden ^#t structure). While replicating the low level structures did in principle allow triggers on the source and receiver of a replicating stream to differ, it required deep knowledge and care to avoid certain problems, including:

  * New trigger definitions on the originating instance could inappropriately overwrite existing installed triggers on the replicating instance.

  * Inappropriate errors such as REC2BIG on a replicating instance with a smaller record size than the originating instance.

  * Integrity errors in the ^#t global on the replicating instance.

  LGTRIG records in journal extracts, including lost / unreplicated transaction files, reflect updates to trigger definitions.

  When replicating to prior releases, V6.2-000 sends trigger updates in the low level structural representation used by prior releases. When replicating to V6.2-000, and subsequent releases, V6.2-000 sends trigger updates in their logical representation. V6.2-000 is the only release that can deal with both representations. Therefore, for any rolling upgrade from a GT.M release preceding V6.2-000 to one subsequent to V6.2-000, during which an application expects to upgrade triggers, V6.2-000 is a required intermediate release. If no trigger updates are expected during the rolling upgrade, an intermediate upgrade to V6.2-000 is not required.

  MUPIP TRIGGER -SELECT and $ZTRIGGER("SELECT") work even if a multi-line XECUTE string does not terminate with a newline character. Previously in this situation, the "SELECT" terminated abnormally with a SIG-11. In addition, $ztrigger("ITEM",<multi-line-trigger-definition>) uses << to denote the definition of a multi-line -XECUTE string and $char(10) to denote the newline separator. For example:

```
set trigstr="+^a -commands=S -xecute=<<"_$char(10)_" do ^twork1"_$char(10)_" do ^twork2"_$char(10) write
 $ztrigger("item",trigstr)
```

  which is equivalent to the following:

```
GTM>write $ztrigger("file","agbl.trg")
```

  where the file agbl.trg contains the following multi-line trigger definition.

```
+^a -commands=S -xecute=<<
```

```
  do ^twork1
  do ^twork2
>>
```

Note that the "file" form requires the >> terminator, but the item form does not require or permit it. Note also that that in the "item" form, the trigger logic does not have to appear immediately adjacent to the -xecute=<<, but must start with a $char(10), so these examples are also equivalent.

```
set trigstr="+^a -xecute=<< -commands=S"_$char(10)_" do ^twork1"_$char(10)_" do ^twork2"_$char(10) write
 $ztrigger("item",trigstr)
set trigstr="+^a -xecute=<< -commands=S "_$char(10)_" do ^twork1"_$char(10)_" do ^twork2"_$char(10) write
 $ztrigger("item",trigstr)
```

MUPIP JOURNAL -EXTRACT identifies (first line in the extract file) V6.2-000 journals with GDSJEX07 for regular extracts and GDSJDX07 for -DETAIL extracts. [UNIX](GTM-7509) ✓

* GT.M issues a MAXBTLEVEL error, which includes the region name, when an application attempts to create a new block that would take the global tree beyond the current depth limit. Previously this condition produced a GVPUTFAIL or TPFAIL error with code failure code SSSS. (GTM-7569)

* GT.M protects buffers used in the alternative collation interface by ensuring any memory mismanagement by the user-supplied alternative collation library gets an immediate segmentation violation (SIG-11). Previously GT.M did not provide such protection, which allowed user-written code to cause delayed, and difficult to diagnose, problems due to damage to memory used by GT.M itself. (GTM-7740)

* MUPIP INTEG, MUPIP EXTRACT (without freeze), and GT.M processes performing read operations can access databases on a filesystem mounted read-only. Previously these operations resulted in a DBFILERR message with TEXT indicating "gtmsecshr failed to update database file header", or "gtmsecshr was unable to write header to disk", "No such file or directory", MUSTANDALONE, and INTEGERRS for MUPIP INTEG -NOONLINE -FILE. Please note:

  * The filesystem must remain read-only for the duration of any process that opens a database file resident on it. If a read-only file system is switched to read-write while GT.M processes have database files open on it, and other processes update those databases, the read-only processes are likely to read incorrect or corrupt data.

  * When the filesystem is read-only the shared memory resources which are typically shared among multiple processes instead become private to each process, so memory resource use increases with each additional concurrent process.

  * M LOCKs mapped to regions that map to database files on read-only filesystems are visible only to the process that owns the locks, and are invisible to other processes.

  [UNIX] (GTM-7897) ✓

* The VIEW command recognizes the keywords DBFLUSH and DBSYNC. Previously, due to a regression introduced in V6.1-000, VIEW incorrectly produced a VIEWNOTFOUND error when passed either keyword. In addition, VIEW "EPOCH" does not perform superfluous fsync system calls, and, with NOBEFORE_IMAGE journaling, it does not flush global or journal buffers from memory to secondary storage. These steps are unnecessary for correct journal recovery. (GTM-7940)

* The GT.M reference implementation for database encryption appropriately handles the last entry in the libconfig configuration file. In V6.1-000, the reference implementation ignored the last entry in the "database.keys" section in the configuration file. The workaround was to add an empty entry at the end of "database.keys" section. [UNIX] (GTM-7948)

* GT.M and its utilities more cleanly maintain the mechanism used to manage database critical sections. Previously a terminating process could inappropriately decrease the number of available entries in a queueing structure within the critical section logic and eventually cause less efficient critical section management. Note: This fix was not present in V6.1-000 as claimed in its release note for GTM-7804. [UNIX] (GTM-7972)

* $$set^%LCLCOL() checks requested against current state and returns success if they match. Previously, any attempt use the function failed when there were local variables even if the requested settings did not require a change. Also, the function returns a 0 for FALSE. Previously, if it did not work, it returned an empty string, which is also FALSE. (GTM-8001)

## M-Database Access

* CLOSE of a PIPE device preceded by a WRITE /EOF to that device works correctly. Previously a CLOSE of a PIPE could inadvertently close file descriptors of any files opened subsequent to a WRITE /EOF to that PIPE device. In the event a database file was opened between the WRITE /EOF and the CLOSE, subsequent updates could potentially cause database damage. This issue was discovered in the GT.M development environment and was never reported by any user. Such damage, while theoretically possible, is unlikely in practice because application code would typically CLOSE a PIPE device after a WRITE /EOF and database files are usually opened early in the lifetime of a process. (GTM-8002)

* REPLINSTNOSHM error message text no longer suggests to verify the database is listed in the instance file because REPLINSTNOSHM can occur if no connection between the database file and journal pool exists. [UNIX](GTM-8011)

* $ZTRIGGER() and MUPIP TRIGGER treat deletion of a non-existent trigger as a success; if that is the only operation, or one of a set of successful operations, they return success - respectively 0 to the shell, and 1 to the GT.M process. Previously these operations treated deletion of a nonexistent trigger as a failure which meant they did not install any other trigger definitions specified in the same set, and returned a failure status - respectively 1 to the shell and 0 to the GT.M process. Also, MUPIP TRIGGER and $ZTRIGGER() return failure in case of trigger selection using trigger names where the number after the pound-sign (#) starts with a 0 (which is an impossible auto-generated trigger name). Previously, this returned success even though it selected no triggers. [UNIX](GTM-8019) ✓

* $VIEW("REGION","") returns a NOTGBL error. In V6.1-000, this terminated with a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). (GTM-8027)

* GT.M minimizes the chance of prolonged inability of an exiting process to acquire a critical section in order to release a resource. Previously, if other processes were keeping a backlog waiting for that critical section, an exiting process could have a negligible chance of acquiring a critical section . [UNIX](GTM-8070)

* The Source and Receiver Servers appropriately process large amount of journal data passing through external filters (user-supplied programs). Previously, the Source and/or Receiver Server could terminate abnormally with a segmentation violation (SIG-11) when a large amount of journal data passed through the external filters. In addition, external replication filters handle large transactions faster than previously.[UNIX] (GTM-8075)

* GT.M correctly handles a TXTSRCMAT message while executing inside of a trigger in an environment where TP RESTARTs are also happening; previously, in this scenario, the process could terminate with a segmentation violation (SIG-11). [UNIX] (GTM-8083)

* GT.M issues a JNLEXTEND error when it encounters an error during the automatic switching of a journal file that has reached its size limit. If replication is not on, GT.M turns journaling off for the corresponding database, but leaves it enabled. If replication is on, INSTANCE_FREEZE_ON_ERROR is configured and JNLEXTEND appears in the gtm_custom_errors file, GT.M freezes the instance, but leaves journaling on; otherwise it places journaling for the region in the WAS_ON state, so that replication can continue as long as it can operate out of the replication journal pool and does not have to use the now inactive journal file(s). To avoid problems with concurrent journal file switches, MUPIP JOURNAL operations which take individual journal files (not "*") no longer recover from interrupted journal switch automatically, instead producing a JNLFILEOPNERR error. [UNIX] (GTM-8086)

* SETs of global nodes with a value close to 1MiB in size work appropriately on a replicated database even if the journal pool buffer size is set to a small value of close to 1MiB. Previously, such a SET could terminate abnormally with a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). The workaround was to set the jnlpool buffer size to at least 2MiB. (GTM-8092)

* The Source Server correctly sends journal records corresponding to a transaction that is spread across more than one region. Previously, in rare cases, if the total journal record size of the transaction exceeded approximately 1MiB, the Source Server could terminate abnormally with a segmentation violation (UNIX SIG-11). [UNIX] (GTM-8095)

* A GT.M process opening a database file concurrently with another process closing that file works correctly. Previously, such a process could receive a REQRECOV error message. [UNIX] (GTM-8142)

* GT.M issues one GTMSECSHRPERM error per failure to remove a shared resource (semaphore or shared memory) associated with a database when the resource was created by different user AND gtmsecshr, which is required to do such a removal, has not been configured with the correct permissions (uid / gid). In versions V6.0-002 through V6.1-000, this resulted in an indefinite sequence of GTMSECSHRPERM messages. FIS strongly recommends using an FIS provided installation method to install GT.M, such as the configure script included with GT.M, in preference to ad hoc procedures which, while they may install one or another GT.M release correctly, may not continue to work with subsequent releases of GT.M[UNIX] (GTM-8157)

## M-Other Than Database Access

* MERGE becomes a NOOP when the destination is a descendant of the source and the source is an undefined, unsubscripted local variable. This is consistent with MERGE when the undefined source is either a global or a subscripted local. Previously, such a MERGE gave a MERGEDESC error. (GTM-3589)

* ZSHOW "D" reports available information on both the local and remote sides of a TCP socket. Previously it only reported information on the remote side for connected sockets. (GTM-5687) ✓

* GT.M appropriately manages the case where a ZBREAK or ZSTEP insert an exclusive NEW or NEW all and local variables are referenced later in the target line of the command; previously, this could cause the process to terminate with a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). (GTM-6047)

* ZCOMPILE with a wild-card does not disturb any $ZSEARCH() stream; previously it used stream 0 which could interact with application code. (GTM-6423)

* GT.M deletes the character under the cursor when you press the key on the keyboard that sends the escape sequence which maps to the kdch1 capability in your current terminfo entry (by convention, the Delete key). Furthermore, if the current terminfo entry is missing the kdch1 capability, GT.M uses a default value derived from members of the DEC VT terminal family, as it does for selected other missing terminfo capabilities. Previously, in response to the escape sequence defined by kdch1, GT.M deleted the character immediately on the left, akin to the Backspace key and assumed an inappropriate value if a definition for kdch1 was missing. If you wish to retain the prior behavior, the simplest way is to configure your terminal emulator to send the same character sequences for the Delete key that it does for the Backspace key. You can alternatively modify your terminfo setting: for example, create an editable version of your terminfo entry in a temporary file with a command such as: **infocmp > /tmp/$$_$TERM** and edit the temporary file to replace the entry for the kbs capability with the one in the kdch1 capability. Save your changes, and compile the edited file into a usable terminfo entry, for example:

  ```
  export TERMINFO=$HOME/.terminfo # You may need to add this to your login profile
  profilemkdir -p $TERMINFO
  tic /tmp/$$_$TERM # or whatever your temporary file name was
  ```

  When modifying terminfo capabilities, always look for unintended changes in the behavior of other applications, e.g., text editors, that also rely on those capabilities. In the worst case, you may need to toggle between alternate terminfo entries for GT.M and other applications while you evaluate different options. Also, for terminfo entries without the cud1 capability, GT.M uses a linefeed when moving to the next line in direct mode. Previously, GT.M used a carriage return, which meant that pressing Ctrl-E to move the cursor to the end of the line for a line displayed wrapped over multiple lines, would place the cursor a visually incorrect position. [UNIX] (GTM-6449) ✓

* Attempting to GOTO to a non-existent label in a routine invoked at run-time from a 64-bit UNIX edition reports a LABELNOTFND error with the location as the last line in the routine, and the 32-bit Linux and AIX editions report a LABELUNKNOWN error. In that case, repeat the compile to get a LABELMISSING error, which provides the name of the label for which to search. Previously, such a GOTO could cause a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). (GTM-7003)

* A process can transfer DETACHed TCP or LOCAL sockets (i.e., sockets in the socket pool) to another GT.M process by issuing WRITE /PASS and WRITE /ACCEPT commands. Both WRITE /PASS and WRITE /ACCEPT, require a current $IO that is a CONNECTed (notLISTENing), LOCAL domain (not TCP), SOCKET device. GT.M issues CONNSOCKREQ or LOCALSOCKREQ errors, respectively, when those conditions are not met.

  The process sending sockets executes:

  ```
  WRITE /PASS([targetpid],[timeout],handle[,handle]...)
  ```

  * If a numeric targetpid is specified, GT.M matches the value against the process id ($JOB) of the process receiving the sockets. GT.M uses a system service to perform this check on platforms that support it; currently Linux, AIX, and Solaris. On platforms which do not

implement the service (HP-UX), GT.M ignores the targetpid. If the pids do not match, GT.M issues a PEERPIDMISMATCH error and does not transfer the sockets.

* If a numeric timeout is specified, GT.M sets $TEST to 1 if the transfer completes within the specified time, and otherwise sets $TEST to 0 and does not transfer any of the sockets.

* Each handle specifies the name of a socket in the socket pool.

* On a successful transfer, GT.M closes the connection of the sending process to the specified and sent sockets. In any case where the transfer does not complete, GT.M retains all the sockets in the socket pool of the sender.

The process receiving sockets executes:

```
WRITE /ACCEPT(.lvar,[sourcepid],[timeout][,[handle]]...)
```

* lvar is an unsubscripted local variable name (lvn) which must be passed by reference indicated with a period (".") prefix. On successful completion, the specified unsubscripted lvn contains the handles of the received socket, in the order they were sent, delimited with a vertical bar ("|"). GT.M places the sockets in the socket pool, so the process can ATTACH them to an appropriate SOCKET device for subsequent use.

* If a numeric sourcepid is specified, GT.M matches the value against the process id ($JOB) of the process sending the sockets. On platforms which do not implement the service (HP-UX), GT.M ignores the targetpid. If the pids do not match, GT.M issues a PEERPIDMISMATCH error and does not transfer the sockets.

* If a numeric timeout is specified, GT.M sets $TEST to 1 if the transfer completes within the specified time, and otherwise sets $TEST to 0 and does not transfer the sockets.

* If any handles are specified, GT.M assigns the provided handle names to the received sockets in the order in which they appear in the WRITE /PASS of the sending process; empty items in the comma delimited handle list act to preserve ordering. Where the list provides no handle, the socket retains the handle provided by the sender. In either case, if there is already a socket with the transfer handle name in the socket pool, GT.M generates a new handle name for the transfer socket. GT.M ignores excess handles specified beyond the number of incoming sockets.

For both WRITE /PASS and WRITE /ACCEPT, $IO must be a SOCKET device, and the current socket of the device must be CONNECTED(not LISTENING) and LOCAL domain (not TCP).

SOCKET devices so not support mixing other READs and WRITEs with socket passing on the same CONNECTED LOCAL socket and produce SOCKPASSDATAMIX errors. The application may perform multiple WRITE /PASS and WRITE /ACCEPT operations in either direction on the socket before issuing a CLOSE.

Note that the receiving process must establish desired deviceparameters (e.g., DELIMITER) either by ATTACHing it to a SOCKET device that provides the characteristic for all its sockets, or by a subsequent USE that specifies the appropriate deviceparameter(s). GT.M transfers only the socket connection itself, the socket handle, and buffered socket data (if any).
[UNIX] (GTM-7320) ✓

* The following note describes field test grade auto-relink functionality that is available in V6.2-000. Although V6.2-000 when used without auto-relink (i.e., without use of the ZRUPDATE command or the special syntax in $ZROUTINES) is a generally available, production grade release, use auto-relink only in development and test environments where field test grade functionality is acceptable.

By suffixing one or more directory names in $ZROUTINES with a single asterisk (*), processes can subscribe to updates of object files published in those directories. At the invocation of DO, GOTO, or ZGOTO, extrinsic functions, $TEXT(), or ZPRINT that specify an entryref which includes a routine name (vs. a label without a routine name), mumps processes (and mupip processes executing trigger logic) automatically relink ("auto-relink") and execute published new versions of routines.

* Label references (i.e., without a routine name) - whether direct, through indirection or through XECUTE - always refer to the current routine, and do not invoke auto-relink logic.

* Use shell quoting rules when appending asterisks to directory names in the gtmroutines environment variable - asterisks must be passed in to GT.M, and not expanded by the shell.

* GT.M accepts but ignores asterisk suffixes to directory names on UNIX platforms where it does not provide auto-relinking (32-bit Linux on x86, and HP-UX on Itanium).

The ZRUPDATE command publishes of new versions of routines to subscribers: ZRU[PDATE][:tvexpr] expr [,...] where exprs have the syntax documented for $ZSEARCH() and expand to object file names. To remove routines, delete the object and source files and publish the names of the deleted object files. Removal requires file names to be explicitly specified, because patterns with wild-cards cannot match deleted files.

If the path to a file is non-existent, the request is ignored except in the case where one desires a currently shared object file (one that was accessed before it was deleted) to no longer be shared. In V6.2-000, if the process executing the ZRUPDATE does not have read permission to any directory in the path to a file, ZRUPDATE ignores the request; FIS intends for this to generate an error in the production release.

The following behaviors of GT.M are unchanged, but interactions with auto-relink may not be obvious and are therefore spelled out:

* Explicit or automatic ZLINK: processes perform explicit ZLINK commands and automatic ZLINKs (when an entryref is invoked, and the process does not have a matching routine linked) when needed. A process that needs a routine links it when needed, without awaiting the completion of concurrent ZRUPDATE activity by other processes. In other words, ZRUPDATE provides notification of processes that have requested notification for auto-relink of updated versions, but does not impede a process that needs a routine.

* Recursive relink: Relinking a routine for which an invocation already exists on the stack is controlled by the setting of VIEW "LINK" (initialized from the gtm_link environment variable). If the setting includes "NORECURSIVE", auto-relinking only occurs if an invocation of a routine with that name is not already on the call stack. If the setting is "RECURSIVE" auto-relinking links the new version following normal recursive relink behavior.

* Break points: break points in a routine that is replaced either explicitly (with a ZLINK command) or implicitly by auto-relink are removed. In the case where VIEW "LINK" is set to "RECURSIVE", the break points are not removed in the earlier version and are not added to the newer version. Since the ZBREAK command only affects the current version of the routine, the only way to remove the break points in the earlier version is to remove ALL breakpoints. FIS expects to change this behavior in the production release such that any breakpoint in a routine prevents auto-relink for that routine in that process.

Notes:

To effect auto-relink, GT.M creates small temporary files in the directory referred to by $gtm_linktmpdir (defaulting to $gtm_tmp, which in turn defaults to /tmp, if unspecified). The names of these files are of the form gtm-relinkctl-<md5sum> where <md5sum> is a hash of the realpath() to an auto-relink directory. In the field test software, the permissions are determined by the umask of the process creating a file. In the production software, FIS intends for the group and permissions to match those for shared resources as described in the section "Shared Resources Authorization Permissions" in Appendix E (GT.M Security Philosophy) of the GT.M Administration and Operations Guide UNIX Edition. FIS recommends that all processes that share a directory whose contents are subject to ZRUPDATE use the same value for $gtm_linktmpdir so that all processes see update notifications - with different values of $gtm_linktmpdir, a ZRUPDATE by a process with one value of $gtm_linktmpdir would not be observed by a process with a different value of that environment variable.

* When a process that has subscribed to updates to routines in a directory links to a routine from that directory, it maps the object file to its address space, sharing the object code with other processes that are so subscribed, and without requiring the object code to be placed in a shared library. Processes that have not subscribed to updates link using process-private heap space as they have in prior versions. Linking from shared libraries works as it has in prior versions.

* AIX has known architectural limitations to scalability when large numbers of processes (thousands to tens of thousands) auto-relink large numbers of routines (again thousands to tens of thousands of routines). In the typical case where the number of dynamically modified routines is much smaller than the total number of routines in an enterprise scale application, a scalable way for large applications to use auto-relinking on AIX is to place all routines in shared libraries, and define $ZROUTINES with an auto-relink enabled patch directory preceding the shared library. Scalability on Solaris is not known: FIS is not aware of any inherent scalability issues other than those imposed by available hardware resources when the product of the number of processes and the number of auto-relinked remains well below the number of file descriptors made available by the OS.

* Changing $ZROUTINES currently causes all routines linked from auto-relink enabled directories in the process to be re-linked. FIS expects to change this behavior in the production release to make it more efficient.

[UNIX except 32-bit Linux on x86 and HP-UX on Itanium] (GTM-7380)

* Sequential files, PIPE, and FIFO devices can READ and WRITE encrypted data. The new KEY/IKEY/OKEY deviceparameters specify a key by name and an optional initialization vector (IV). GT.M calls an encryption plugin using the GT.M encryption API and can use any library that conforms to the API. The encryption plugin in turn can call user-selected cryptographic libraries for cryptographic functionality. GT.M ships with a reference implementation of the plugin that FIS tests in-house with selected ciphers as implemented by selected versions of standard libraries. The operation of the reference implementation of the encryption plugin is described below under: "Additional Information for GTM-7449 Encrypted File IO". GT.M does not include, and FIS support for GT.M does not extend to, any cryptographic software or customized plugin. Note that encrypted files must be written and read sequentially from the beginning (including the Byte Order Marker for UTF files); GT.M supports READ and WRITE operations at arbitrary locations in a file only for unencrypted files.

  The KEY/IKEY/OKEY deviceparameters apply to OPEN and USE operations and consist of two space-separated parts: a key name and an optional IV for the encryption/decryption state machine. The case-sensitive key name must match one of the key names in the 'files' section of the configuration file; the key name and IV are passed as binary sequences of bytes to the plugin. Since GT.M only uses the first space in the deviceparameter to delimit the end of the key, the IV can include any content, including spaces. The GT.M runtime system merely uses the plugin: to pass the IV to the cryptographic libraries used, which determine the length of the IV, whether an IV that is less than the required size is zero padded, and whether an IV that is longer than the required length generates an error. FIS suggests using $ZCHAR() in preference to $CHAR() when building IV byte sequences, and to make sure that IV sequences are not unintentionally subjected to numeric conversion.

  The basic steps to use a key and IV to create an encrypted file and decrypt its data in a testing environment are as follows. These steps are solely for demonstration purposes. You must understand and appropriately adjust the steps before using them in a production environment. For example, in a production environment you should keep the key files in a secure location, protected with appropriate permissions from unauthorized access (such as 0500 for directories and 0400 for individual files). File encryption is just one of many components of a comprehensive security plan.

```
export LD_LIBRARY_PATH=/usr/local/lib
export GNUPGHOME=$PWD/mygnupg
$gtm_dist/plugin/gtmcrypt/gen_keypair.sh mykeypair@gtm Keymaster
$gtm_dist/plugin/gtmcrypt/gen_sym_key.sh 0 Sunday.key
$gtm_dist/plugin/gtmcrypt/gen_sym_key.sh 0 Monday.key
$gtm_dist/plugin/gtmcrypt/gen_sym_key.sh 0 Tuesday.key
$gtm_dist/plugin/gtmcrypt/gen_sym_key.sh 0 Wednesday.key
$gtm_dist/plugin/gtmcrypt/gen_sym_key.sh 0 Thursday.key
$gtm_dist/plugin/gtmcrypt/gen_sym_key.sh 0 Friday.key
$gtm_dist/plugin/gtmcrypt/gen_sym_key.sh 0 Saturday.key
echo -n "Enter password for gtm_passwd";export gtm_passwd="$gtm_dist/plugin/gtmcrypt/maskpass|cut -f 3 -d " " "
export gtmcrypt_config=mygtmcryptfile
cat mygtmcryptfile
 files: {
           CustomerReportKey1: "Sunday.key";
           CustomerReportKey2: "Monday.key";
           CustomerReportKey3: "Tuesday.key";
           CustomerReportKey4: "Wednesday.key";
           CustomerReportKey5: "Thursday.key";
           CustomerReportKey6: "Friday.key";
           CustomerReportKey7: "Saturday.key";
};
$gtm_dist/mumps -dir
GTM>zprint ^encrfile
encrfile
  set now=$horolog
  set timestamp=$zdate(now,"YYYYMMDDAM1260SS")
  set dayofweek=$zdate(now,"DAY","","1,2,3,4,5,6,7")
  set file="Customers"_timestamp_".log"
  open file:(newversion:key="CustomerReportKey"_dayofweek_" "_timestamp)
  use file
  write "Customer Report - Page 1",!
  close file
  write "IV : ",timestamp,!,"Key : CustomerReportKey"_dayofweek

GTM>do ^encrfile
IV : 20140911AM042419
Key : CustomerReportKey5
GTM>zprint ^readencrfile
readencrfile(key,iv)
  set file="Customers"_iv_".log"
  open file:(key=key_" "_iv)
  use file
  for  read data use $principal write data,! use file quit:$zeof
  close file
GTM>do ^readencrfile("CustomerReportKey5","20140911AM042419")
Customer Report - Page 1
GTM>
```

In this example, the key name is CustomerReportKey followed by the number representing the day of the week, and IV is a timestamp, which is also a part of the file name. Although all reports start with the same string "Customer Report - Page 1", using a different IV for each file ensures that encrypted data begins with a different sequence of bytes, and making that IV a part of the file name ensures that the recipient of a report (who would have access to the key) can easily deduce the IV needed to decrypt the contents.

Separate IKEY and OKEY deviceparameters allow different keys for READ from and WRITE to a device; for example, when a GT.M process is an element of a UNIX pipe. Because encryption ciphers use state machines (which are initialized with the IV at the beginning of the file), GT.M permits READ and WRITE operations only either starting at the beginning of a file, or at the position at which the last READ or WRITE operation completed. In particular, non-empty files cannot be opened in APPEND mode; the SEEK deviceparameter is prohibited; and the TRUNCATE is only permitted at the beginning of a file or at the end, the former deleting the contents, and the latter effectively a no-op.

A USE command with a KEY/IKEY/OKEY deviceparameter that attempts to change the cipher key or IV, including disabling encryption (by specifying an empty key), only succeeds prior to the first WRITE or READ, or after the encryption or decryption state has been reset, such as after a REWIND (only for READ) or a TRUNCATE at the start of a file (both READ and WRITE).

For more information, refer to Additional Information Additional Information for GTM-7449 Encrypted File IO Plugin Reference Implementation.

[UNIX] (GTM-7449) ✓

* GT.M manages timer-driven events during shutdown with care. Previously, GT.M cancelled all scheduled timed events early in process shutdown, which caused very occasional inappropriate behavior, such as repeated system log messages. [UNIX] (GTM-7472)

* NEW $ETRAP has the effect of saving the current value of $ETRAP or $ZTRAP, if $ZTRAP is in effect, without changing the existing value of $ETRAP, as required by the ISO standard. This means you should revise application programs that NEW $ETRAP without immediately setting it to a value such that a SET $ETRAP="" proximately follows the existing NEW $ETRAP. SET $ETRAP no longer saves the value of $ZTRAP, if $ZTRAP was in effect, to be more compliant with our current understanding of the intent of the ISO standard. This means If your application does not intend to change the trap mechanism for one or more prior frames, you must explicitly NEW $ETRAP (or $ZTRAP) before the SET $ETRAP. Because GT.M supports mixing of $ETRAP and $ZTRAP, two values which address the same need, NEW of either stacks the one that was in control so that as GT.M unwinds the M virtual machine stack in response to QUIT (or ZGOTO) commands, it restores control to the trap that last had control at the current stack level. If you are using only $ETRAP

or only $ZTRAP these changes should have no impact, but even if you are mixing them, we expect the impact should be small. Previously, NEW $ETRAP modified $ETRAP to the empty string and SET $ETRAP implicitly NEW'd a non-empty $ZTRAP. (GTM-7547) ✓

* ZGOTO 0:entryref tidies up process memory; previously it sometimes failed to free small amounts of space which could cause a slow memory leak. (GTM-7662)

* The $ZSOCKET() function returns information about a SOCKET device and its attached sockets. The format of the $ZSOCKET() function is:

```
$ZSOCKET(expr1,expr2[,[expr3][,expr4]])
```

  * The first expression specifies the SOCKET device name; an empty string returns the same result as the current device ($IO). If the first expression is not specified, $ZSOCKET() returns information about sockets in the socketpool. Specifying a device other than a SOCKET device for the $ZSOCKET() function produces a ZSOCKETNOTSOCK error.

  * The second expression specifies a keyword identifying the type of information returned and the optional third expression usually specifies the index (starting at zero) of a socket attached to the device; if the index is outside the range of attached sockets, $ZSOCKET() returns an empty string. If the third expression is not specified, $ZSOCKET() returns information about the current socket. Using an invalid keyword produces a ZSOCKETATTR error. The fourth expression specifies an individual delimiter when the second expression specifies DELIMITER. For more information, see the following table. Note that changes to the socket collection for a SOCKET device using OPEN, CLOSE, USE :ATTACH, or USE :DETACH may change the index for a socket.

| Keyword | Arguments | Returns |
|---|---|---|
| CURRENTINDEX | | The index (starting at zero) of the current socket for the SOCKET device. |
| DELIMITER | index[, delimiter] | If only index is specified, the number of delimiters.<br><br>If delimiter is also specified, selects which delimiter to return. The first delimiter is zero. |
| DESCRIPTOR | index | The OS socket descriptor for the socket. |
| HOWCREATED | index | LISTEN, CONNECT, ACCEPTED, PRINCIPAL, or PASSED<br><br>PRINCIPAL indicates that the socket is the $PRINCIPAL of the process.<br><br>PASSED indicates a socket passed by WRITE /ACCEPT. |
| INDEX | handle | The current index of the socket named by handle. |
| IOERROR | index | 1 if IOERROR=TRAP otherwise 0. |
| LOCALADDRESS | index | The address of the local side of the socket. For TCP sockets: the IPv4 or IPv6 numeric address. For LOCAL socket: the path. |
| LOCALPORT | index | The numeric port of the local side of a TCP socket. |

| Keyword | Arguments | Returns |
|---|---|---|
| MOREREADTIME | index | The value of the MOREREADTIME device parameter if it was specified, otherwise an empty string. |
| NUMBER | | The number of sockets in the SOCKET device. |
| PARENT | index | If the socket was created from a LISTENing socket: the handle of the LISTENing socket. |
| PROTOCOL | index | TCP, TCP6, or LOCAL |
| REMOTEADDRESS | index | The address of the remote side of the socket. For TCP sockets: the IPv4 or IPv6 numeric address. For LOCAL socket: the path. |
| REMOTEPORT | index | The numeric port of the remote side of a TCP socket. |
| SOCKETHANDLE | index | The handle for the selected socket. |
| STATE | index | One of LISTENING, CONNECTED, BOUND, or CONNECTINPROGRESS |
| ZBFSIZE | index | Size of the GT.M buffer in bytes. |
| ZFF | index | The value of the ZFF device parameter. |
| ZIBFSIZE | index | Size of the OS buffer in bytes (SO_RCVBUF). |
| ZDELAY | index | 1 if Nagle algorithm enabled, otherwise 0. |

(GTM-7735) ✓

* A unencrypted sequential disk device accepts the SEEK=strexpr deviceparameter for OPEN and USE. The format of strexpr is a string of the form "[+|-]integer" where an unsigned value specifies an offset from the beginning of the file, and an explicitly signed value specifies an offset relative to the current file position. For STREAM or VARIABLE format, the positive intexpr after any sign is a byte offset, while for a FIXED format, it is a record offset. $ZKEY contains the current position in the file if $IO is a sequential disk device. This is in bytes for STREAM and VARIABLE formats, and in a record,byte pair for FIXED format.

  SEEKs and normal READs or WRITEs always produce a zero byte position; a non-zero byte position in $ZKEY for FIXED format operation indicates a partially read record, caused by a READ # or READ *, or a partially written record where less than WIDTH characters are written. In FIXED mode, the information returned for $ZKEY is a function of record size. If a USE command changes record size by specifying the WIDTH deviceparameter while the file is open, $ZKEY offsets change accordingly; if record size changes, previously saved values of $ZKEY are likely inappropriate for use with SEEK. Previously, sequential disk devices did not support SEEK, and always reported an empty string $ZKEY.

  When $PRINCIPAL identifies a device that supports SEEK, the SEEK or INSEEK device parameters perform a SEEK on input and OUTSEEK performs a SEEK on the output.

  For a sequential disk device CLOSEd with the NODESTROY deviceparameter, a subsequent OPEN of the device with no deviceparameters restores the device state including its file position; or to the file position specified by a SEEK deviceparameter. An OPEN with additional deviceparameters positions the device to the beginning of the file or to the end of file if APPEND is specified and then applies any SEEK specified as a deviceparameter. Previously, an OPEN after a CLOSE with NODESTROY did not save the state such as file position and subsequent READs using a UTF character set could produce errors. [UNIX](GTM-7808) ✓

* GT.M discards reads and writes against an empty SOCKET device (i.e, one with all sockets detached) if it is the $PRINCIPAL device. Read and write operations against an empty non-$PRINCIPAL SOCKET device continue to produce a NOSOCKETINDEV message. Previously all such operations resulted in a NOSOCKETINDEV message. GT.M opens /dev/null as a placeholder for a socket which used to be associated with $PRINCIPAL via stdin when it is closed. Previously opening new devices or creating new sockets following the SOCKET CLOSE resulted in reuse of the stdin file descriptor and improper device operation. [UNIX] (GTM-7863) ✓

* ZMESSAGE inserts specified context information for all the data types used internally by GT.M and for up to 17 context arguments. Previously, ZMESSAGE acted as a NOOP when context was specified for an item with one of certain internal data types, or if the number of specified context items exceeded 12. [UNIX] (GTM-7919)

* When possible, GT.M verifies that MUMPS, MUPIP, DSE and LKE reside in $gtm_dist. If the path to the executable and the path to $gtm_dist do not match each executable issues an error. When the path cannot be determined, each executable defers issuing an error until it is required. Previously it was possible for an executable path to be different from $gtm_dist. Such cases resulted in an errors or out-of-design situations. Call-in initialization verifies that $gtm_dist/gtmsecshr is usable. Previously the call-in initialization did not validate $gtm_dist or verify that $gtm_dist/gtmsecshr was usable, which could result in uncommitted data if the call-in need to invoke gtmsecshr. [UNIX] (GTM-7926) ✓

* GT.M no longer includes code for the deprecated RPC-based external call facility on Solaris. The UNIX external call functionality replaced the RPC-based interface many years ago. [Solaris] (GTM-7928)

* WRITE /WAIT on a SOCKET device with multiple sockets does not wait for new network events if there are pending events to be processed. Previously WRITE /WAIT returned only after a network event. Also in V6.1-000 on UNIX, WRITE /WAIT on a SOCKET device with multiple connected sockets would, in rare circumstances, cause a GTMASSERT2. (GTM-7929)

* The LITNONGRAPH warning message includes a $ZCHAR() representation of the non-printable character that caused the warning. Previously the message did not supply this context. (GTM-7931) ✓

* When ZLINK encounters a bad object file it produces an INVOBJFILE error that identifies the problem object file. Previously, it produced an INVOBJ error with no identification of the file in question. [UNIX](GTM-7942) ✓

* The 32-bit version of GT.M is now compiled to require the 586 ("Pentium compatible") instruction set. It previously required the 686 ("Pentium Pro compatible") instruction set. [x86 Linux] (GTM-7950) ✓

* $ORDER(,-1) or $ZPREVIOUS() of a gvn works correctly when the last subscript is a null string AND the subscripted gvn (not including the last null subscript) spans multiple regions AND the spanned region has a MAX_KEY_SIZE value greater than 998. Previously this resulted in a buffer overflow which might have caused the GT.M process to give unexpected results such as a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). (GTM-7953)

* GT.M safely processes XECUTEs and indirections following the setting of $ZSTATUS. Previously, the use of XECUTEs or indirection following the setting of $ZSTATUS could in rare cases result in indeterminate behavior. (GTM-7962)

* UNDEF errors while setting a subscripted lvn leave a clean state. Previously if one of the subscripts to an lvn being assigned using a SET, FOR, MERGE, READ or ZSHOW command or $INCREMENT() function was an undefined local variable, it was possible for a subsequent ZWRITE command or $QUERY() function of that lvn to terminate the process with a segmentation violation (UNIX SIG-11, or OpenVMS ACCVIO). In addition, UNDEF errors display the name of the local variable in all cases; previously, if the control variable in a FOR command had subscripts that were undefined local variables, the UNDEF error displayed an empty name. This fixes a regression introduced in GT.M V5.4-002B. (GTM-7963)

* When full Boolean processing is enabled, GT.M appropriately handles certain cases where an extrinsic returning a FALSE result appears as other than the first element in a Boolean expression. Previously, such a combination of code and condition could cause the process to loop indefinitely. (GTM-7971)

* m**n for negative m greater than or equal to -1.00010822724729531 and a large positive odd integer n with at least seven decimal digits returns a negative value; previously, it returned a positive rather than negative value. (GTM-7975)

* Specifying the combination of STREAM and NOWRAP for a Sequential Disk (SD) device, causes WRITE to neither truncate nor insert line-feed terminators; previously this combination, until recently documented to work as it now does, actually only did so if the RECORDSIZE was 65,535, otherwise when a single WRITE argument exceeded the RECORDSIZE, WRITE truncated it. WRITE in VARIABLE mode, which is the default mode for SD OPEN, truncates at WIDTH; previously it did not truncate in an edge case (when a WRITE happened at a $X such that $X+65535 was equal to the WIDTH). FIFO, SD and SOCKET devices opened as $PRINCIPAL (at process start-up) default to STREAM and NOWRAP; previously, $PRINCIPAL defaulted to VARIABLE for FIFO and SD, and to WRAP for all three of these devices. ZSHOW "D" for SD, FIFO and PIPE devices displays STREAM when it is in effect, as the default in the general case is VARIABLE; previously, ZSHOW "D" omitted STREAM. ZSHOW "D" for SOCKET devices displays NOWRAP when it is in effect, as the default is WRAP; previously, ZSHOW "D" omitted NOWRAP. MUPIP JOURNAL -EXTRACT, -RECOVER and -ROLLBACK produce complete records for journal extracts, and broken and lost transaction files; previously MUPIP truncated records that exceeded 32KiB. A shell pipe of GT.M and/or utilities (DSE or MUPIP) shows the actual output; previously, such an action truncated any output line that exceeded 32KiB. [UNIX] (GTM-7984)

* When VIEW "NOUNDEF" is in effect, ZWRITE ignores any local or global variable that is not defined. Previously, it issued a (GV)UNDEF error even when NOUNDEF was in effect. (GTM-7988)

* GT.M does not default values for individual capabilities missing from a terminfo entry specified by the environment variable TERM. Previously, GT.M provided values for missing capabilities. Because the default values could conflict with values for other capabilities of the specified terminfo, this caused issues such as arrow keys not working without terminfo adjustments. [UNIX] (GTM-7993)

* GT.M provides a mechanism to retain default line terminators for direct mode user interaction (including the BREAK command) independent of any TERMINATOR deviceparameter changes for $PRINCIPAL. VIEW "DMTERM" controls this behavior; with VIEW "NODMTERM" the TERMINATOR deviceparameter applies to both READs from $PRINCIPAL and direct mode interactions. A case-insensitive value of the environment variable gtm_dmterm is "1", "yes", or "true" establishes a NODMTERM state at process initiation; all other values, including no value, result in the default VIEW "NODMTERM" behavior. $VIEW("DMTERM") returns 1 for DMTERM mode or 0 for NODMTERM mode. [UNIX] (GTM-7994) ✓

* GT.M returns the correct values for $X and $Y for FIXED READs for Sequential Disk (SD), FIFO, and PIPE devioces in M mode and limits the number of characters read to WIDTH-$X in the current record. Note that $X may need to be set to zero prior to a FIXED READ or WRITE following a FIXED READ to maintain current FIXED READ or WRITE behavior. Subsequent to V5.2-000, $X and $Y returned zero for FIXED READs in M mode and these READs could cross record boundaries. [UNIX] (GTM-7997)

* The $ZSYSLOG() function sends its string parameter to the system log and always returns TRUE (1). The text appears in the syslog with the same format as any other GT.M syslog message (i.e. in the user.info log with GTM-MUMPS[pid]" or "GTM-MUPIP[pid]" prefix along with instance information where appropriate). [UNIX] (GTM-7999) ✓

* Ctrl-C terminates potentially long-running operations, for example, ZWRITE. Since V6.0-003, the revisions associated with GTM-5576 caused ZWRITE and ZSHOW "V" to not respond to <CTRL-C> in direct mode. Additionally, unlike utility programs, GT.M does not display informational messages unless they expand on another message, or you are in direct mode. Previously under rare circumstances, such informational messages could potentially appear in a running application. [UNIX](GTM-8003)

* GT.M no longer produces a segmentation violation (SIG-11) when a WRITE /WAIT selects a CONNECTED LOCAL socket with data for READ. [UNIX](GTM-8013)

* GT.M ignores values other than those documented for the gtm_boolean and gtm_side_effects environment variables; previously, undocumented values caused unspecified behavior including a GTMASSERT2. Also, $VIEW("FULL_BOOLEAN") reports "Standard Boolean evaluation side effects" when it is not explicitly set, but the setting of gtm_side_effects implicitly turns it on, and "Standard Boolean side-effect warning" when warnings have been specified; previously, $VIEW("FULL_BOOLEAN") reported the Boolean setting as if the side effect setting had no impact and did not include the word "Standard" when warnings were on. In addition, VIEW "NOFULLBOOLEAN" produces an error when gtm_side_effects is on; previously, this VIEW had no effect on subsequent compilations but failed to indicate that. (GTM-8015) ✓

* WRITE for a Sequential Disk (SD) device works at the current file position, whether attained with APPEND, REWIND or SEEK; previously WRITE required that the current position be at the end of file, and if not, produced a NOTTOEOFONPUT error. GT.M manages any BOM for UTF mode files by ensuring they are at the beginning of the file and produces a BOMMISMATCH error for an attempt to change the byte-ordering on OPEN for an existing file; previously, it could inappropriately write a BOM after an APPEND, required re-

READing the file from the beginning before certain actions, and did not detect byte-ordering conflicts on the OPEN. An attempt to OPEN a non-zero length file WRITEONLY without either NEWVERSION or TRUNCATE in UTF mode produces an OPENDEVFAIL due to the fact that any existing BOM information cannot be verified. Previously, such a file OPENed with APPEND could result in data WRITEn to the file in incorrect mode. Note that with GT.M SD encryption, because of the state information associated with encryption processing, encrypted files require the file to be WRITEn or READ from the beginning rather than from an arbitrary position. [UNIX](GTM-8018) ✓

* GT.M allows a WRITE to a zero length file for a sequential device after a USE REWIND; previously, in this circumstance GT.M issued a NOTTOEOFONPUT error. [UNIX] (GTM-8021)

* OPEN or USE for a SOCKET device can change CHSET only if the device doesn't have any sockets in it. Previously, unless the same CHSET was specified explicitly or implicitly on each OPEN for a SOCKET device, GT.M applied the most recent CHSET to all sockets in the device, including changing the CHSET to "UTF-8" if the command did not specify a CHSET. When OPEN or USE of a SOCKET device specifies one of the UTF-16 CHSETs the the command appropriately maintains the value of ZFF; previously, if a USE command specified ZFF before the first WRITE for a socket, it could assign a corrupt value. [UNIX] (GTM-8030)

* GT.M reports a COMMAORRPAREXP error when a patcode containing alternation is missing a right parenthesis at the end of the line, and a PATCODE error when that circumstance appears within a line. Previously it did not detect the end-of-line case, and could report an inappropriate PATCLASS error and a sub-optimal source code location indicator in the other case. (GTM-8036)

* $ZPEEK() function generates an UNDEF error when VIEW UNDEF is not set and the 4th $ZPEEK() argument (the optional format parameter) is specified but is undefined. Previously $ZPEEK() masked this situation by using the default value ("C"). [UNIX](GTM-8042) ✓

* GT.M creates a valid fatal ZSHOW dump file in case of a GTM-F-MEMORY run-time error. Previously, it was possible in very rare situations to create a corrupt ZSHOW dump file (e.g. variable subscripts or value incorrect, etc.) due the memory shortage occurring while attempting to rearrange the process heap space. Note there are still cases where an out-of-memory condition can cause an incomplete ZSHOW dump file (GTM-8047)

* A JOB command within a transaction bracketed by TSTART/TCOMMIT that has restarted enough to fallback to using preemptive (conventional) locking works appropriately. In GT.M V6.0-003 and V6.1-000, such a JOB command failed with a GTMASSERT due to a regression introduced as part of GTM-7680. As GT.M provides ACID transaction properties only for database updates and not non-database activities such as JOB commands, use of JOB commands that are not suitably protected (for example conditional on '$TRESTART) can result in additional JOBs being launched on TP retries. FIS stongly recommends designing TP transactions that avoid or at least minimize commands that violate ACID properties such as WRITE, ZSYSTEM or those that can have timeouts (JOB, LOCK, OPEN, READ). [UNIX] (GTM-8050)

* $KEY, $DEVICE, and $ZKEY return information about the input device of $PRINCIPAL when the input and output devices are different; previously, when the input was a terminal, Sequential Disk (SD), FIFO, or SOCKET device, and the output was a different device, these ISVs returned information about the output device. Also, the LISTEN and WAIT controlmnemonics operate on the input device of $PRINCIPAL when the input and output devices are different; previously, these controlmnemonics operated on the output device regardless of whether the $PRINCIPAL device type was SOCKET. [UNIX] (GTM-8052)

* CLOSE removes LOCAL socket files when operating on a LISTENING socket; previously, CLOSE required the DELETE and SOCKET=handle deviceparameters in order to remove a LOCAL socket file. [UNIX] (GTM-8055)

* GT.M creates a SOCKET device for $PRINCIPAL when standard input is a LOCAL domain socket and sets the default DELIMITER to "$CHAR(10)" for sockets in the device. Previously GT.M created a FIFO device for $PRINCIPAL in this case. [UNIX] (GTM-8056) ✓

* When a GT.M process starts, if any of stdin, stdout, or stderr (devices 0-2) lack an open file descriptor, GT.M defaults the device to /dev/null. Previously, GT.M produced DEVOPENFAIL, "Error in fstat", and "Bad file descriptor" messages for a missing stdin or stdout, but for a missing stderr it silently assumed that the next opened device descriptor (2) identified stderr, potentially leading to file corruption. [UNIX] (GTM-8063)

* Several seemingly unrelated & obscure defects introduced in V5.3-004 were remedied as follows:

  * Memory used by alias variables and alias containers is appropriately released. Previously, use of variables passed-through in exclusive NEWs and also specified in one or more of the TSTART variable lists in nested TP levels, resulted in potential memory leaks, premature variable removal, processing errors, and/or process failures due to a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO).

  * SET @x=y where the left-hand side of the SET is a subscripted local variable that is also an alias container and the destination evaluates to a subscripted local variable works correctly. Previously, the destination local variable incorrectly became an alias container which could later result in the process terminating unexpectedly with a segmentation violation (UNIX SIG-11 or VMS ACCVIO).

  * TP RESTART properly restores restart local variables. Previously if the transactions performed alias actions on such variables, RESTART could have given them incorrect values.

  * KILL * with no arguments and QUIT * maintain data cells appropriately. Previously, in the case of KILL *, when there were more than 8192 alias variables and alias containers, or, in the case of QUIT * when it encountered an error while returning its argument to the caller these commands leaked memory.

  * VIEW "LV_GCOL" command and $VIEW("LV_GCOL") function work properly. Previously, if there was a lot of orphaned alias data or they occurred in logic invoked via call-in to GT.M from an external language, they could result in a memory leak.

  (GTM-8064)

* The JOB command treats an undefined local variable as an UNDEF error, or, if in NOUNDEF mode, JOB treats it as an empty string. Previously JOB treated such arguments as missing. [UNIX](GTM-8065)

* $TEXT() and ZPRINT appropriately handle the case where another process concurrently overwrites a source file the operation is trying to acquire. Previously, this scenario could have caused a segmentation violation (SIG-11). [UNIX] (GTM-8085)

* $QUERY(lvn) handles three combinations of circumstances appropriately: (a) ZSHOW into a subscripted local variable where there was nothing to show, which could happen for codes "B" or "C"; (b) SET @x=y where the right hand side of the set was an undefined unsubscripted local variable, the left hand side evaluated to a subscripted local variable, the process was in NOUNDEF mode, and environment variable gtm_side_effects was non-zero; and (c) MERGE lvn(i)=^gvn inside a TP transaction where lvn(i) was a subscripted local variable and ^gvn was a subscripted or unsubscripted global variable and the ^gvn reference resulted in TP restarts due to concurrent database updates. Previously these situations could cause $QUERY(lvn) to encounter a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). Also, if the process is in UNDEF mode, SET @x=y issues an UNDEF error if x evaluates to a local variable (subscripted or unsubscripted) and y is an undefined unsubscripted local variable. Previously GT.M did the SET causing the local variable target to be set to an undefined value. (GTM-8108)

* A non-FIXED, FOLLOW READ x#n:t or READ *x:t set $TEST to TRUE (1) when the READ is fulfilled prior to the specified timeout (t); previously, such READs only set $TEST to TRUE if they encountered an End-of-Line. [UNIX](GTM-8109)

* Routine source code can be embedded in object code using the -[NO]EMBED_SOURCE qualifier of the mumps command, defaulting to NOEMBED_SOURCE. Like other GT.M compilation qualifiers, this qualifier can be specified through the $ZCOMPILE intrinsic special variable and gtmcompile environment variable. EMBED_SOURCE provides $TEXT and ZPRINT access to the correct source code, even if the original M source file has been edited or removed. Where the source code is not embedded in the object code, GT.M attempts to locate the source code file. If it cannot find source code matching the object code, $TEXT() returns a null string. ZPRINT prints whatever source code found and also prints a TXTSRCMAT message in direct mode; if it cannot find a source file, ZPRINT issues a FILENOTFND error. [UNIX] (GTM-8111)

* GT.M correctly handles run-time errors soon after a TP RESTART. Previously in this scenario, if subscripted local variable nodes were set before the TP RESTART, it was possible for GT.M to terminate abnormally with a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). (GTM-8114)

* GT.M appropriately manages the case where an exclusive NEW or BREAK command appears after a FOR command with a postconditional, the postconditional is not met, and local variables are newly referenced later in the line; previously, this could cause the process to terminate with a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). (GTM-8115)

* $TEXT() that is not within another transaction, with an argument specifying a trigger, and that encounters an error, returns an empty string; previously this combination could subsequently cause an inappropriate TPQUIT error. (GTM-8116)

* A JOB command has no effect on open non-terminal, non-SOCKET devices - FIFO, PIPE, and sequential device (SD). Previously a JOB command inserted a new-line character in the stream of any non-terminal devices with non-zero $X values. Our thanks to Nitin Mendiratta, who worked with us as an intern, for finding and fixing this. [UNIX] (GTM-8123)

**M-Other Than Database Access**

* When $PRINCIPAL identifies a device that supports REWIND, the REWIND or INREWIND device parameters perform a REWIND of the input and OUTREWIND performs a REWIND of the output. Previously, no REWIND of the output side of $PRINCIPAL was possible. [UNIX](GTM-8132) ✓

* The TRUNCATE device parameter on a USE $PRINCIPAL command works on a stdout device when the device supports the action; previously, USE ignored TRUNCATE for $PRINCIPAL. [UNIX](GTM-8133) ✓

* GT.M correctly operates when its $gtmroutines is set to a string containing '..' (two dots) and returns correct results from $zparse() executed on such expressions. Previously, $gtmroutines containing '..' resulted in ZROSYNTAX errors at GT.M process start-up, and null string returns from $zparse() when provided to it as an argument. [UNIX] (GTM-8141)

* ZHELP, and the utility HELP commands report misconfiguration issues without any secondary errors; previously misconfiguration could cause a %SYSTEM-E-ENO55808, Unknown error 55808 error. In addition, <CTRL-C> returns the session to the lowest level menu; previously it could enter into direct mode. Also, the help content is now current with V6.2-000. [UNIX](GTM-8145)

## Utilities-MUPIP

* MUPIP LOAD reports DBFILERR (Error with database file) error if the database is not present; previously, it terminated with a segmentation violation (UNIX SIG-11). Also, MUPIP LOAD reports a LDBINFMT (Corrupt binary format header information) error if the file is not a valid binary file; previously, it terminated with a either PREMATEOF or, in case of empty file, a segmentation violation (UNIX SIG-11). In addition, MUPIP LOAD accepts the -ONER[ROR]= qualifier with values of STOP, PROCEED or INTERACTIVE to control how MUPIP LOAD deals with an error: it respectively exits immediately, proceeds with the next record or prompts the user to continue or stop; By default, LOAD exits.[UNIX] (GTM-4250) ✓

* MUPIP ENDIANCVT now correctly recognizes directory tree leaf blocks. Previously, if a such a block lay at an offset approximately 4GB past the beginning of the database file and contained only one record, ENDIANVCT would incorrectly identify it as global variable tree block. ENDIANCVT would therefore leave the block pointer value unconverted, causing a subsequent MUPIP INTEG to fail. [UNIX] (GTM-7390)

* The REORGINC message is a warning, which aligns with the documentation; in addition, we reworded the message description. Previously, this message was an error, which did not match the documentation. (GTM-7816)

* When returning with an error, such as MUUSERLBK or MUUSERECOV, MUPIP RUNDOWN removes any semaphores it has created during its operation. Previously, in case of certain errors, MUPIP RUNDOWN left semaphores that showed up in messages from a subsequent RUNDOWN or UNIX IPCS report. [UNIX] (GTM-7859)

* MUPIP ROLLBACK and RUNDOWN appropriately handle an error subsequent to an initial error caused by use of a GT.M release other than that of the global directory or instance replication journal pool. Previously, this sequence caused a segmentation violation (SIG-11). [UNIX] (GTM-7938)

* If there is no need to update the former journal file, MUPIP SET -JOURNAL ignores the access permissions of that journal file; previously, MUPIP SET -JOURNAL gave JNLRDONLY error for an attempt to change the journal file path. In addition, MUPIP SET -JOURNAL -NOPREVJNLFILE cuts the journal file backlinks links as documented; previously, -NOPREVLINKS only worked with -JNLFILE. (GTM-7976)

* MUPIP EXTRACT accepts the -R[EGION]= qualifier, which specifies a comma separated list of region names that restricts EXTRACT to a set of regions; previously, EXTRACT operated on the set of globals in all regions defined by the current global directory. It was possible to achieve the same result as that provided by the -REGION qualifier by using custom global directories. Also, MUPIP BACKUP, EXTRACT, FREEZE, INTEG, RUNDOWN, SET and REORG treat region names as case-insensitive; previously, in UNIX, they required explicit upper-case, while in OpenVMS the DCL shell defaulted them to upper-case. [UNIX] (GTM-7990) ✓

* MUPIP SIZE produces adjacency information in its output and accepts the -ADJACENCY qualifier to control the range considered adjacent and MUPIP INTEG -FAST reports adjacency as well. Note that using scan mode for MUPIP SIZE, the results for levels other than 0 show adjacency for the next lower (one less) level. Additionally note that adjacency is only a proxy for database organization and its usefulness may be limited by the technology and configuration of your secondary storage. Previously, neither MUPIP INTEG -FAST nor MUPIP SIZE provided this information. [UNIX](GTM-7991) ✓

* MUPIP JOURNAL -ROLLBACK run standalone appropriately handles an error which causes an Instance Freeze; previously this scenario caused a shared memory leak. [UNIX] (GTM-7995)

* GT.M replication supports on-the-fly transition of an instance from a replicating instance to an originating instance (using the MUPIP REPLIC -SOURCE -ACTIVATE command). In versions GT.M V6.0-000 to V6.1-000 if the Instance Freeze scheme was enabled (i.e. gtm_custom_errors environment variable was set) AND there was at least one active GT.M process that had opened a database the instance was replicating, this transition failed with an ACTIVATEFAIL error. A workaround for those versions was for the GT.M process to attempt at least one update which generates a GTM-E-SCNDDBNOUPD error. [UNIX] (GTM-8023)

* A number of MUPIP commands use a documented NOREGION message when they cannot find a region. Previously they used undocumented text for these notices. (GTM-8072) ✓

* MUPIP JOURNAL -ROLLBACK -FETCHRESYNC works appropriately on a supplementary instance if the previous rollback terminated before completion. In GT.M versions V5.5-000 thru V6.1-000, the rollback in this scenario did more backward processing than necessary, which could result in a NOPREVLINK error (due to not enough journal file history available). As long as the journal files are available, the extended rollback did the right thing, even if it took much longer to do it. Note that this was an issue only if the previous rollback had abnormally terminated. [UNIX](GTM-8074)

* MUPIP INTEG -REGION on a list of regions gives an accurate count of Spanning Nodes in each region. Previously, if more than one region was specified, the Spanning Nodes and Blocks count printed after each region was incorrectly reproted as the cumulative sum of all the regions processed until then. [UNIX] (GTM-8084)

* The Source Server responds to user commands (for example MUPIP REPLIC -SOURCE -SHUTDOWN) in an order of seconds; previously, when reducing a large backlog, it could take on the order of minutes for it to respond. Also, after a MUPIP JOURNAL ROLLBACK, the Source Server minimizes reading from current journal files; previously the Source Server might unnecessarily repeatedly read the same portions of the journal file such that the impact had a direct correlation with alignsize. [UNIX] (GTM-8118)

* MUPIP RUNDOWN, with or without the -OVERRIDE qualifier, ignores the FILE_CORRUPT flag in the database file header. Previously if this flag was set (for example if a MUPIP JOURNAL ROLLBACK terminated with errors) MUPIP RUNDOWN issued a DBFLCORRP error which the documentation says to fix with DSE. However, in this case, DSE issued a REQRUNDOWN or REQRECOV or REQROLLBACK error making it problematic to recover from this situation. [UNIX] (GTM-8121)

* MUPIP EXTRACT handles large record counts (Key cnt: field in the RECORDSTAT message) appropriately; previously record counts overflowed at 2GiB, and on UNIX that caused MUPIP EXTRACT to issue an incorrect GTM-W-NOSELECT error and then delete the output file. The workaround was to break the EXTRACT up into multiple operations, none of which exceeded the implicit limit. (GTM-8128)

* The replication Source Server replicates reliably to a cross-endian Receiver server; previously, the Source Server sometimes inappropriately terminated with a REPLXENDIANFAIL error. [UNIX] (GTM-8155)

* MUPIP RUNDOWN with no arguments completely runs down orphaned journal pool shared memory segments (those left by an abnormal source server termination). Previously, in versions V6.0-002 through V6.1-000, argumentless RUNDOWN did an incomplete job, leaving a crash indicator set in the replication instance file, which resulted in a REPLREQROLLBACK error when restarting the source server. MUPIP RUNDOWN with no arguments does not run down journal/receive pool ipcs (shared memory segments and/or semaphores) in case the source server (or receiver server) terminated abnormally and other mumps processes (or update process in case of the receiver side) are still actively attached to it. Previously, in version V6.1-000, in this case, argumentless RUNDOWN incorrectly issued a confusing sequence of MUJPOOLRNDWNSUC, SEMREMOVED and MUJPOOLRNDWNFL messages for the jnlpool (or MURPOOLRNDWNSUC, SEMREMOVED and MURPOOLRNDWNFL messages for the receive pool) for the same replication instance file when it actually did not remove any ipc but cleared the corresponding fields in the replication instance file header, which resulted in a REPLREQROLLBACK error when restarting the source server (or receiver server) or a REPLINSTDBMATCH error when running a MUPIP JOURNAL ROLLBACK.[UNIX] (GTM-8156)

## Utilities-Other Than MUPIP

* DSE SAVE guards the array it uses to save block and issues a DSEMAXBLKSAV error message if you try to exceed its limit of 128 saves; previously, it did not, which left the possibility of a memory segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). DSE REMOVE -BLOCK guards its handling of the array it uses to manage saved blocks; previously it left the possibility of a memory segmentation violation. If there is only one version of the specified -FROM= block, DSE RESTORE defaults to that version and it always restores the original block transaction number; previously DSE RESTORE required -VERSION= in all cases, and always changed the transaction number. If there is only one version of the specified -BLOCK= block in the current region, DSE REMOVE defaults to that version; previously DSE REMOVE -BLOCK required -VERSION= in all cases. In addition, REMOVE -BLOCK announces the successful removal of a saved block; previously it did its work silently. DSE FIND -FREE and DSE RANGE accept bit maps as starting or ending points; previously, they did not. DSE MAP -BUSY and -MASTER accept bit map blocks; previously they did not. DSE REMOVE -BLOCK, RESTORE and SAVE -LIST all accept blocks higher than the current database size because they deal with set of saved block copies rather than the database and there are situations where a saved block may be outside the current database size (for example, due to a concurrent MUPIP REORG -TRUNCATE); previously, these commands were inconsistent in when they allowed selection of blocks larger than the database size. DSE SHIFT accepts a -BLOCK specification; previously it did not; DSE commands that accept block selections uniformly change the default block only if the block selection is valid for the command; previously, some commands changed the default to an invalid value or did not change the default. Some DSE messages are better documented and may differ from those used in prior versions. (GTM-230) ✓

* DSE FIND -REGION=<region-name> treats the region name as case-insensitive and uses documented error messages for issues with the specified region. Previously, DSE FIND -REGION required region names entered in upper-case, and its error messages were not documented and had slightly different wording. (GTM-6197)

* Setting the environment variable gtm_destdir directs the configure script to use a different path from the installation path. Use this environment variable in conjunction with the environment gtm_compile set to -embed_source to allow for pre-packaged installations. Previously open source partners relied on external patches to pre-package GT.M. [UNIX] (GTM-7934)

* DSE CACHE -SHOW displays the shared memory size as an 8-byte hexadecimal quantity. Previously, it displayed a 4-byte quantity, so if the database shared memory size was greater than 4GiB, it displayed a truncated (and incorrect) value. Additionally, if the database has encryption turned on, DSE CACHE -SHOW displays an element giving information about the encrypted global buffer section in shared memory; previously it omitted this information. (GTM-7936) ✓

* GDE SHOW -COMMANDS generates the GDE commands in the correct order. Previously when the GDE SHOW -COMMANDS output was fed to a fresh GDE session, it was possible to get a NAMRANGEORDER error in cases where a GBLNAME object had a non-zero alternate collation OR to get a KEYSIZIS/KEYFORBLK/OBJNOTCHNG error in cases where the KEY_SIZE value of any region was greater than 984. (GTM-7954)

* DSE DUMP -GLO and DUMP -ZWR display appropriate output for unsubscripted global notes. Previously they incorrectly displayed a spurious subscript, for example: ^x(-000000000000000000000000000000000000000000000000000000000000000) but displayed the correct node value. (GTM-7983)

* GDE SHOW -MAP -REG handles the case where "Up to" column of the map entry contains non-printable characters in the subscripts or corresponds to a ++ notation. Previously, in these cases, GDE either printed garbage subscripts or failed with a GDECHECK error. Note that GDE SHOW -MAP without the -REG qualifier was not affected by this issue, and can be used as a workaround in previous versions. (GTM-8006)

* ^%RANDSTR returns an empty string if its arguments specify a zero-length result. Previously specifying a zero-length result produced an UNDEF error. (GTM-8048)

* %XCMD uses any current $ETRAP, expected to be the one defined by the environment variable $gtm_etrap when executing MUMPS code. Previously %XCMD overrode $ETRAP with its internal error handler without regard for the $gtm_etrap setting. [UNIX] (GTM-8100)

* The open source release kit instructions correctly place CMake options ahead of source paths. Previously the CMake usage listed in the instructions resulted in an error. [UNIX] (GTM-8147)

**Utilities-Other Than MUPIP**

* DSE FIND -EXHAUSTIVE locates blocks that are in a tree even if the itself block is badly damaged; previously some such blocks caused the following error: No keys in block, cannot perform ordered search. (GTM-8154)

* The gtminstall script extends the --verbose qualifier to report problems with downloading missing GT.M components from repositories. Such messages are helpful in diagnosing proxy issues. Prior versions did not provide these messages. [UNIX](GTM-8159)

## More Information

### Additional Information for GTM-7449 - Encrypted File IO Plugin Reference Implementation

GT.M can use any encryption plugin that conforms to the specified API; see Encryption Plugin below. There is no requirement to use the reference implementation of the encryption included with the GT.M distribution, or a even a modified version thereof. Note that GT.M expects hashes of symmetric keys as provided by the plugin in gtmcrypt_to obtain_db_key_hash_by_keyname() (described in the API below) and subsequently pass them back to the plugin in gtmcrypt_init_db_cipher_context_by_hash() to be exactly 512-bits (64 bytes). If you customize the plugin to use a shorter hash, you should ensure that the customization continues to pass 512-bits, and that unused bits are either preset to a known value or are ignored.

### Reference Implementation

The reference implementation of the encryption plugin ignores the gtm_dbkeys environment variable. Device and database encryption, as well as TLS, require a libconfig format configuration file referenced by the gtmcrypt_config environment variable. The libconfig tool is available form http://www.hyperrealm.com/libconfig/ and for Linux users likely via the package manager of their distribution) In addition to other sections, a "files" section in the configuration file maps key names to keys:

```
files: {
<keyname1>: "<keypath1>";
<keyname2>: "<keypath2>";
...
}
```

Fieldnames, keyname1 and keyname2 in the example above, denote key names that can be referenced in M code, while the corresponding values, keypath1 and keypath 2, specify paths to files containing the actual keys to symmetric ciphers. A key needs to be specified only once in the configuration file; any number of files or other devices can use that key simply by naming it in deviceparameters. As with keys used for database encryption, the symmetric cipher keys in the files are encrypted using a public key whose corresponding private key is available in the GnuPG keyring, which is in turn secured with a passphrase supplied to the GT.M process via the gtm_passwd environment variable, as described in Chapter 12, Database Encryption, of the Administration and Operations Guide UNIX Edition.

See the example in the GTM-7449 release note a sample of a files section.

FIS tests the reference implementation of the plugin using the following: (list ciphers and libraries here, including versions on each platform).

The ciphers against which FIS tests the reference implementation require an IV of exactly 16 bytes. If the length of a provided IV is less than 16 bytes, the reference implementation pads it with NULL bytes at the end. The reference implementation generates an error if the IV is longer than 16 bytes.

### Encryption Plugin API

**Note**

The GT.M encryption plugin API has undergone considerable interface and behavioral changes. Please read this section carefully if you have customized the plugin.

As there is no change to database format, applications currently using databases encrypted using a prior version of GT.M with the same database format as V6.2-000, and an unmodified reference implementations of the plugin supplied with that version of GT.M, operate with V6.2-000 and the unmodified version of the reference implementation of the plugin supplied with V6.2-000.

**More Information**

The interface includes the following functions:

```
/*
 * Initialize encryption if not yet initialized. Use this function to load neccessary libraries and set appropriate
 configuration
 * options. Upon a successful return this function is never invoked again.
 *
 * Arguments: flags Encryption flags to use.
 *
 * Returns: 0 if encryption was initialized successfully; -1 otherwise.
 */
gtm_status_t gtmcrypt_init(gtm_int_t flags);

/*
 * Return the error string. Use this function to provide the current error status. The function is normally invoked
 following a
 * non-zero return from one of the other functions defined in the interface, which means that each of them should
 start by clearing
 * the error buffer.
 *
 * Returns: The error string constructed so far.
 */
gtm_char_t *gtmcrypt_strerror(void);

/*
 * Find the key by hash and set up database encryption and decryption state objects, if not created yet. Use this
 function to locate
 * a particular key by its hash and, if found, initialize the objects for subsequent encryption and decryption
 operations on any
 * database that will use this key, unless already initialized. The reason any database relying on the same key may
 use the same
 * encryption and decryption state objects is that for every encryption and decryption operation the initial IV is
 used, effectively
 * reverting to the original state.
 *
 * Arguments:   handle  Pointer to the database encryption state object supplied by the caller and filled in by
 this routine.
 *              hash    Hash of the key.
 *              iv      Initialization vector to use for encryption or decryption.
 *
 * Returns:     0 if the routine found the key, and either found existing database encryption and decryption state
 objects or initialized them;
 *              -1 otherwise.
 */
gtm_status_t    gtmcrypt_init_db_cipher_context_by_hash(gtmcrypt_key_t *handle, gtm_string_t hash, gtm_string_t
 iv);

/*
 * Find the key by keyname and set up device encryption or decryption state object. Use this function to locate a
 particular key by
 * its name (as specified in the configuration file) and, if found, initialize an object for subsequent encryption
 or decryption
 * operations (depending on the 'encrypt' parameter) with one device using this key. Note that, unlike databases,
 different devices
 * relying on the same key require individual encryption and decryption state objects as their states evolve with
 each encryption or
 * decryption operation.
 *
 * Arguments:   handle          Pointer to the database encryption state object supplied by the caller and filled
 in by this routine.
 *              keyname         Name of the key.
 *              iv              Initialization vector to use for encryption or decryption.
 *              operation       Flag indicating whether encryption or decryption is desired; use
 GTMCRYPT_OP_ENCRYPT or
 *                              GTMCRYPT_OP_DECRYPT, respectively.
 *
 * Returns:     0 if the routine found the key, and either found existing database encryption and decryption state
 objects or initialized them;
 *              -1 otherwise.
 */
gtm_status_t    gtmcrypt_init_device_cipher_context_by_keyname(gtmcrypt_key_t *handle, gtm_string_t keyname,
                                           gtm_string_t iv, gtm_int_t operation);
/*
 * Find the key by keyname and obtain its hash. Use this function to locate a particular key by its name and
 calculate (or copy, if
 * precalculated) its hash to the 'hash_dest' address. Note that the keyname corresponds to a particular 'files'
 field in the
 * configuration file in case of devices, or a path to a database file otherwise.
 *
 * Arguments:   keyname         Name of the key.
 *              hash_dest       Pointer to the location for this routine to copy the key's hash.
 *
 * Returns:     0 if the routine found the key and copied its hash to the specified location;
 *              -1 otherwise.
 */
gtm_status_t    gtmcrypt_obtain_db_key_hash_by_keyname(gtm_string_t keyname, gtm_string_t *hash_dest);

/*
 * Release the specified encryption or decryption state object, also releasing the decryption state if database
 encryption state is
 * specified.
 *
 * Arguments:   handle          Pointer to the encryption or decryption state object to release.
 *
 * Returns:     0               if the operation succeeded; -1 otherwise.
 */
gtm_status_t    gtmcrypt_release_key(gtmcrypt_key_t handle);

/*
 * Perform encryption or decryption of the provided data based on the specified encryption / decryption state. If
 the target buffer
 * pointer is NULL, the operation is done in-place.
 *
 * It is also possible to set the initialization vector (IV) to a particular value, or reset it to the original
 value, before
 * attempting the operation. The results of mixing different IV modes on the *same* encryption / decryption state
 object are
 * different between OpenSSL and Gcrypt, though. The difference is that modifying the IV (iv_mode !=
 GTMCRYPT_IV_CONTINUE) with
 * OpenSSL does not affect the actual encryption / decryption state, and subsequent IV-non-modifying encryptions /
 decryptions
 * (iv_mode == GTMCRYPT_IV_CONTINUE) are performed on whatever state the prior IV-non-modifying encryptions /
 decryptions arrived
 * at. With Gcrypt, on the other hand, modifying the IV (iv_mode != GTMCRYPT_IV_CONTINUE) before an operation
 influences the
 * subsequent IV-non-modifying (iv_mode == GTMCRYPT_IV_CONTINUE) operations.
 *
 * Arguments:   handle                  Encryption state object.
 *              unencr_block            Block of unencrypted data.
 *              unencr_block_len        Length of the unencrypted and encrypted data blocks.
 *              encr_block              Block of encrypted data.
 *              operation               Flag indicating whether to perform encryption or decryption; use
 GTMCRYPT_OP_ENCRYPT or
 *                                      GTMCRYPT_OP_DECRYPT, respectively.
 *              iv_mode                 Flag indicating whether to change the initialization vector (IV) prior to
 the
 *                                      operation; use GTMCRYPT_IV_CONTINUE to proceed without changing the IV,
 GTMCRYPT_IV_SET to
 *                                      set the IV the value supplied in the iv argument, and GTMCRYPT_IV_RESET to
 reset the IV to
 *                                      the value specified at initialization.
 *              iv                      Initialization vector for the encryption state to when iv_mode is
 GTMCRYPT_IV_SET.
 *
 * Returns:     0 if the operation succeeded; -1 otherwise.
 */
gtm_status_t    gtmcrypt_encrypt_decrypt(gtmcrypt_key_t handle, gtm_char_t *src_block, gtm_int_t src_block_len,
                                         gtm_char_t *dest_block, gtm_int_t operation, gtm_int_t iv_mode,
 gtm_string_t iv);

/*
 * Compare the keys associated with two encryption or decryption state objects.
 *
 * Arguments:   handle1         First encryption or decryption state object.
 *              handle2         Second encryption or decryption state object.
 *
 * Returns:     1               if both encryption or decryption state objects use the same key; 0 otherwise.
 */
gtm_int_t       gtmcrypt_same_key(gtmcrypt_key_t handle1, gtmcrypt_key_t handle2);

/*
 * Disable encryption and discard any sensitive data in memory.
 *
 * Returns:     0 if the operation succeeded; -1 otherwise.
 */
gtm_status_t    gtmcrypt_close(void);
```

Changes to the toolset are limited to the add_db_key.sh script, which is not present in the new version due to the elimination of $gtm_dbkeys and database-mapping file support.

Other enhancements in the encryption plug-in version include:

* Lowercase hexadecimal characters in $gtm_passwd are allowed.
* Undetected non-hexadecimal characters in, and odd length of, $gtm_passwd produce an appropriate error.
* Various preexisting messages better explain the error conditions at hand. GTM limits individual message arguments to 256 characters, adding trailing ellipses to indicate the truncation of the argument.

**More Information**

* GT.M checks the last modified date of a configuration file using nanoseconds resolution (and precision as provided by the OS), thus preventing errors that could result from multiple configuration file updates performed within one second (for instance, by a script).

* An empty last array entry in the 'database.keys' section of the configuration file is not required and does not cause an error.

* Improvements in the management of symmetric cipher keys, including (but not limited to) more efficient memory by eliminating the duplication of stored information for all databases using a particular key, and eliminating unsuccessful key lookups even in the face of prior errors encountered while parsing the configuration file.

* Faster key lookups, resulting in higher performance with encrypted databases than previously.

* Various previously undetected and / or unhandled faulty conditions, such as failed encryption or decryption, now raise proper errors.

## Error and Other Messages

### ACTIVATEFAIL

**ACTIVATEFAIL ,** Cannot activate passive source server on instance iiii while a receiver server and/or update process is running

MUPIP Error: MUPIP REPLIC -SOURCE -ACTIVATE -ROOTPRIMARY (or -UPDOK) issues this error when the command attempts to activate a passive source server (and switch the instance from being a replicating secondary instance to an originating primary) while a receiver server and/or update process is already running

Action: Shutdown the receiver and/or update process and reissue the MUPIP REPLIC -SOURCE -ACTIVATE -ROOTPRIMARY (or -UPDOK) command. Note that any other GT.M or MUPIP process that was running before the activation does not need to be shut down for the activation to succeed.

### AIMGBLKFAIL

**AIMGBLKFAIL,** After image build for block bbbb in region rrrr failed in DSE or MUPIP

MUPIP/DSE/GT.CM Error: DSE creates after images of blocks as a result of its physical manipulation of blocks and in MUPIP processes them in the course of RECOVER or ROLLBACK. This error indicates that such a manipulation failed on the block and region indicated.

Action: If you get this error from DSE, you may be working with a block with a damaged state that your DSE action does not sufficiently address - analyze the situation and consider other approaches. If you get this error from MUPIP, it may mean your journal or replication has damage, in which case you should investigate the state of block bbbb.

### BLKINVALID

**BLKINVALID,** bbbb is not a valid block as database file ffff has nnnn total blocks

DSE Error: The block (bbbb) you selected is not currently a valid block in the database file (ffff) for the region in which you are working.

Action: Select a block less than nnnn or move to a different region

### CANTBITMAP

**CANTBITMAP,** Can't perform this operation on a bit map (block at a 200 hexadecimal boundary)

DSE Error: The selected DSE operation does not apply to bit maps (blocks divisible by 0x200).

Action: Select an appropriate block for the operation or an appropriate operation for a bit map.

### CHSETALREADY

**CHSETALREADY,** CHSET xxxx already specified for socket device

Run Time Error: The code specified a CHSET for a SOCKET device different than the CHSET previously assigned to it.

Action: If different CHSETs are needed for different sockets, place them in different SOCKET devices.

### COLTRANSSTR2LONG

**COLTRANSSTR2LONG,**Output string after collation transformation is too long

Run-time Error: an alternative collation transform or reverse transform attempted to use more bytes than the configuration permits.

Action: Adjust the implementation of the collation transform to minimize key expansion; increase the maximum permitted key size if appropriate. Note the current supported maximum is 1019 bytes. If the key size is already maxed out and the transformation algorithm is optimal, you must modify the application to reduce the key size.

## CONNSOCKREQ

**CONNSOCKREQ,** Socket not connected

Run Time Error: The operation attempted requires a socket in the CONNECTED state, and the provided socket was not connected.

Action: Make sure the correct socket is being used and that the socket is connected. ZSHOW "D" may provide useful details on the current socket state.

## CORRUPTNODE

**CORRUPTNODE,** Corrupt input in Record # rrrr, Key #yyyy; resuming with next global node

MUPIP Error: This message reports record rrrr with apparent key kkkk does not have a valid format for MUPIP LOAD.

Action: USE %GO or MUPIP EXTRACT to recapture the problematic node(s), or, use an editor to create valid copies of the nodes in an LOAD file.

## CREDNOTPASSED

**CREDNOTPASSED,** Socket message contained no passed credentials

Run Time Error: WRITE /PASS or WRITE /ACCEPT was given a process id to verify, but GT.M was unable to obtain the peerprocess id.

Action: See the accompanying ENO error for details.

## CRYPTBADWRTPOS

**CRYPTBADWRTPOS,** Encrypted WRITE disallowed from a position different than where the last WRITE completed

Run-time Error: A WRITE attempt to an encrypted device violates the integrity of the produced ciphertext. This is the case, for example, when trying to WRITE to a previously encrypted and CLOSEd file. Because encryption ciphers rely on state machine algorithms, GT.M prohibits WRITEs performed in non-sequential fashion or when they threaten to overlay already encrypted data.

Action: Revise your M code to avoid illegal I/O operations with encryption. Note, in particular, that when using encryption non-empty files cannot be opened in APPEND mode; the SEEK deviceparameter is prohibited; and the TRUNCATE is only permitted at the beginning or end of a file.

## CRYPTKEYTOOBIG

**CRYPTKEYTOOBIG,** Specified key has length xxxx, which is greater than the maximum allowed key length yyyy

Run-time Error: A key name value specified with the [I|O]KEY deviceparameter on an OPEN or USE command is too long.

Action: Verify that the key name portion of the [I|O]KEY deviceparameter's value (substring before the first space, if any) corresponds to an existing field name in the 'database.keys' or 'files' section of the configuration file and does not exceed yyyy characters in length.

## CRYPTNOAPPEND

**CRYPTNOAPPEND,** APPEND disallowed on the encrypted file xxxx

Run-time Error: An OPEN command specifies both an APPEND deviceparameter and a non-empty value for the [I|O]KEY deviceparameter.

Action: Because encryption algorithms maintain state as they process text, APPENDing encrypted data to a non-empty file is prohibited; revise your application code accordingly.

## CRYPTNOKEYSPEC

**,CRYPTNOKEYSPEC** Key name needs to be specified with KEY, IKEY, or OKEY device parameter for encrypted I/O

Run-time Error: A key name value specified with the [I|O]KEY deviceparameter on an OPEN or USE command is empty while the initialization vector (IV) is not.

Action: If enabling encryption or modifying encryption attributes, be sure to include an appropriate key name; if disabling encryption, leave the IV portion of the [I|O]KEY deviceparameter's value (substring after the first space) empty.

## CRYPTNOOVERRIDE

**CRYPTNOOVERRIDE,** Cannot override IVEC and/or key without compromising integrity

Run-time Error: An OPEN or USE command attempts to change the encryption attributes - that is: attempted to enable or disable encryption or change the key name, initialization vector (IV), or both, after a prior encrypted READ or WRITE.

Action: Because encryption algorithms maintain state as they process text, changing encryption attributes of a device is prohibited if an encrypted READ or WRITE has already occurred, so revise your application code accordingly.

## CRYPTNOSEEK

**CRYPTNOSEEK,** SEEK disallowed on the encrypted file ffff

Run-time Error: An OPEN or USE command specifies a SEEK deviceparameter on an encryption-enabled device.

Action: Because encryption algorithms maintain state as they process text, SEEKs are prohibited with encrypted devices, so revise your application code accordingly.

## CRYPTNOTRUNC

**CRYPTNOTRUNC,** Not positioned at file start or EOF. TRUNCATE disallowed on the encrypted file ffff

Run-time Error: An OPEN or USE command specifies a TRUNCATE deviceparameter on a encryption-enabled device which is not positioned at the end of a file.

Action: When using encryption, because encryption algorithms maintain state as they process text, a TRUNCATE is only permitted at the beginning or end of a file, the former deleting the entire contents, and the latter effectively a no-op.

## DBCOLLREQ

*DBCOLLREQ,* JOURNAL EXTRACT proceeding without collation information for globals in database. eeeee fffff .

MUPIP Warning: This is MUPIP JOURNAL EXTRACT Warning. This indicates that MUPIP process uses the default collation as it is not able to read the database file fffff because of eeeee

Action: Be aware that if the EXTRACT contains variables with alternative collation that this extract represents them as GT.M stores them, rather than as they are used by the application. Attempting to LOAD such an EXTRACT will produce incorrect results.

## DSEMAXBLKSAV

**DSEMAXBLKSAV,** DSE cannot SAVE another block as it already has the maximum of mmmm

DSE Error: The current SAVE -BLOCK operation exceeds DSE's capacity to hold more than mmmm saved blocks

Action: Delete some saved blocks, possibly after RESTORE'ng them to free blocks, or restart DSE if none of the currently saved blocks have value.

## DSENOTOPEN

**DSENOTOPEN,** DSE could not open region rrrr - see DSE startup error message for cause

DSE Error: DSE could not operate on region rrrr because it was not able to open it when DSE started.

Action: Review the error messages issued when DSE started and address the issue(s) they describe.

## GTMDISTUNVERIF

**GTMDISTUNVERIF,** Environment variable $gtm_dist (dddd) could not be verified against the executables path (pppp)

Run Time/MUPIP/LKE/DSE/GT.CM Error: This indicates that the executable pppp does not resides in the path pointed to by environment variable gtm_dist, dddd.

Action: Ensure that the setting for $gtm_dist matches that of the executable.

## INSNOTJOINED

**INSNOTJOINED,** Replicating Instance RRRR is not a member of the same Group as Instance IIII

MUPIP Error: A Receiver Server or a MUPIP JOURNAL -ROLLBACK -FETCHRESYNC on instance RRRR produces this error when it attempts to establish a replication connection with an instance that belongs to a different replication configuration or a Group. MUPIP performs this safety check at the time it establishes a replication connection between two instances.

Action: Use the Remote IP Address in the Receiver / Source Server log files or the primary instance name field from MUPIP REPLICATE -JNLPOOL -SHOW command to identify the Source Server that may have inadvertently attempted to establish a replication connection with your Source Server. Shut down the Source Server if the Source Server does not belong to your replication configuration. If you are attempting to move a Source Server from a different Group, reinitialize the Source Server.

Note that only supplementary instances started with -UPDOK can accept updates from a different Group.

## INVLINKTMPDIR

**INVLINKTMPDIR,** Value for $gtm_linktmpdir is either not found or not a directory: dddd

Run-time Error: Indicates the process cannot access directory dddd, which it needs in order to do auto-relink as specified by its $ZROUTINES; the directory may not exist as a directory or the process lacks authorization to the directory.

Action: The directory specification comes from $gtm_linktmpdir if it is defined, otherwise from $gtm_tmp if that is defined; otherwise it defaults to the system temporary directory, typically /tmp. Either correct the environment variable definition or ensure directory dddd is appropriately set up. Note that all users of auto-relink for a directory normally need to use the same temporary directory for their relink control files.

## INVOBJFILE

**INVOBJFILE,** Cannot ZLINK object file ffff due to unexpected format

Run Time Error: This indicates that ZLINK encountered invalid records in the object file ffff it was trying to integrate into the image.

Action: Determine whether ZLINK has the intended argument. If the object file has been damaged, recreate it with a ZLINK that specifies the source file using an .M extension, a ZCOMPILE or a mumps command at the shell.

## KEYWRDBAD

*KEYWRDBAD, xxxx is not a valid yyyy in this context*

GDE Error: This indicates that GDE did not encounter a valid syntax element. xxxx is the invalid element. yyyy designates whether the element in context is a verb (command), object, or qualifier.

Action: Look for and correct typographical errors.

## LABELNOTFND

**LABELNOTFND,** GOTO referenced a label that does not exist

Run-time Error: A GOTO referenced a label with neither a routine nor an offset but that label does not currently exist in the current routine. The location that accompanies this message is the last line in the routine.

Action: Check the errors from the compilation, as they provide the name of the missing label. As appropriate add the label or a routine, or better yet refactor to remove the GOTO.

## LDSPANGLOINCMP

**LDSPANGLOINCMP,** Incomplete spanning node found during load!/!_!_at File offset : oooo

MUPIP Error: This error indicates that MUPIP LOAD encountered an issue with a spanning node in the input file at offset oooo. MUPIP LOAD produces the following LDSPANGLOINCMP errors:

* **Expected chunk number : ccccc but found a non-spanning node**
* **Expected chunk number : ccccc but found chunk number : ddddd**
* **Not expecting a spanning node chunk but found chunk : ccccc**
* **Global value too large: expected size : sssss actual size : tttttt chunk number : ccccc**
* **Expected size : sssss actual size : ttttt**

Action: Refer to the LDSPANGLOINCMP Errors section in the "Maintaining Database Integrity" chapter of the Administration and Operations Guide

## LITNONGRAPH

**LITNONGRAPH,** standard requires graphics in string literals; found non-printable: $ZCHAR(cccc)

Compile-time warning: flags a standard violation. The generated code will accept the string, even though it contains cccc, which is not a visible character.

Action: Consider revising the literal to use $[Z]CHAR() and possibly concatenation to make the code more maintainable.

## LOCALSOCKREQ

**LOCALSOCKREQ,** LOCAL socket required

Run Time Error: The operation attempted requires a LOCAL socket, and a non-LOCAL (TCP) socket was specified.

Action: Make sure the correct socket is being used and that the socket is OPENed with the ":LOCAL" suffix. ZSHOW "D"may provide useful details on the current socket state.

## MAXBTLEVEL

**MAXBTLEVEL,** Global ^gggg in region rrrr reached maximum level

Run-time or MUPIP Error: This indicates that the global-variable-tree for global xxxx reached the maximum level permissible. Very likely, MUPIP REORG was specified with a fill-factor much less than 100. Small fill-factors can cause REORG to revise existing GDS-blocks (in order to accommodate the fill-factor requirement), in turn causing block-splits, which might lead to an increase of the tree height. Alternatively a SET or MERGE has made the global really too large for the current block size, which is most likely to happen with large (spanning) database nodes. Note that if this message does not specify the global name, it means the directory tree for the region hit the limit - FIS believes the directory tree full condition is almost impossible to create in practice.

Action: If MUPIP reorg was specified with a small fill-factor, try higher number (close to 100) to reduce tree-height. Other techniques include increasing GDS-block-size, reducing reserved bytes, killing unwanted portions of the tree or moving some nodes in the global to another place.

## NOGTCMDB

**NOGTCMDB,** ffff does not support operation on GT.CM database region: rrrr

GT.M Utility Error: Facility ffff cannot perform the requested operation on a GT.CM database such as rrrr.

Action: Use the utility on the remote GT.CM server system or move the database so it is local rather than remote.

## NOSOCKHANDLE

**NOSOCKHANDLE,** No socket handle specified in WRITE /PASS

Run Time Error: WRITE /PASS was called without specifying at least one socket handle to pass.

Action: Make sure the code is specifying at least one socket handle.

## NOTGBL

**NOTGBL,** Expected a global variable name starting with an up-arrow (^):

Run Time/MUPIP Error: This indicates that VIEW argument expression for tracing is not a valid global name. In case of MUPIP error, it indicates that LOAD aborted because it encountered xxxx in its input stream, which is not a valid global name.

Action: Correct the argument of the VIEW command to point to a valid global name. For MUPIP error, refer to the topic MUPIP LOAD Errors in About This Manual section of this manual.

<indexterm><primary>MUPIP</primary><secondary>NOTGBL</secondary></indexterm>

## NOUSERDB

**NOUSERDB,** ffff does not support operation on non-GDS format region: rrrr

GT.M Utility Error: Facility ffff cannot perform the requested operation on a user-defined database such as rrrr.

Action: Convert the database to a GDS format or use the utilities appropriate to the user-defined format.

## PEERPIDMISMATCH

**PEERPIDMISMATCH,** Local socket peer with PID=pppp does not match specified PID=qqqq

Run Time Error: WRITE /PASS or WRITE /ACCEPT was given a process id qqqq to verify, but the connection peer process id is pppp.

Action: Make sure that only the specified process has opened the socket connection.

## RECLOAD

**RECLOAD,** Error loading record number: nnnn

MUPIP Error: This message identifies a record that MUPIP could not LOAD and follows a message about the cause.

Action: Address the cause or, for GO and ZWR format input files, examine the record with a text editor for possible correction or alternate action and for BIN format if fixing the cause does not resolve the error switch to ZWR format EXTRACT.

## RELINKCTLERR

**RELINKCTLERR,** Error with relink control structure for $ZROUTINES directory dddd

Run-time Error: Indicates a problem accessing a relink control file in the temporary directory typically specified by the gtm_linktmpdir environment variable.

Action: Use the accompanying message(s) for a detailed error status to diagnose and address the access issue.

## REORGINC

**REORGINC,** Reorg was incomplete. Not all globals were reorged.

MUPIP Warning: This indicates that MUPIP did not reorg all the globals because of some resource constraint errors.

Action: Review the accompanying message(s) for more information.

## REPLINSTNOSHM

*REPLINSTNOSHM, Database dddd has no active connection to a replication journal pool*

Run Time Error: The Source server was started with a repication instance that had this database file listed but later the source server and this particular database file was shut down while other database files in this instance file were still active.

Action: To recover from this, restart the source server.

## SDSEEKERR

**SDSEEKERR,** Sequential device seek error

Run Time Error: This indicates that a GT.M process encountered an error using the SEEK deviceparameter for an OPEN or USE on a sequential disk device. A supplementary TEXT message provides more details about the cause of the error.

Action: Analyze the accompanying message and appropriately adjust the SEEK deviceparameter or its value.

## SEFCTNEEDSFULLB

**SEFCTNEEDSFULLB,** Current side effect setting does not permit full Boolean to be turned off

Run-time Error: A VIEW "NOFULL_BOOLEAN" cannot enable GT.M short-circuit Boolean compilation for a process running with a gtm_side_effects setting of 1 or 2.

Action: Keeping in mind that gtm_boolean and gtm_side_effects affect compilation behavior, and that gtm_boolean must be 1 or 2 (Standard Boolean mode) for gtm_side_effects setting 1 or 2 (Standard side effects), choose appropriate compilation modes. Note that once you choose the modes for your application you would typically not change them except to get warnings and modify the application to be somewhat more efficient.

## SOCKACCEPT

**SOCKACCEPT,** Socket accept failed

Run Time Error: WRITE /ACCEPT encountered an I/O error while attempting to accept the sockets. No sockets were added to the socket pool.

Action: See the accompanying ENO or TEXT message for details.

## SOCKNOTPASSED

**SOCKNOTPASSED,** Socket message contained no passed socket descriptors

Run Time Error: WRITE /ACCEPT received no sockets over the LOCAL connection.

Action: Verify the connection and make sure the WRITE /PASS on the sender is correct.

## SOCKPASS

**SOCKPASS,** Socket pass failed

Run Time Error: WRITE /PASS encountered an I/O error while attempting to pass the sockets. No sockets were closed.

Action: See the accompanying ENO or TEXT message for details.

## SOCKPASSDATAMIX

**SOCKPASSDATAMIX,** Attempt to use a LOCAL socket for both READ/WRITE and PASS/ACCEPT

Run Time Error: The code attempted to use a LOCAL socket for both data communication, using READ and/or WRITE, and socket passing, using WRITE /PASS or WRITE /ACCEPT. Using both forms of communication on the same socket is not supported.

Action: Use separate sockets for data and socket passing.

## TPRESTART ⚠

**TPRESTART,** Database mmmm; code: xxxx; blk: yyyy in glbl: zzzz; pvtmods: aaaa, blkmods: bbbb, blklvl: cccc, type: dddd, readset: eeee, writeset: ffff, local_tn: gggg

Severity: Information

Run Time Information: The UNIX environment variables or OpenVMS logical names GTM_TPRESTART_LOG_FIRST and GTM_TPRESTART_LOG_DELTA control the logging of TPRESTART messages. GTM_TPRESTART_LOG_FIRST indicates the number of TP restarts to log from GT.M invocation. Once that many have been logged, every GTM_TPRESTART_LOG_DELTA TP restarts, GT.M logs a restart message. If GTM_TPRESTART_LOG_DELTA is undefined, GT.M performs no operator logging. The default value for GTM_TPRESTART_LOG_FIRST is 0 (zero), which leaves the control completely with GTM_TPRESTART_LOG_DELTA. This message can serve as a diagnostic tool in developmental environments for investigating contention due to global updates. A zzzz of "*BITMAP" indicates contention in block allocation which might involve multiple globals.

Action: Disable, or adjust the frequency of, these messages with the mechanism described above. To reduce the number of restarts, consider changes to the global structure, varying the time when work is scheduled. Consider whether the business and program logic permits the use of NOISOLATION.

## TRIGLOADFAIL ⊕

**TRIGLOADFAIL,** MUPIP TRIGGER or $ZTRIGGER operation failed. Failure code: xxxx

Run Time or MUPIP Error: This indicates that a trigger install (using $ZTRIGGER() or MUPIP TRIGGER) encountered a database problem when it attempted to update a global variable. xxxx contains the failure codes for the four attempts as documented in section R2 of the Maintaining Database Integrity chapter in the Administration and Operations Guide. It is very likely that the database may have integrity errors or that the process-private data structures are corrupted.

Action: Report this database error to the group responsible for database integrity at your operation

## TRIGZBREAKREM ⚠

**TRIGZBREAKREM,** ZBREAK in trigger tttt removed due to trigger being reloaded

Run Time Warning: This indicates your process had a ZBREAK defined within the XECUTE code for trigger tttt, but some action replaced the definition for trigger tttt so GT.M removed the ZBREAK.

Action: If appropriate examine the trigger with ZPRINT and reestablish the ZBREAK. The message is tied to BREAKMSG mask 16 (See VIEW BREAKMSG). The default message mask is 31, which includes masks 1, 2, 4, 8, and 16. Using the VIEW command to set the BREAKMSG mask to 7 or any other pattern that excludes 16, disables this message.

## ZSOCKETATTR ⊕

**ZSOCKETATTR,** Attribute "xxxx" invalid for $ZSOCKET function msg name

Run Time Error: This indicates the named attribute is not recognized for the $ZSOCKET function.

Action: Check for spelling, review the documented list of available attributes and adjust the attribute argument.

## ZSOCKETNOTSOCK ⊕

**ZSOCKETNOTSOCK,** $ZSOCKET function called but device is not a socket

Run Time Error: The code invoked the $ZSOCKET() function with a device which is not a Socket Device.

Action: Review device usage and revise as appropriate.