V6.2-001

GT.M

Release Notes

Contact Information

GT.M Group
Fidelity National Information Services, Inc.
2 West Liberty Boulevard, Suite 300
Malvern, PA 19355
United States of America

GT.M Support for customers: gtmsupport@fisglobal.com
Automated attendant for 24 hour support: +1 (610) 578-4226
Switchboard: +1 (610) 296-8877
Website: http://fis-gtm.com

Legal Notice

Revision History		
Revision 1.0	26 December 2014	V6.2-001. First published revision.

Table of Contents

V6.2-001

Overview

V6.2-001 brings important enhancements to GT.M: a production grade implementation of auto-relink, triggers for global variables that span regions, SOCKET devices that support Transport Layer Security (TLS), and several other enhancements and fixes.

- Introduced as field-test grade functionality in V6.2-000, auto-relink is production grade in V6.2-001. While the API for programs that use auto-relink is upward compatible, the underlying implementation is more robust & scalable. Whereas in V6.2-000, processes mapped object files directly, in V6.2-001 GT.M dynamically loads object code into shared memory for execution. Instead of each process mapping each object file - potentially hundreds to thousands of object files by each of hundreds to thousands of processes - processes only need to map shared memory segments (as they do for database files) within which the GT.M run-time system dynamically manages object code. As Linux on x86 permits shared memory segments to be mapped using hugepages, V6.2-001 provides the opportunity for further scalability enhancements resulting from smaller page tables, and a greater probability of successful memory accesses directly from on-chip translation lookaside buffers. See GTM-8160 below.

- GT.M supports triggers independent of global directory and for global variables that span multiple regions. Previously trigger maintenance required careful choice of a default region and GT.M only supported global variables whose span was limited to a single region. See GTM-6901 and GTM-7877 below. In addition trigger maintenance is more more robust and user-friendly.

- SOCKET devices support encrypted communication using TLS (formerly known as SSL). Because this functionality has a wide variety of user stories (use cases) and has substantial complexity, although the code appears robust, we are not confident that we have exercised a sufficient breadth of use cases in testing. Also we may make changes in future releases that are not entirely backwards compatible. We encourage you to use with this facility in development and testing, and to provide us with feedback. If you are an FIS customer and wish to use this in production, please contact us beforehand to discuss your use case(s). As with all cryptographic functionality, GT.M includes no code that performs cryptographic calculations, uses a plugin architecture, and ships with a reference implementation of a plugin that FIS tests against selected, widely used, cryptographic libraries. See GTM-7418 below.

V6.2-001 also has numerous smaller enhancements and fixes.

The online help facilities provided through the ZHELP command in GT.M, and the HELP commands in the utilities, provide current, hierarchically organized information current. 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: $char(10) mupip backup -database ACN,HIST /backup	

	UNIX	OpenVMS
Reference Number	A reference number is used to track software $char(10) enhancements and support requests. $char(10) It is enclosed between parentheses ().	
Platform Identifier	[UNIX]	[OpenVMS]
	The platform identifier does not appear $char(10) for those new features or $char(10) 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. 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) -rootprimary (still accepted)
replicating instance (or secondary instance) and propagating instance	N/A for replicating instance or secondary instance. -propagateprimary for propagating instance	replicating instance. 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. 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 adopted 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.

Δ 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 HP-UX	11V3 (11.31)	*FIS intends to no longer support this platform after October 1, 2015. 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	*V6.2-001 is the last GT.M release for this platform. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.* 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. 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. Although this platform remains at present fully supported with respect to bug fixes, new functionality is supported on this platform only for FIS' convenience.
IBM System p AIX	6.1, 7.1	Only 64-bit versions of AIX are Supported.

Platform	Supported Versions	Notes
		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. 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)	*FIS intends to no longer support this platform after December 5, 2015. 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. GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.12 or later) and ncurses (version 5.7 or later). 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 4.2) to the subsequent prompt **Enter ICU version (ICU version 3.6 or later required. Enter as major-ver.minor-ver):** 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): • Find the directory where libncurses.so is installed on your system. • 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.
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 diminishes 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.

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).

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 V6.2-001 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.2-001_arch (for example, /usr/lib/fis-gtm/V6.2-001_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.2-001_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.

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 distribution 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 V6.2-001

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 V6.2-001 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 V6.2-001 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 V6.2-001.
- 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 V6.2-001.

- 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 database created with V6 supports up to 992Ki blocks and is not backward compatible. V6 databases that take advantage of V6 limits on key size and records size cannot be downgraded. 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 V6.2-001 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

V6.2-001 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.2-001 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 V6.2-001 and you have database triggers defined in V6.2-000 or earlier, you need to ensure that your trigger definitions are wholesome in the older version and then run MUPIP TRIGGER -

UPGRADE. If you have doubts about the wholesomeness of the trigger definitions in the old version use the instructions below to capture the definitions delete them in the old version (-*), run MUPIP TRIGGER -UPGRADE in V6.2-001 and then reload them as described below.

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 V6.2-001 or if you find your prior version trigger definitions have problems. For versions V5.4-000/V5.4-000A/V5.4-001 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/V6.2-001.

To extract and reapply the trigger definitions on V6.2-001 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 V6.2-001, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.2-001 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 V6.2-001.
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 V6.2-001 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_V6.2-001_i686, then gtmprofile and gtmcshrc set $gtm_dist to /usr/lib/fis-gtm/gtm_V6.2-001_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. In particular, ICU changed its numbering scheme, such that after version 4.8, the next version of ICU was 49. As GT.M continues to use the original numbering scheme, you must convert the new version number to the old in the environment variable gtm_chset. For example, if **icu-config --version** reports 52.1, you should set gtm_icu_version to 5.2.

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 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

V6.2-001

Fixes and enhancements specific to V6.2-001 are:

Id	Prior Id	Category	Summary
GTM-227	C9605-000064	MUPIP	In UNIX, automatically detects file format for MUPIP LOAD ✓
GTM-4414	C9C06-002043	MUMPS	Jobparameter PASSCURLVN on UNIX provides the current local variable context to the process create by the JOB command ✓
GTM-5022	C9D07-002344	MUMPS	ZWRITE and ZSHOW "V" no longer insert spaces in their output
GTM-6597	C9K04-003267	MUPIP	See GTM-227
GTM-6900	-	DB	Improved error messages from $ZTRIGGER() and MUPIP TRIGGER ✓
GTM-6901	-	DB	Eliminate trigger definition dependency on the global directory and address some edge cases ✓
GTM-7076	-	DB	Trigger deletes specified with a wild-card (*) that do not match any existing trigger are not an error
GTM-7083	-	DB	MUPIP TRIGGER -UPGRADE revises trigger formats from prior to current releases ✓
GTM-7418	-	MUMPS	SOCKET devices support TLS connections ✓
GTM-7522	-	DB	$ZTRIGGER() and MUPIP TRIGGER always use TP and defer output on stdout until transaction commit ✓
GTM-7641	-	DB	See GTM-6901
GTM-7678	-	DB	Better handling of trigger definitions containing multiple command specifications
GTM-7779	-	DB	Additional GVSTATS items and $VIEW(PROBECRIT,<region>) facility ✓
GTM-7843	-	DB	Minimal MUTEXLCKALERT messages due to an Instance Freeze
GTM-7862	-	MUPIP	Replicate solely from Journal Files ✓
GTM-7868	-	MUPIP	See GTM-227
GTM-7877	-	DB	Triggers can now apply to spanning regions ✓

Id	Prior Id	Category	Summary
GTM-7914	-	MUMPS	Better handling of environment veriable manipulation prevents very unusual hangs
GTM-7927	-	Other Utilities	The CMake driver supports builds for all the reference implmentation encryption libraries
GTM-7929	-	MUMPS	The CMake driver supports builds for all the reference implmentation encryption libraries
GTM-7947	-	DB	Fix to trigger deletion using a wild-card when triggers have long names
GTM-7964	-	MUMPS	Fix for PARSE on a PIPE device with a missing PATH
GTM-7974	-	DB	Trigger select operations with a wild-card work across multiple regions
GTM-7985	-	DB	Enforced limits on line length and total length for a trigger XECUTE action
GTM-8068	-	MUMPS	Better error for mumps -run of a routine whose object file name does not match its routine name
GTM-8081	-	MUPIP	Replication Fix for Large Compressed Messages
GTM-8093	-	Other Utilities	Appropriate message from a failure to start gtmsecshr
GTM-8094	-	MUMPS	Better handling of errors creating an additional process such as with the JOB command
GTM-8101	-	Other Utilities	Better handling of <CTRL-C> and other signals in the maskpass utility
GTM-8106	-	DB	VIEW "GVSRESET" resets more database-related tracking ✓
GTM-8132	-	MUMPS	In UNIX, REWIND support for $PRINCIPAL ✓
GTM-8134	-	MUMPS	ZSHOW "D" shows both input and output for $PRINPCIPAL with separate devices ✓
GTM-8146	-	MUMPS	$ZSEARCH() checks its stream argument to be within 0-255
GTM-8153	-	Other Utilities	More robust error handling and compilation for the Makefile of the reference implementation plugin. ✓
GTM-8160	-	MUMPS	Performance and scalability modifications to the auto-relink mechanism and add additional controls ✓
GTM-8168	-	MUMPS	Improved accuracy on timed actions
GTM-8172	-	MUMPS	Better cleanup after exit from recursively linked routines
GTM-8174	-	MUMPS	Raising zero to a negative power (0**-x) produces a DIVZERO error

Change History

Id	Prior Id	Category	Summary
GTM-8175	-	MUPIP	MUPIP LOAD correctly reports key counts above 2**32-1
GTM-8182	-	MUPIP	See GTM-227
GTM-8185	-	MUPIP	Correctly issue a REC2BIG error for replicated transactions
GTM-8191	-	DB	Available as field test grade functionality, VIEW "POOLLIMIT":<region>:expr restricts a process to a subset of a region's global buffer pool, a technique used by MUPIP REORG ✓
GTM-8194	-	MUMPS	Source Hash Change to improve performance ✓
GTM-8195	-	MUPIP	Better handling of errors on database file open
GTM-8196	-	MUMPS	Fixes for WRITE /WAIT for SOCKET devices
GTM-8204	-	MUMPS	$ZPEEK() mnemonics to access the jnl_private_control & jnl_buffer structures and to format time fields in $HOROLOG format ✓
GTM-8205	-	MUPIP	Fix cross-endian replication broken in V6.2-000
GTM-8207	-	DB	$ZTWORMHOLE always replicates properly when originating and replicating instances have differing null collation
GTM-8209	-	MUPIP	Fix off by a day error in MUPIP JOURNAL -EXTRACT -DETAIL reporting UNIX Epoch time (0) in $HOROLOG format
GTM-8210	-	MUPIP	Prevent looping in replication Update Process dealing with triggers and contending updates

M-Database Access

- MUPIP TRIGGER limits trigger expression source lines to 80 characters including a trailing ellipsis to indicate there was more text, and they also replace any non-graphic characters with a dot (.) Previously, such messages had non-graphic characters represented by OS or terminal [emulator] conventions and no embedded indication of the truncation. [UNIX] (GTM-6900) ✓

- GT.M maps trigger definitions to the region to which they apply. Previously, it mapped them partially to the default region of the global directory in use at the time of their definition, which caused documented limitations and pitfalls when using multiple global directories or spanning regions. A $ZTRIGGER() or MUPIP TRIGGER action (delete or select) applies to all triggers in all regions matching the specified signature. If the argument specifies an incomplete trigger signature, for example, only the name, the specification may match multiple triggers and apply the delete or select to all of them. FIS recommends you run a select and analyze the scope of the signature before any signature limited delete. GT.M allows defining triggers with the same name and signature in different regions, but does not allow defining triggers with the same name in different regions within the same global directory if they have different signatures; previously it allowed neither. When loading a trigger definition, if a user-defined name conflicts with a name in any region to which the trigger applies, $ZTRIGGER() and MUPIP TRIGGER report an error. However, when the name is auto-generated, they generate a name in every region, so if there are multiple (spanning) regions, the same trigger might have a differing auto-generated name in each region.

  Trigger definitions are subject to the same maximum record length checks as documented for other GT.M components; when GTM-6341 changed the definition changed in V6.0-000, trigger definitions retained the prior convention. A trigger deletion attempt on a read-only database fails with a TRIGMODREGNOTRW error, but only if the attempt specifies existing triggers; previously it failed even if there were no matching triggers to delete.

  GT.M manages concurrent trigger definition updates appropriately. Previously, in the unlikely circumstance of conflicting trigger definition updates, a process could get a segmentation violation (SIG-11). ZPRINT of a trigger concurrently being deleted produces the trigger source; previously it caused a GTMASSERT.

  $ZTRIGGER() and MUPIP TRIGGER skip regions with a GT.CM (client) access method; previously, using these facilities with a global directory caused a segmentation violation (SIG-11) on the client if the corresponding server was active..

  ZPRINT/$TEXT()/ZBREAK recognize both a runtime-disambiguator, delimited with a hash-sign (#), and a region-disambiguator, delimited by a slash(/); previously they supported only the runtime-disambiguator. ZPRINT and ZBREAK treat a trigger-not-found case as a TRIGNAMENF error, while $TEXT() returns the empty string. When their argument contains a region-disambiguator, these features ignore a null runtime-disambiguator. When their argument does not contain a region-disambiguator, these features act as if runtime-disambiguator is specified, even if it has an empty value. When an argument specifies both runtime-disambiguator and region-disambiguator and the runtime-disambiguator identifies a trigger loaded from a region different from the specified region, or the region-disambiguator identifies a region which holds a trigger that is not mapped by $ZGBLDIR, these features treat the trigger as not found. Here are some examples of disambiguator combinations:

  ```
  ZPRINT ^x#/BREG      : Print           trigger routine user-named "x" in region BREG
  ZPRINT ^x#1#/BREG    : Print           trigger routine auto-named "x#1" in region BREG
  ZPRINT ^x#1#A/BREG   : Print           trigger routine auto-named "x#1", runtime disambiguated by "#A", AND in
   region BREG
  ZPRINT +1^x#1#A/BREG : Print line 1 of trigger routine auto-named "x#1", runtime disambiguated by "#A", AND in
   region BREG
  ```

DSE accepts ^#t as a valid global name when specifying a key; previously it treated ^#t as invalid. Also, DSE uses the same maximum record length checks as documented for other GT.M components; when GTM-6341 changed the definition changed in V6.0-000, DSE retained the prior convention. [UNIX](GTM-7641)(GTM-6901) ✓

- $ZTRIGGER() and MUPIP TRIGGER do not require a delete specified with a wild-card (*) to find any matching triggers, but rather report this in the action summary. Previously they treated this as an error. [UNIX](GTM-7076)
- MUPIP TRIGGER -UPGRADE upgrades older trigger definitions into current format. If GT.M encounters an old trigger definition it produces a NEEDTRIGUPGRD message. To preserve the possibility of a straightforward downgrade to an earlier version, perform a select "*" action with MUPIP TRIGGER (or $ZTRIGGER() and save the result. Note that TRIGGER -UPGRADE assumes that the existing trigger definitions are properly defined; if the prior release has produced defective triggers delete them with a wild-card ("*") and redefine them in the new release. In the event of a downgrade, delete "*" all triggers before the downgrade and insert the saved version from before the upgrade. Attempting to perform a MUPIP TRIGGER -UPGRADE on a database without write authorization to the database produces a TRIGMODREGNOTRW error. [UNIX] (GTM-7083) ✓
- $ZTRIGGER()and MUPIP TRIGGER always operate within a TP transaction even if they need to implicitly create one to encapsulate their work. These trigger maintenance operations reserve their output until the transaction commits (TCOMMIT where $TLEVEL goes to zero) at which time they deliver the entire output consistent, including that for all $ZTRIGGER() invocations in the successful processing of a larger transaction, to the $IO current at that ultimate TCOMMIT. If an explicit transaction ends with a TROLLBACK, it does not produce any $ZTRIGGER() output. Previously, $ZTRIGGER() produced potentially inappropriate error reports indicating name collisions where none existed, and also incorrect, differing, and in cases of RESTARTs, multiple, outputs. Trigger definition reporting operations, $ZTRIGGER("SELECT",) and MUPIP TRIGGER -SELECT, return a non-zero exit status when their selection criteria encounter an error in the middle of the select. Previously, under a variety of conditions, these operations returned a success indicator even though the operation actually failed. MUPIP TRIGGER -SELECT now displays all output on stdout. Previously it displayed valid output on stdout and error output on stderr. [UNIX](GTM-7522) ✓
- See GTM-6901. (GTM-7641)
- In cases where a KILL trigger shared some common characteristics (e.g. global name, XECUTE string etc.) with one or more SET triggers, $ZTRIGGER() and MUPIP TRIGGER appropriately establish the KILL trigger. Previously, they could set up an inappropriate KILL trigger, corrupt the trigger database or issue an incorrect TRIGDEFBAD error. In addition, if a specified trigger signature otherwise exactly matches an existing trigger, any -options specified in the specification replace any -options in the existing trigger. However, if the command portion of the input definition only partially matches an existing trigger, $ZTRIGGER() and MUPIP TRIGGER issues an error. Previously the trigger update facilities added the specified -options to any existing options. Similar to the trigger update -options handling, the trigger update facilities allow changing the name of an existing trigger only if the every other element of the definition signature matches an existing trigger. Previously the specified -name replaced the existing trigger name even when the command portion of the input definition only partially matched an existing trigger. Also, the error handling, output and error messages for trigger loads, selects and deletes changed to improve clarity, completeness and consistency. Finally, the trigger update facilities ignore -options specified for trigger deletes; previously if a delete included a -options specification, the delete inappropriately removed the options from definitions of similar remaining triggers with differing command characteristics. [UNIX](GTM-7678)
- Access statistics for global variables include the following items:

```
CAT - total of critical section acquisitions successes
CFE - total attempts in CFT caused by epochs
CFS - sum of squares of blocked critical section acquisitions
CFT - total of blocked critical section acquisitions
CQS - sum of squares of critical section acquisition queued sleeps
```

```
CQT - total of critical section acquisition queued sleeps
CYS - sum of squares of critical section acquisition processor yields
CYT - total of critical section acquisition processor yields
```

Note that statistics reported by $VIEW() or DSE output are those for the database region, while those reported by ZSHOW "G" output are process-specific.

$VIEW("PROBECRIT",<region>) acquires and releases a critical section for the region (the "probe"), returning a string with the following fields:

```

CPT - nanoseconds for the probe to get the critical section
CFN - number of failures of the probe to get the critical section
CQN - number of queue sleeps by the probe
CYN - number of process yields by the probe
CQF - number of queue fulls encountered by the probe
CQE - number of empty queue slots found by the probe
CAT - total of critical section acquisitions successes
```

[UNIX](GTM-7779) ✓

- When Instance Freeze is in effect, GT.M seldom produces MUTEXLCKALERT messages in the operator log; previously it frequently produced such messages. [UNIX](GTM-7843)

- GT.M triggers now apply to spanning regions (which means the TRIGNOSPANGBL error is obsolete). When $ZTRIGGER() or MUPIP TRIGGER define triggers that apply to globals spanning multiple regions, each of the spanned regions install a definition. In the example below, ^example is a non-spanning global and ^example2 spans AREG, BREG and DEFAULT:

```

File tmp4.trg, Line 1: All existing triggers (count = 2) deleted
File tmp4.trg, Line 5: Added Non-SET trigger on ^example named example#1
File tmp4.trg, Line 6: Added Non-SET trigger on ^example2 (region AREG) named example2#1
File tmp4.trg, Line 6: Added Non-SET trigger on ^example2 (region BREG) named example2#1
File tmp4.trg, Line 6: Added Non-SET trigger on ^example2 (region DEFAULT) named example2#1
File tmp4.trg, Line 11: Modified SET and/or Non-SET trigger on ^example named example#1
```

note unlike the output for the spanning region global ^example2, the output for the single-region global ^example does not include the region. The "select" and "delete-by-trigger-name" operations report the region for the trigger, regardless of whether the global name corresponding to this trigger spans regions or not. [UNIX](GTM-7877) ✓

- Deletion of a trigger with wildcards using $ZTRIGGER() or MUPIP TRIGGER works correctly. Previously if more than one trigger matched the specified wild-card and corresponded to the same global name which was also greater than 21 characters, it caused a segmentation violation (SIG-11). [UNIX](GTM-7947)

- Specifying a trigger name with a trailing wild-card in $ZTRIGGER() and MUPIP TRIGGER selects all triggers that start with the prefix before the asterisk (*); previously it only selected triggers matching the prefix that were mapped to the same region as the first name that matches. Specifying a trigger name more than once in a list of names selects the trigger just once; previously this caused a segmentation violation. [UNIX](GTM-7974)

- $ZTRIGGER() and MUPIP TRIGGER update operations enforce a limit of 8KiB bytes per line and a total of 1MiB bytes for a multi-line trigger XECUTE string. Previously, exceeding the limits could cause a segmentation violation (SIG-11) [UNIX](GTM-7985)

- VIEW "GVSRESET"[:"<region>"] resets the process-specific fields that are part of the ZSHOW "G" result and database file header fields holding records reported by: GVSTAT, BG trace, buffer pool accounting and the TP block modification details. Note a VIEW "GVSRESET" performed by a process with read-only database access changes only the process-specific information and has no effect on the database file header. DSE CHANGE -FILEHEADER -GVSTATSRESET clears the same database file header fields as VIEW "GVRESET"; previously it only cleared the records for GVSTAT. In addition, regions for VIEW and $VIEW() arguments with region elements treat region names as case insensitive, and, in VIEW command cases where a missing region element means all regions, accept a region name of "*" as meaning all regions. Previously region name elements in VIEW command arguments did accept "*" and required upper-case, as did $VIEW() region arguments. (GTM-8106) ✓

- Available as field test grade functionality in a production release, VIEW "POOLLIMIT":<region>:expr, where expr is of the form n[%] provides a mechanism for a process that has the potential to "churn" global buffers to limit the potential impact on other processes by restricting the number of global buffers it uses. If the expression ends with a per-cent sign (%), the number is taken as an as a percentage of the configured global buffers and otherwise as an ordinal number of preferred buffers; standard M parsing and integer conversions apply. Preferred buffer values are limited to between 32 and one less than half the buffer pool inclusive; with the exception of zero (0) or 100 per cent, which turn off the limitation; specifications exceeding those limits provide the value of the nearer limit. If the argument specifies "*" for the region, the command applies to all regions. $VIEW("POOLLIMIT",<region>) returns the current value for the region as an ordinal number - zero (0) when there is no limit in place. Note that this facility is designed for use by a relatively small subset of processes. In addition, MUPIP REORG uses this facility to limit its buffers to a value established by the UNIX environment variable (or OpenVMS logical name) gtm_poollimit using the syntax described for VIEW "POOLLIMIT" with a default of 64 if gtm_poollimit is not specified. Note that this may slightly slow a standalone REORG but can be overridden by defining gtm_poollimit as 0 or "100%". Previously REORG used all of the available buffer pool and thus could have a larger impact on concurrent processing. (GTM-8191) ✓

- GT.M replicates $ZTWOrmhole correctly when the originating and replicating instance do not use the same null collation; previously, in an especially rare case, which required using $ZTWOrmhole, it could be modified during replication. [UNIX] (GTM-8207)

M-Other Than Database Access

- When a JOB command specifies the PASSCURLVN jobparameter, the new process inherits the current collation, all locals, aliases, and alias containers from the JOB'ng process' current stack level. As a result of this, a ZWRITE issued in the JOB'd process has the same output, except for any out of scope aliases, as a ZWRITE in the context of the JOB command. If the JOB command finds a ZWRITE representation of any lvn, consisting of a its full name, its subscripts, corresponding value, quotes and the "=" sign, exceeding 1MiB, it produces an JOBLVN2LONG error both in the JOB'ng process and in the JOB'd processes error output stream. If a JOB command does not specify PASSCURLVN, the JOB'd process(es) inherits no local variables from the parent. While not an inexpensive command, you can use the "exclusive" NEW command to control the context passed to the JOB'd process; for example, adding "NEW (LOCALA, LOCALB)" before the JOB command would pass only LOCALA and LOCALB. [UNIX](GTM-4414) ✓

- ZWRITE and ZSHOW "V" do not add spaces within a given local variable's corresponding value. Previously, both of the commands could insert spaces at the beginning of every line after the first. (GTM-5022)

- SOCKET devices support encrypt connections with TLS using an encryption plugin. GT.M ships with a reference implementation of the plugin which uses OpenSSL and also supports TLS of replication stream. OpenSSL options are controlled by a configuration file. The WRITE /TLS command activates this feature for connected sockets. Because this functionality has a wide variety of user stories (use cases) and has substantial complexity, although the code appears robust, we are not confident that we have exercised a sufficient breadth of use cases in testing. Also we may make changes in future releases that are not entirely backwards compatible. We encourage you to use with this facility in development and testing, and to provide us with feedback. If you are an FIS customer and wish to use this in production, please contact us beforehand to discuss your use case(s).

  The WRITE /TLS command has the form:

  ```
  WRITE /TLS(option[,[timeout][,tlsid]])
  ```

  where option is "server" or "client" indicating which TLS role to assume or "renegotiate" which is used by a server to request a new handshake to reevaluate the client credentials. The server role requires a certificate specified in the configuration file section with the label matching tlsid. The client role may require a certificate depending on the OpenSSL options. If the argument specifies a timeout, GT.M sets $TEST to 1 if the command successfully completed or to 0 if it timed out. $DEVICE provides status information in case of an error. ZSHOW "D" includes "TLS" in the second line of the output for an encrypted socket.

  Note that SOCKET device actions may produce the following previously documented errors (all with a TLS prefix and previously only issued by MUPIP for a TLS replication stream): TLSDLLOPEN, TLSINIT, TLSCONVSOCK, TLSHANDSHAKE, TLSCONNINFO, TLSIOERROR, and TLSRENEGOTIATE.

  The TLS plugin uses OpenSSL options in the configuration file specified under the tls: label as the default for all TLS connections and under the specific labels to override the defaults for corresponding connections. Previously, GT.M recognized a more limited set of OpenSSL options under the tls: label and only certificate related items under the lower level labeled sections. In addition to the previous items allowed in the tls: section of the configuration file for TLS replication, GT.M recognizes the following for both sockets and TLS replication:

  - The cipher-list option specifies which cryptographic algorithms to use. The format of this option is described by the OpenSSL ciphers man page. An empty string uses a default value of "ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH" for replication connections and the OpenSSL default cipher list for socket connections.

- The ssl_options, documented in the man page for SSL_set_options, modify the default behavior of OpenSSL. When specifying multiple options, separate them with acolons (:) delimiter. The ssl-options specified in a labeled section add to, or override, those specified at the "tls" level. An exclamation mark ("!") preceding an option in a labeled section disables any default for that option specified at the tls: level; for example:

```
        tls: {
                ssl-options: "SSL_OP_CIPHER_SERVER_PREFERENCE";
                mylabel: {
                        ssl-options: "!SSL_OP_CIPHER_SERVER_PREFERENCE";
                };
             }
```

- The verify-mode option specifies how OpenSSL verifies certificates. If no verify-mode is specified, a socket connection defaults to SSL_VERIFY_NONE. We have tested with SSL_VERIFY_PEER and SSL_VERIFY_NONE See the man page for SSL_set_verify for details. SSL_VERIFY_PEER has two additional flags which modify verification only for the server role; when adding them to the option string, use the colon (:) delimiter.

- A verify-depth option specified in a labeled section applies to connections associated with that section.

When placing the private key for a certificate at the beginning of the certificate file, you may omit the "key" item from the configuration file. The format of the combined file is:

```
             -----BEGIN RSA PRIVATE KEY-----
             [encoded key]
             -----END RSA PRIVATE KEY-----
             [empty line]
             -----BEGIN CERTIFICATE-----
             [encoded certificate]
             -----END CERTIFICATE-----
             [empty line]
```

GT.M buffers WRITEs to TLS enabled sockets until a subsequent USE :FLUSH, WRITE !, WRITE #, or an internal 400 millisecond timer expires.

Owing to the range of OpenSSL versions in use across the breadth of platforms and versions supported by GT.M, on all platforms, but especially on non-Linux UNIX platforms, FIS recommends rebuilding the plugin from sources included with the GT.M binary distribution with the specific version of OpenSSL installed on your systems for any production or production staging environments that use TLS. Please also review & edit the following common header and library paths in the Makefile to reflect those on your systems when you rebuild the plugin.

```
# Common header and library paths
IFLAGS += -I /usr/local/ssl/include -I /usr/local/include -I /usr/include -I $(gtm_dist) -I $(CURDIR)
ifeq ($(BIT64),0)
 LIBFLAGS += -L /usr/local/ssl/lib -L /usr/lib/x86_64-linux-gnu -L /usr/local/lib64
 LIBFLAGS += -L /usr/local/lib -L /usr/lib64 -L /usr/lib -L /lib64 -L /lib -L %60pwd%60
else
 LIBFLAGS += -L /usr/local/ssl/lib -L /usr/lib/x86-linux-gnu -L /usr/local/lib32
 LIBFLAGS += -L /usr/local/lib -L /usr/lib32 -L /usr/lib -L /lib32 -L /lib -L %60pwd%60
endif
```

[UNIX](GTM-7418) ✓

- On the rare occasions GT.M or its utilities add or change an environment variable, they protect the operation against interruption by a signal (such as SIGTERM). Previously, there was a very small chance a signal could interrupt a GT.M process in the middle of adding or changing an environment variable, causing the process to hang indefinitely. This issue was discovered in the GT.M development environment and was never reported by any user. [UNIX] (GTM-7914)
- CMakeLists.txt now builds all three of the reference implementation encryption libraries. Previously it built only the reference implementation that worked with libgcrypt. [UNIX](GTM-7929)
- If an OPEN command argument for a PIPE device specifies the PARSE deviceparameter and a COMMAND that GT.M cannot locate and the PATH environment variable has not been defined for the process, the DEVOPENFAIL message contains these circumstances. Previously, this combination of conditions caused a segmentation violation (SIG-11). [UNIX](GTM-7964)
- GT.M reports ZLINKFILE/ZLMODULE errors when a routine name in the corresponding object file does not match the routine name specified with $gtm_dist/mumps -run <routine-name>. The mismatch can arise from moving an object file to a different name or compiling with the -NAMEOFRTN option on the mumps command. This matches the report produced by direct mode in response to the same circumstance. Previously this raised a GTMASSERT2 referencing job_addr.c. [UNIX] (GTM-8068)
- A GT.M process (including utilities) that cannot start a child process handles and reports the error appropriately; previously such an error could cause a failure in the parent process and/or produce misleading error messages. This issue was discovered in the GT.M development environment and was never reported by any user. [UNIX](GTM-8094)
- 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) ✓
- When $PRINCIPAL input and output are different devices, ZSHOW "D" shows them as separate items identified as 0 for input and 0-out for output. Note: algorithms processing ZSHOW "D" output may need adjustment. If they process output saved to a variable the additional output line[s] must be accounted for by $ORDER() or an integer iteration. Previously, ZSHOW "D" only provided the input side of the device. [UNIX](GTM-8134) ✓
- $ZSEARCH() range (0-255) checks its stream argument; previously it silently used a stream matching stream#256. This was fixed in V6.2-000, but not documented in the original release notes. (GTM-8146)
- GT.M loads an object file linked from an object directory (in $ZROUTINES) with a *-suffix (i.e. auto-relink-enabled) into a shared memory segment (referred to henceforth as a Rtnobj shared memory segment). In V6.2-000 this was in mmap() memory which could increase the number of file descriptors required.

  For each auto-relink enabled directory which a GT.M process accesses while searching through $ZROUTINES, GT.M creates a small control file (Relinkctl) in the directory identified 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-<murmur> where <murmur> is a hash of the realpath() to an auto-relink directory; for example: /tmp/gtm-relinkctl-f0938d18ab001a7ef09c2bfba946f002). With each Relinkctl file, GT.M creates and associates a block of shared memory that contains associated control structures. Among the structures is a cycle number corresponding to each routine found in the routine directory; a change in the cycle number informs a process that it may need to determine whether there is a new version of a routine. Although GT.M only creates relinkctl records for routines that actually exist on disk, it may increment cycle numbers for existing relinkctl records even if they no longer exist on disk.

  GT.M creates both the Relinkctl file and shared memory with permissions based on the logic described in the "IPC Permissions" column of the "Shared Resource Authorization Permissions" section in the Administration and Operations Guide, except that the object directory, rather than the database file, provides the base permissions.

An auto-relink-enabled directory uses one or more Rtnobj shared memory segments to store the object (.o) files, creating the first one when linking the first object file from an auto-relink-enabled directory. By default, GT.M creates an initial Rtnobj shared memory segment of 1 MiB. You can control this size by defining an environment variable gtm_autorelink_shm to an integer value in MiB. If the value of gtm_autorelink_shm is not a power of two, GT.M rounds the value to the next higher integer power of two. If the first object (.o) file does not fit in a new Rtnobj segment, GT.M rounds the allocation up to the smallest integer power of two required to make it fit. When GT.M needs room for object files, and existing Rtnobj segments have insufficient free space, it creates an additional shared memory segment, double the size of the last. Note that when hugepages are enabled, the actual Rtnobj shared memory size might be more than requested implicitly or explicitly through $gtm_autorelink_shm.

GT.M creates Rtnobj shared memory with the same permissions as Relinkctl shared memory, as described above, but with execute permissions enabled to correspond with read permissions.

The MUPIP RCTLDUMP command reports information related to relinkctl files and their associated shared memory segments. The syntax is MUPIP RCTLDUMP [dir1]. If the optional parameter dir1 is not specified, MUPIP RCTLDUMP dumps information on all its active auto-relink-enabled directories (those with with a *-suffix) identified by $gtmroutines. With a directory path specified for dir1, MUPIP RCTLDUMP reports information on the one directory. An example output follows. It lists the full path of the Object directory; its corresponding relinkctl file name; the number of routines currently loaded in this relinkctl file; the number of processes including the reporting MUPIP process that have this Relinkctl file open; the shared memory id and length of the Relinkctl shared memory segment; one or more Rtnobj shared memory segment(s); and a listing of all the routine names loaded in this file (lines starting with rec#...).

```
> mupip rctldump .
Object Directory         : /obj
Relinkctl filename       : /tmp/gtm-relinkctl-f0938d18ab001a7ef09c2bfba946f002
# of routines            : 1
# of attached processes  : 2
Relinkctl shared memory  : shmid: 11534344  shmlen: 0x57c6000
Rtnobj shared memory # 1 : shmid: 11567113  shmlen: 0x100000  shmused: 0x200  shmfree: 0xffe00  objlen: 0x1c0
    rec#1: rtnname: abcd  cycle: 1  objhash: 0xedbfac8c7f7ca357  numvers: 1  objlen: 0x1c0  shmlen: 0x200
```

Two lines need more explanation.

- The Rtnobj shared memory line : All the length fields are displayed in hexadecimal. shmlen is the length of the allocated shared memory segment in bytes. shmused is the length that is currently used. shmfree is the length available for use. objlen is the total length of all the objects currently loaded in this shared memory. As GT.M allocates blocks of memory with sizes rounded-up to an integer power of two bytes, shmused is always greater than objlen; for example with an objlen of 0x1c0, the shmused is 0x200.

- Lines of the form rec#... indicate the record # in the relinkctl file. Each relinkctl file can store a maximum of 1,000,000 records, i.e., the maximum number of routines in a directory with auto-relink enabled is one million. Each record stores a routine name (rtnname:), the current cycle # for this object file record entry (cycle:) which gets bumped on every ZLINK or ZRUPDATE command, the hash of the object file last loaded for this routine name (objhash:), the # of different versions of object files loaded in the Rtnobj shared memory segments with this routine name (numvers:), the total byte-length of the one or more versions of object files currently loaded with this routine name (objlen:), the total length used up in shared memory for these object files where GT.M allocates each object file a rounded-up perfect 2-power block of memory (shmlen:). In V6.2-000, reports and error messages gave a file name rather then a routine name.

Given a relinkctl file name, one can find the corresponding directory path using the Unix "strings" command on the Relinkctl file. For example, "strings /tmp/gtm-relinkctl-f0938d18ab001a7ef09c2bfba946f002", corresponding to the above MUPIP RCTLDUMP output example, would output "/obj" the corresponding directory name.

The environment variable gtm_autorelink_keeprtn if set to 1, t[rue], or y[es] causes exiting processes to leave auto-relinked object code in the shared memory repositories, while if undefined, 0, f[alse] or n[o] causes exiting processes to purge any routines currently use by no processes. All values are case-independent. When gtm_autorelink_keeprtn is defined and TRUE:

- Process exit is simplified, with the performance gain - faster process termination - likely to be observable only when a large number of processes exit concurrently.

- Where routines are likely to be repeatedly used by other processes, such as in a production environment, leaving a routine in shared memory even when no longer used by existing processes, will result in slightly faster linking of that routine by future processes, although the effect may not be observable except when an application frequently uses short-lived processes, such as GT.M routines invoked by web servers using a CGI interface.

ZSHOW "A" (A code to mean Autorelink) provides output in the same format as MUPIP RCTLDUMP, but restricted to the routines linked by the process issuing the command. ZSHOW "*" does not include ZSHOW "A" because of an expectation that the typical volume of the information does not provide a good return for its value. If you wish your error handling or INTRPT routines to dump this information, ask for it explicitly, possibly by doing a ZSHOW "A" into a local variable before doing a ZSHOW "*".

GT.M does not support VIEW "RCTLDUMP" as it has been supplanted by ZSHOW "A" and MUPIP RCTLDUMP.

The MUPIP RUNDOWN -RELINKCTL command cleans up orphaned Relinkctl files. FIS strongly recommends avoiding actions that tend to make such cleanup necessary - for example, kill -9 of GT.M processes or ipcrm -m of active Relinkctl and/or Rtnobj shared memory segments. The syntax is MUPIP RUNDOWN -RELINKCTL [dir1]. If the optional parameter dir1 is not specified, MUPIP RUNDOWN -RELINKCTL examines the env var $gtmroutines, attempts to verify and correct their attach counts and runs down all its inactive auto-relink-enabled directories (those with with a *-suffix). Alternatively, one can specify a directory path for the parameter dir1 and MUPIP RUNDOWN -RELINKCTL treats it as an auto-relink-enabled directory and runs down the resources associated with this one directory. It prints a RLNKCTLRNDWNSUC message on a successful rundown and a RLNKCTLRNDWNFL message on a failure (usually because live processes are still accessing the Relinkctl file).

GTM issues a REQRLNKCTLRNDWN message if it finds evidence of need for a rundown of the Relinkctl file (e.g. processes killed using kill -9 or relinkctl/rtnobj shared memory segments removed using ipcrm -m).

FIS recommends that a directory in the $zroutines of a process be either auto-relink-enabled or auto-relink-disabled for the life of the process. Changing the auto-relink mode of the directory within a process is likely to result in counter-intuitive results.

As arguments, ZRUPDATE takes object file names, including wild-cards of the form accepted by $ZSEARCH(). If ZRUPDATE fails to find at least one file to match an argument with a wild card, it issues an INFO message (seen only if $PRINCIPAL has CENABLE). When an explicit name without a wild card is specified, but there is no file in the directory or a corresponding entry in the Relinkctl, ZRUPDATE produces an error. ZRUPDATE issues most errors as FILEPARSE errors with a secondary error describing the actual issue although some errors, depending on the reason and path by which ZRUPDATE detects them, can be rather cryptic. In V6.2-000, some wild card handling was incorrect.

An explicit ZLINK or an auto-relink check the hash of an object and its replacement. If they are identical, GT.M takes no action to replace the current object, saving both memory and time.

An explicit ZLINK from an auto-relink directory acts as an implicit ZRUPDATE.

Any ZBREAK in a routine disables that routine from auto-relinking by a process until all ZBREAKs are removed.
The $gtmroutines value set by the gtmprofile file enables auto-relink by default for object files in the $gtmdir/$gtmver/o directory in M mode, and $gtmdir/$gtmver/o/utf8 in UTF-8 mode.[UNIX](GTM-8160) ✓

- GT.M schedules and delivers user-initiated timed events, such as HANGs and gtm_start_timer() call-in invocations, with the highest accuracy available to GT.M via standard operating system calls. Note that GT.M ensures that the HANG or other operation waits for at least as long as the specified time; the actual time depends on when the process gets resources to allow it to resume execution depends on the operating system. Previously, GT.M scheduled such events with both an extended duration and highest accuracy of 10ms. [UNIX] (GTM-8168)

- GT.M appropriately manages old copies of recursively linked routines when they retire from use. Previously, GT.M did not release the parts of a recursively linked routine when the last instance of the old (replaced) routine left the M virtual-machine stack due to a QUIT or ZGOTO. ZBREAKs in the replaced routine increased the size of the memory leak. [UNIX] (GTM-8172)

- Raising zero to a negative power (0**-x) produces a DIVZERO error; previously it returned 0, which was inconsistent with the result of dividing by zero: i.e. 1/(0**x). (GTM-8174)

- GT.M uses a 128 bit hash based on the MurmurHash3 algorithm, for source file text identification as visible in the output of ZSHOW "R" and $VIEW("RTNCHECKSUM",rtn). MUPIP HASH <file-names> provides the hash of source files from the command line. Previously GT.M used an MD5 hash. [UNIX](GTM-8194) ✓

- WRITE /WAIT with no timeout appropriately resumes waiting for a socket event after processing an INTRPT; previously it could return prematurely with no socket selected. In UNIX, WRITE /WAIT on a socket which has an error condition appropriately defers reporting the error until the next command that accesses the socket; starting with V6.1-000 WRITE / WAIT on a socket with an error condition could result in a GTMASSERT. (GTM-8196)

- $ZPEEK() accepts the following keyword first arguments:

```

JNL[REG]:region - to obtain fields in the jnl_private_control structure.
JBF[REG]:region - can obtain fields in shared jnl_buffer structure
```

The associated field mnemonics and characteristics are defined by the running the GTMDefinedTypesInit.m utility, which produces a cross-index in the form:

```
gtmtypfldindx(<structure-name>.<field-mnemonic>)=<n>
```

where gtmtypes(<structure-name>,<n>,*) nodes contain the field characteristics.$ZPEEK recognizes the "T" format code as selecting a $HOROLOG format for a field of 4 or 8 bytes which is intended for use on fields in UNIX time format (seconds since 01/01/1970).[UNIX](GTM-8204) ✓

Utilities-MUPIP

- MUPIP LOAD detects file format (BINARY/ZWR/GO) based on the file header of extract files from MUPIP EXTRACT, ^%GO and DSE. If it cannot recognize the header, or if the detected format does not match that selected by the user supplied -FORMAT=(BINARY/ZWR/GO) qualifier, LOAD produces a LDBINFMT error. For example, MUPIP LOAD -STDIN of a BNIARY file using shell redirection for STDIN works with or without the -FORMAT=BINARY. Previously LOAD required the -FORMAT qualifier to recognize the BIN format, ignored the qualifier for GO and ZWR, and, when it did not recognize the header, it tried to skip the first two records and treat the file as GO format and could cause a segmentation violation (SIG-11) if the selection was -FORMAT=BIN, but the file was in GO format. [UNIX](GTM-6597)(GTM-7868)(GTM-8182)(GTM-227) ✓

- See GTM-227. (GTM-6597)

- MUPIP REPLICATE -SOURCE -START accepts the -[NO]JNLF[ILEONLY] qualifier. Specifying -JNLFILEONLY forces the source server to read transactions from journal files instead of journal pool shared memory. When combined with the SYNC_IO journal option, this feature delays replication of transactions until their journal records are hardened to disk. This may be useful when replicating to a supplementary instance, as a crash and rollback on the primary could otherwise necessitate a rollback of local updates on the receiving instance. The default is -NOJNLFILEONLY. [UNIX] (GTM-7862) ✓

- See GTM-227. (GTM-7868)

- The receiver process correctly handles large compressed messages (> 16 MiB before compression) which are partially read. Such messages can occur when there is a large backlog of journal records for this receiver on the source, though the particular circumstances required to encounter this issue are rare. Previously the messages could be handled incorrectly, resulting in a corrupt message, causing a zlib message to be written to the receiver log and compression to be disabled; otherwise, replication was unaffected. This issue was introduced with GTM-7743 in V6.1-000. [UNIX] (GTM-8081)

- MUPIP LOAD reports an accurate key count; in prior releases, key counts above 2**32-1 wrapped to inappropriately smaller values. [UNIX](GTM-8175)

- See GTM-227. (GTM-8182)

- The Update Process correctly issues a REC2BIG error message when handling updates that exceed the configured maximum. Previously, the Update Process terminated with a segmentation violation (UNIX SIG-11, or OpenVMS ACCVIO) while issuing the REC2BIG error for a TP fenced update. This fixes a regression introduced in V6.0-003. (GTM-8185)

- GT.M handles failures it encounters while opening database files correctly, issuing the appropriate error messages. Previously, on rare occasions, when it attempted to report such errors, it could produce a bus error (SIG-7) or segmentation fault (SIG-11). [UNIX] (GTM-8195)

- The replication receiver server works correctly when connecting to a cross-endian source server running an older version of GT.M. In V62000 if the receiver connected to a source server running any version of GT.M from V55000 through V61000, it inappropriately exited with an INSNOTJOINED error . [UNIX](GTM-8205)

- The MUINFOUINT6 message reports used in MUPIP JOURNAL -EXTRACT -DETAIL show $HOROLOG representations of times close, or prior, to UNIX Epoch (0 seconds on 1-Jan-1970) accurately; previously such times (typically 0), appeared to be a day too close to the current day. [UNIX] (GTM-8209)

- The Update Process appropriately handles restarts that occur within trigger logic due to concurrent database activity (from MUPIP REORG or local updates on a Supplementary Instance). Previously in rare instances, the Update Process might get into an indefinite repetition of implicit TPRESTARTs recognizable by repeating TPRETRY messages in its log file. [UNIX] (GTM-8210)

Utilities-Other Than MUPIP

- CMakeLists.txt now builds all three of the reference implementation encryption libraries (libgtcrypt: AES256, OpenSSL: AES256 and BLOWFISH). Previously it built only one reference implementation that used libgcrypt with AES256. [UNIX] (GTM-7927)

- A GT.M process (including utilities) that cannot start gtmsecshr puts a UNABLETOEXECGTMSECSHR message in the operator log; previously it used an inappropriate INVTRANSGTMSECSHR message. [UNIX](GTM-8093)

- The maskpass utility restores terminal settings when terminated with a Ctrl-C or a TERM signal. Additionally, all other signals except SEGV, ABRT, BUS, FPE, TRAP, KILL, and STOP are ignored. Previously, it could leave the terminal in a state where an "stty sane" command would be required to restore terminal settings to a usable state. [UNIX] (GTM-8101)

- Makefile shipped with the reference implementation of the GT.M encryption plug-in consolidates a number of related simplistic targets into more functional ones (such as 'maskpass', 'maskpass.o', and 'gtmcrypt_util_syslib.o' into 'maskpass'; 'gtmtls' and 'gtm_tls_impl.o' into 'libgtmtls.so'; and so on). Compilation and linking of shared libraries and executables occur in one step rather than individually. Each target announces the actions it is going to perform. The 'install' target copies the maskpass executable to the plugin's gtmcrypt subdirectory and safely overwrites the existing libgtmcrypt.so softlink rather than erroring out if it is present. AIX installations no longer print warnings about duplicate symbols. Targets correctly specify and resolve internal dependencies to minimize the risk of errors or redundant operations. [UNIX] (GTM-8153) ✓

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.

DBCOLLREQ

DBCOLLREQ, JOURNAL EXTRACT proceeding without collation information for globals in database. eeee ffff .

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 ffff because of error eeee

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.

HLPPROC

HLPPROC, Helper Process error

MUPIP Error: GT.M replication was not able to start a helper process.

Action: Ensure that the gtm_dist environment variable points to a valid GT.M distribution that is executable by the user.

INSNOTJOINED

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

Receiver Server log/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 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.

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.

LDBINFMT ⚠

LDBINFMT, Unrecognized header for load file

MUPIP Error: This message identifies a MUPIP load file that is not having the correct header format in either BINARY, ZWR or GO format.

Action: Examine the file with a text editor for possible correction to format header. If fixing the header does not resolve the error, attempt MUPIP EXTRACT with a different file format.

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.

MAXBTLEVEL

MAXBTLEVEL, Global ^gggg in region rrrr reached maximum level

Run Time/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 a different database region.

NOEDITOR

NOEDITOR, Can't find an executable editor: eeee

Run Time Error: The ZEDIT command cannot find an executable editor.

Action: Ensure that the EDITOR environment variable points to an editor that is executable by the user.

NOTGBL

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

Run Time/MUPIP Error: This indicates that the VIEW argument expression for tracing specifies xxxx which 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.

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 were shut down while other database files in this instance file were still active.

Action: Restart the source server.

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

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. The facility that produces 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.

TRESTNOT

TRESTNOT, Cannot TRESTART, transaction is not restartable

Run Time Error: This indicates that a TRESTART command attempted to RESTART a transaction that did not enable RESTART; the error occurs at the point of the initial TSTART, so all global updates within the transaction are rolled back and any local variables specified have their values as of the TSTART.

Action: Enable RESTART with the initial TSTART command argument, add external LOCKs to serialize the transaction, or eliminate the TRESTART command.

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.

UPDPROC

UPDPROC, Update Process error

MUPIP Error: GT.M replication was not able to start the update process.

Action: Ensure that the gtm_dist environment variable points to a valid GT.M distribution that is executable by the user.