GT.M

Release Notes

V6.2-002

Empowering
the Financial World

FIS

Contact Information

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

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

Legal Notice

Revision History		
Revision 1.5	5 February 2019	Updated the Platforms section to add AIX 7.1 TL 4 and AIX 7.2 as supported versions; Correct the maximum V6 database size.
Revision 1.4	13 October 2015	* In Platforms and under the entry for x86_64 GNU/Linux, added information and workaround about a bug in Linux 3.13 kernels that affects GT.M operations. * Improved the description of GTM-5894.
Revision 1.3	28 August 2015	In Recompiling the Reference Implementation Plugin, removed all references to creating symlinks and corrected the instructions for recompiling the reference implementation plugin.
Revision 1.2	29 June 2015	Updated for V6.2-002A. Added the release note for GTM-8376.
Revision 1.1	08 June 2015	Added release notes for GTM-8369, GTM-8370, and GTM-8371.
Revision 1.0	26 May 2015	V6.2-002 - First published revision.

Table of Contents

V6.2-002A

Overview

V6.2-002 brings a number of modest enhancements to GT.M. Improving performance:

* By reducing the number of dirty global buffers to be flushed in anticipation of an impending epoch, epoch tapers aim to ameliorate spikes in response time that can occur at epochs. (GTM-6638)

* $ORDER(gvn,-1) - "reverse dollar order" - of global variables is faster. (GTM-7917)

* Under conditions of high contention, processes holding locks exit faster (GTM-8228), processes acquire database critical sections more efficiently when the mutex queue slots are all used (GTM-8286), and lock acquisition is more efficient with substantially reduced impact on database throughput (GTM-8241).

* Code size reduction that also improves performance (GTM-8224).

Enhancements include:

* Intrinsic special variables: $ZUT provides a universal (across systems and across time zones) time stamp, and $ZHOROLOG extends $HOROLOG with additional pieces that provide microsecond resolution and time zone information. (GTM-7949)

* TLS for SOCKET devices benefits from enhancements to WRITE /TLS and $ZSOCKET(). (GTM-8219)

* The environment variable gtm_autorelink_ctlmax provides a control to set the number of unique routine names in relink conrol files. (GTM-8240)

VIEW "POOLLIMIT" functionality introduced as field test grade functionality in the production release V6.2-001 is considered production grade functionality in V6.2-002

As always, the release bring numerous smaller enhancements, including to the $ZQGBLMOD() function (GTM-8298), better handling of split $PRINCIPAL (GTM-5688), more helpful TPRESTART messages (GTM-8350), and more. Robustness is improved, especially in the triggers, and security has been tightened by dropping support for dubious edge cases (GTM-8278). These changes are described below.

Conventions

This document uses the following conventions:

Flag/Qualifiers	-
Program Names or Functions	upper case. For example, MUPIP BACKUP
Examples	lower case. For example: mupip backup -database ACN,HIST / backup
Reference Number	A reference number is used to track software enhancements and support requests. It is enclosed between parentheses ().
Platform Identifier	Where an item affects only specific platforms, the platforms are listed in square brackets, e.g., [AIX]

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 on IA64, GNU/Linux on x86 (32- and 64-bits_, and Solaris on SPARC.

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.

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 variety of UNIX/Linux implementations. 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	-	*V6.2-001 was the last GT.M release for this platform, which is no longer supported. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.*
IBM System p AIX	6.1, 7.1 TL 4, 7.2	Only 64-bit versions of AIX are Supported. 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. AIX 7.1 TL 5 is Supportable. 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 and 7; Ubuntu 12.04 LTS and Ubuntu 14.04 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 (2.6.32 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):** As a consequence of the change in numbering implemented by ICU, for ICU versions 49 and greater, enter the version number divided by 10, e.g., 5.2 for ICU version number 52.

Platform	Supported Versions	Notes
		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. To support the optional WRITE /TLS fifth argument (the ability to provide / override options in the tlsid section of the encryption configuration file), the reference implementation of the encryption plugin requires libconfig 1.4.x. As this is a higher level than that distributed with Red Hat Enterprise Linux 6 or Ubuntu 12.04 LTS, in order to use this feature of WRITE/TLS on those platforms with the reference implementation, please install libconfig 1.4.x, including the header files, and recompile the reference implementation of the reference implementation of the encryption plugin. A bug in the Linux 3.13 kernels used in Ubuntu 14.04 LTS (https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1502168) affects GT.M operation. As newer kernels do not exhibit this misbehavior, FIS recommends that you follow the Ubuntu LTS Enablement Stack procedure (https://wiki.ubuntu.com/Kernel/LTSEnablementStack) and use newer kernels to avoid the behavior until such time as the bug is fixed in the 3.13 kernels.
x86 GNU/Linux	Red Hat Enterprise Linux 6 and 7	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. To support the optional WRITE /TLS fifth argument (the ability to provide / override options in the tlsid section of the encryption configuration file), the reference implementation of the encryption plugin requires libconfig 1.4.x. As this is a higher level than that distributed with Red Hat Enterprise Linux 6 or Ubuntu 12.04 LTS, in order to use this feature of WRITE/TLS on those platforms with the reference implementation, please install libconfig 1.4.x, including the header files, and recompile the reference implementation of the reference implementation of the encryption plugin.

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 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).

* 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 V6.2-002A in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.2-002A_arch (for example, /usr/lib/fis-gtm/V6.2-002A_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.2-002A_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.
* 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.

Recompiling the Reference Implementation Plugin

If you plan to use database encryption and TLS replication, FIS recommends that you recompile the reference implementation plugin. Recompiling the reference implementation plugin is recommended because the shared library dependencies in the pre-compiled reference implementation in GT.M distributions may not always match the shared libraries available on your platform. Recompile the reference implementation plugin as follows:

1. Install the header files for libgcrypt, libgpgme, libconfig, and libssl. On Linux, the header files are usually available in package libraries that have a -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
   ```

   The package names vary by distribution / version.

2. Note down the file permissions of the *.so files in $gtm_dist/plugin directory.

3. 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
   ```

4. Open Makefile. Review and 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 `pwd`
   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 `pwd`
   endif
   ```

5. With the absolute path to the directory where GT.M is installed set in the gtm_dist environment variable, (a) recompile the reference plugin, and (b) as root, replace the plugin in the $gtm_dist/plugin directory, for example:

   ```
   make
   sudo make uninstall && sudo make install
   make clean
   ```

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

7. Your reference implementation plugin is recompiled successfully.

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.

Upgrading to GT.M V6.2-002A

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 V6.2-002A 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 V6.2-002A 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 V6.2-002A.
* 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 V6.2-002A.
* 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 992Mi 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 V6.2-002A 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. A database file created with V6.0-000 (or above) can grow up to a maximum of 992Mi blocks. This means that, for example, the maximum size of a V6 database file having 8KiB block size is 7936GiB (8KiB*992Mi).

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

V6.2-002A does not require new replication instance files if you are upgrading from V5.5-000. However, V6.2-002A 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 GT.M 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 GT.M 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 V6.2-002A 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 V6.2-002A 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 V6.2-002A 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/V6.2-002A.

To extract and reapply the trigger definitions on V6.2-002A 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 V6.2-002A, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.2-002A 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 V6.2-002A.
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 V6.2-002A 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_V6.2-002A_i686, then gtmprofile and gtmcshrc set $gtm_dist to /usr/lib/fis-gtm/gtm_V6.2-002A_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 GT.M 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 GT.M 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 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

V6.2-002A

Fixes and enhancements specific to V6.2-00A are:

Id	Prior Id	Category	Summary
GTM-8376	-	MUMPS	Fix certain cases of $SELECT() within a Boolean expression

V6.2-002

Fixes and enhancements specific to V6.2-002 are:

Id	Prior Id	Category	Summary
GTM-1759	C9812-000745	MUMPS	See GTM-8239
GTM-5428	S9E05-002459	MUMPS	See GTM-7949
GTM-5688	C9F04-002718	MUMPS	$ZPIN and $ZPOUT allow USE and $ZSOCKET() to access one side of a split device $PRINCIPAL
GTM-5894	C9G06-002801	MUPIP	Option to preallocate the database file space on disk
GTM-6112	C9I01-002941	MUMPS	See GTM-8239
GTM-6524	C9K01-003223	MUMPS	Fix stack alignment on Linux X86-64 to prevent rare process failures.
GTM-6638	C9K06-003289	DB	Reduce epoch duration by minimizing the number of dirty buffers to flush
GTM-7409	-	DB	See GTM-6638
GTM-7725	-	MUMPS	Fix for special case in $$ZHOROLOG^%POSIX in GT.M POSIX plugin
GTM-7783	-	MUPIP	Source Server Abnormal Exit Logging
GTM-7894	-	MUMPS	See GTM-6524
GTM-7917	-	DB	$ORDER(gvn,-1) speed improvement
GTM-7949	-	MUMPS	$ZUT ISV for UTC and $ZHOROLOG ISV for microseconds and offset from UTC
GTM-8041	-	Other Utilities	Fix Debian Bug #775302 associated with fis-gtm-6.2-000
GTM-8071	-	DB	Add trigger ISV $ZTDELIM

Change History

Id	Prior Id	Category	Summary
GTM-8087	-	MUMPS	ZSHOW "C" lists loaded external call packages and routine contents
GTM-8167	-	MUMPS	Fixes to JOB command for long and many arguments
GTM-8183	-	DB	Protect against JNLBADRECFMT errors resulting from file block size larger than the OS page size
GTM-8187	-	MUPIP	MUPIP REORG -TRUNCATE improvement to return more space
GTM-8197	-	MUMPS	Better error for triggers specified with long names as arguments
GTM-8214	-	DB	Fix an edge case in $ZTRIGGER() and MUPIP TRIGGER
GTM-8219	-	MUMPS	Additional functionality for WRITE /TLS and $ZSOCKET()
GTM-8221	-	MUMPS	JOBLVN2LONG message no longer reports sizes
GTM-8224	-	MUMPS	Cleaner handling of complex ZBREAK management and better REQRLNKCTLRNDWN messages
GTM-8228	-	MUMPS	Minimize the impact of a mass exodus of processes holding LOCKs
GTM-8229	-	MUMPS	GT.M correctly handles I/O errors when its stdout and stderr are conjoined
GTM-8231	-	MUMPS	VIEW and $VIEW() fixes, particularly related to POOLLIMIT
GTM-8232	-	MUPIP	Prevent MUPIP from interacting with shell utilities in ways that disrupt terminal characteristics
GTM-8235	-	Other Utilities	GT.M reference encryption plugin works with GPG libraries compiled on AIX
GTM-8237	-	MUMPS	$ZSEARCH() skips symbolic links that result in loops
GTM-8238	-	MUMPS	ZRUPDATE better handles specific types of arguments and recognizes a question-mark (?) wild-card character
GTM-8239	-	MUMPS	Prevent extraneous newlines to $PRINCIPAL; correctly maintain $X and $Y when mixing output from stderr and stdout
GTM-8240	-	MUMPS	Setting the maximum number of auto-relink routine entries
GTM-8241	-	MUMPS	Improvement in M LOCK efficiency
GTM-8242	-	DB	Fix an interaction between a KILL with a trigger and MUPIP REORG TRUNCATE or ROLLBACK ONLINE

Change History

Id	Prior Id	Category	Summary
GTM-8245	-	DB	Fix MERGE gvn2=gvn1 when gvn1 falls in a spanning node and has an effective key shorter than that of gvn2
GTM-8246	-	MUMPS	$ZWIDTH() respects [NO]BADCHAR setting
GTM-8247	-	Other Utilities	^%RSEL recognizes auto-relink directories
GTM-8249	-	MUMPS	JOB command takes 32 arguments instead of 28
GTM-8256	-	DB	Fix to a rare condition that could damage data SET from within trigger code
GTM-8258	-	MUMPS	Fix VIEW "NOISOLATION":strexpr where strexpr is a variable containing only "+" or "-"
GTM-8261	-	MUMPS	VIEW "FULL_BOOLEAN" and variants take effect immediately for indirection and XECUTE
GTM-8269	-	DB	Fix edge case involving NOISOLATION, a TP transaction with SET and KILL in the same block as a concurrent SET
GTM-8272	-	Other Utilities	PINENTRY routine robustness improvements
GTM-8273	-	MUMPS	Using ZPrint or $TEXT() with relative entryrefs works correctly
GTM-8274	-	Other Utilities	See GTM-8275
GTM-8275	-	Other Utilities	Improvements to the reference encryption plug-in that deal more efficiently with GnuPG
GTM-8277	-	DB	UPGRADE and other MUPIP TRIGGER or $ZTRIGGER() operations make identical trigger definitions
GTM-8278	-	DB	More restrictive enforcement of authorizations in certain unusual cases
GTM-8280	-	MUMPS	Auto-relink facility appropriately manages memory and authorization of new objects
GTM-8282	-	MUMPS	Any time spent processing an INTRPT counts against a LOCK timeout
GTM-8286	-	DB	Better critical section management under unusual and stressful conditions
GTM-8287	-	Other Utilities	DBCERTIFY corrections
GTM-8290	-	MUMPS	Fix an occassional SIG-–11 caused by recursively relinking a routine containing ZBREAK breakpoints
GTM-8291	-	DB	Fix $ORDER() within a global subscript within a Boolean expression and with a non-literal direction expression

Change History

Id	Prior Id	Category	Summary
GTM-8295	-	MUMPS	Timed commands won't end early by up to one millisecond
GTM-8298	-	DB	New journal files on a switchover and other changes to make $ZQGBLMOD() less pessimistic
GTM-8299	-	DB	Ensure timely Journal Flush after MM database extension
GTM-8300	-	DB	LASTWRITERBYPAS Warning
GTM-8303	-	MUPIP	Prevent segmentation violations from terminating a MUPIP process with a signal, such as SIGTERM
GTM-8306	-	MUMPS	$ZC acts as a synonym for $ZCSTATUS
GTM-8315	-	Other Utilities	gtmprofile corrcetly configures gtm_icu_version
GTM-8316	-	Other Utilities	DSE no longer recognizes the -SPIN_SLEEP_CNT qualifier for CHANGE -FILEHEADER, as it has no effect
GTM-8317	-	MUMPS	INVTMPDIR and its avoidance by gtmsecshr
GTM-8320	-	MUMPS	GTM_FATAL_ERROR* files have a .txt extension and avoid an occassional SIG-11
GTM-8330	-	MUMPS	Expediciously deal with processes holding auto-relink resources for inappropriate times
GTM-8332	-	MUPIP	Ensure journal time stamps are always increasing
GTM-8333	-	MUMPS	Prevent a <CTRL-C> early in process startup from causing a SIG-11
GTM-8334	-	MUPIP	Improved idenpotency for MUPIP JOURNAL -ROLLBACK and -RECOVER
GTM-8335	-	DB	Incorrect upgrades from V5.0-000, V5.3-003, V5.3-004
GTM-8339	-	MUMPS	GT.M processes with open PIPE devices terminate promptly when interrupted by a terminal signal
GTM-8350	-	DB	TPRESTART messages from GT.M processes include the $ZPOSITION information
GTM-8360	-	MUPIP	Assorted enhancements to MUPIP EXTRACT and MUPIP LOAD operations on a binary extract
GTM-8364	-	MUMPS	Prevent <CTRL-C> from killing a process as it is starting
GTM-8365	-	MUMPS	Close a small window where an interrupt could cause a process to terminate
GTM-8369	-	MUMPS	Fix file positioning for sequential disk files greater than 4GiB

Change History

Id	Prior Id	Category	Summary
GTM-8370	-	MUMPS	ZSHOW works correctly when a MERGE command invokes a trigger which results in a runtime error and ZGOTO transfers control out of trigger code
GTM-8371	-	MUMPS	$QUERY(lvn) works correctly in the case lvn was previously the target of a ZSHOW "V" when no variables existed.

M-Database Access

* Intended to reduce spikes in response time at epochs caused by flushing global buffers and the filesystem cache (using fsync()), the database segment setting [NO]EPOCHTAPER controls whether GT.M tries to minimize epoch duration by reducing the number of dirty buffers in anticipation of an approaching epoch (time-based or due to a journal file auto-switch). GDE ADD and CHANGE REGION, DSE CHANGE FILEHEADER and MUPIP SET all have -[NO]EPOCHTAPER qualifiers. As a setting of EPOCHTAPER (i.e., to taper) resulted in throughput equal to or better than NOEPOCHTAPER in all workloads tested, it is the default. A setting of NOEPOCHTAPER reverts to the prior behavior should you find that preferable.. (GTM-6638)

* See GTM-6638. (GTM-7409)

* $ORDER(gvn,-1) has a code path only slightly longer than $ORDER(gvn,1); previously a longer code path made the reverse $ORDER(), or $ZPREVIOUS(), take close to twice the time of a $ORDER() for the identical gvn. Also, MERGE and $QUERY() show a modest improvement. (GTM-7917)

* Within a SET trigger context, $ZTDE[LIM] returns the piece separator, as specified by -delim in the trigger definition. This allows triggers to extract updated pieces defined in $ZTUPDATE without having the piece separator hard coded into the routine. Outside of a SET trigger context, $ZTDELIM is null. Previously GT.M did not implement $ZTDELIM. (GTM-8071)

* GT.M appropriately handles a case where the file system block size exceeds the operating system page size. Previously, such a situation caused journal file damage (e.g. JNLBADRECFMT errors) on rare occasions. This was only encountered in the GT.M development environment, and was never reported by a user. (GTM-8183)

* $ZTRIGGER() and MUPIP TRIGGER work reliably; previously they could occasionally produce segmentation violations (SIG-11) or garbling of trigger names. This issue was discovered in the GT.M development environment and was never reported by any user. (GTM-8214)

* KILL appropriately handles the case where there the target gvn has a KILL trigger and there is a concurrent MUPIP REORG TRUNCATE or ROLLBACK ONLINE that moves the root block of the target gvn. Previously, in this unusual case the target gvn might not be killed. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8242)

* MERGE gvn2=gvn1 where gvn1 falls in a spanning node, either because of its own key-value length or the key-value length of one of its descendent nodes that starts in the same data block as gvn1, works correctly. Previously, if the block size of the region holding gvn2 was less than that of the region holding gvn1 or $LENGTH($VIEW("YGVN2GDS",gvn2))>$LENGTH($VIEW("YGVN2GDS",gvn1)), such a MERGE damaged gvn2 so that a subsequent reference to it resulted in a TPFAIL error with retry codes uuuu. (GTM-8245)

* SETs within trigger code that updates nodes having the same unsubscripted name as the node with the trigger work reliably; previously there was a small chance such a SET placed a garbled value in the destination. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8256)

* GT.M correctly handles a KILL of a NOISOLATION global node within a TP transaction where the same transaction had SET at least one node in the same GDS block as the node being KILL'd. Previously, if a concurrent SET from another process happened at the same time as the KILL and both updated the same GDS block and the KILL action made the block empty, GT.M incorrectly applied the NOISOLATION optimization which meant that the concurrent SET was lost. Note that any missing SET was captured in the journal and by replication. (GTM-8269)

* Trigger upgrades with MUPIP TRIGGER UPGRADE and trigger maintenance with $ZTRIGGER() and MUPIP TRIGGER actions, other than UPGRADE, refer to the same trigger. In V6.2-001, identical triggers could appear as duplicates, and attempts to delete triggers that had been upgraded did not work. The workaround was to delete all triggers with a "-*", and then load the correct set of triggers. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8277)

* When execution of GT.M is restricted to members of a group, regardless of database file owner and group, and independent of whether the owner is a member of the group, shared resources never have world permissions. When the owner is not a member of the group, and when execution of GT.M is restricted to members of a group, the group of all GT.M-generated shared resources is the group permitted to execute GT.M. Previously, in this edge case, GT.M created shared resources with world access. GT.M sets permissions for shared

resources (journal files, shared memory, and semaphores) that are rational, but as restrictive as possible, in certain implausible but technically feasible edge cases; for example: a database file with read-only access for an owner, who is not a member of its group, with read-write group access. Previously, in such edge cases, it was possible for journal file and shared memory permissions to be impractical; e..g, with the combination of permissions above, a journal file created by a user who is a member of the group (but not the owner of the database) could have read-only permissions for the creator of the journal file. Note that FIS recommends against attempting such dubious group memberships and file permissions. In addition, When installing GT.M with the option to restrict GT.M access to members of a group, the configure script terminates with the GROUPNOTVALID error if the group is root or bin; and with the NOTINGROUP error if owner is not (a) root, (b) bin, or (c) a member of the group. Previously, the configure script did not check for these unusual combinations. (GTM-8278)

* GT.M deals appropriately with a situation when high contention for a database resource results in all available mutex_slots being used. Previously such an unusually high contention could cause processes to inappropriately wait a couple of seconds for a resource even after it became available. Note that in releases V6.0-000 through V6.1-000, a mutex queue slot leak, fixed in V6.2-000 with the tracking number GTM-7804, a leak in mutex queue slots could cause the same behavior by causing all slots to appear to be in use. (GTM-8286)

* $ORDER(gvn,expr) correctly handles the case where expr is not a literal; previously if a function with these characteristic appeared in a global subscript within a Boolean expression, this somewhat unusual case caused a segmentation violation (SIG-11). (GTM-8291)

* When an instance changes roles from a secondary (also known as a propagating primary) to a primary (also known as a root primary), the first Source Server process on the primary, or the MUPIP REPLICATE SOURCE ACTIVATE command for on-the-fly switching, creates a new set of journal files for all journaled regions (which are also all replicated regions) in the global directory. When called by lost (unreplicated) transaction processing logic on the root primary, the new journal files make $ZQGBLMOD(gvn) less likely to report a false positive (1) as to whether the node gvn was changed by a local update since the instance became the new root primary. Note that by design $ZQGBLMOD() must be pessimistic, in that while a return value of 0 means that the node has not changed, a return value of 1 reports that the node may have changed. For applications that use $ZQGBLMOD() to attempt automatic lost (unreplicated) transaction processing, making $ZQGBLMOD() less pessimistic means that falling back to manual reconciliation is less likely. Starting a Receiver Server in a secondary instance and allowing it to communicate with the Source Server no longer resets $ZQGBLMOD() on the primary instance. Although deployments usually perform a ROLLBACK FETCHRESYNC before starting up a Receiver Server, starting a secondary instance with AUTOROLLBACK, or inadvertently forgetting to perform a ROLLBACK FETCHRESYNC previously reset internal state information used to compute $ZQGBLMOD() so that all calls to $ZQGBLMOD() thereafter returned 1, thereby requiring manual reconciliation for all lost (unreplicated) transaction processing using $ZQGBLMOD() to make this determination. The workaround was to first perform a separate ROLLBACK FETCHRESYNC step, before starting replication. In addition, $ZQGBLMOD() works on Supplementary Instance (SI) replication as it does for Business Continuity (BC) replication; previously ROLLBACK always caused $ZQGBLMOD() return 1 in SI environments. This was only encountered in the GT.M development environment and was never reported by a user. Note that lost (unreplicated) transaction processing is always the responsibility of the application designer. GT.M lacks the knowledge of application design required to determine whether a zero return value means that it is safe to apply the update, and $ZQGBLMOD() is only a tool to assist the application. (GTM-8298)

* For database segments opened with the MM access method, GT.M ensures it continuously flushes journal buffers. Previously after at least one database file extension, when there were no active updates, occasionally GT.M did not flush the journal buffers until subsequent updates filled journal buffers, a VIEW "JNLFLUSH", or the exit of the last process with the database open for writing. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8299)

* In a case where the last process with a database open for writing bypasses the rundown on exit, leaving associated the shared memory open, allocated and potentially unflushed, GT.M sends a LASTWRITERBYPAS warning message to the system log. This can occur if the instance is frozen at the time of the process exit. Previously, GT.M did not provide this notification. (GTM-8300)

* GT.M performs automatic database upgrades for all V5 and V6 versions. Previously, (a) versions subsequent to V5.4-002A did not properly upgrade databases created in or upgraded to V5.3-003 and V5.3-004; and (b) databases created in or upgraded to V5.0-000 had to first be upgraded to V5.1-000. If you need to upgrade a V5.3-003 or V5.3-004 database to a version prior to V6.2-002, first upgrade to V5.4-002A and then, as a second step upgrade, to the target version. If you need to upgrade a V5.0-000 database to a version prior to V6.2-002, first upgrade to V5.2-000, and then to the version prior to V6.2-002. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8335)

* TPRESTART messages from GT.M processes include the $ZPOSITION of the line of M code that caused the restart of the transaction; utilities leave this field blank. Previously, the restart recording facility did not report this information. (GTM-8350)

M-Other Than Database Access

* See GTM-8239 (GTM-1759)

* See GTM-7949. (GTM-5428)

* When $PRINCIPAL input is a terminal device and $PRINCIPAL output is /dev/null GT.M directs WRITEs to $PRINCIPAL to /dev/null; previously, it sent such output to the $PRINCIPAL input terminal. When $PRINCIPAL input and output are different sockets, GT.M treats them as separate input and output SOCKET devices; previously, it set up a SOCKET device with only the input socket so output went to the input socket. When $PRINCIPAL has different input/output devices, the USE command recognizes intrinsic special variables $ZPIN or $ZPOUT and applies deviceparameters to the input or output side of $PRINCIPAL, respectively. USE $PRINCIPAL continues to apply appropriate deviceparameters to both input and output devices. A USE with any of the $PRINCIPAL variants sets $IO to $PRINCIPAL. Previously, it was not possible to modify a split device separately. $ZSOCKET() also accepts $ZPIN or $ZPOUT as its first argument and, if the device is a split SOCKET device, supplies information on input or output side, respectively. In any context other than USE or $ZSOCKET(), or if $PRINCIPAL is not a split device, $PRINCIPAL, $ZPIN and $ZPOUT are synonyms. In the case of a split $PRINCIPAL, $ZPIN and $ZPOUT return the value of $PRINCIPAL followed by the strings "< /" and "> /", respectively. Any attempt to OPEN $ZPIN or $ZPOUT results in a DEVOPENFAIL error. Note: if $PRINCIPAL input and output are different SOCKETs, WRITE /WAIT applies to the input side of the device. (GTM-5688)

* See GTM-8239 (GTM-6112)

* On Linux x86-64 the C stack is aligned to 16 bytes. Previously problems such as SIG-10 bus error or SIG-11 segmentation faults could occur as a consequence of an 8-byte stack alignment. These were only encountered in the GT.M development environment, and were never reported by a user. [Linux x86-64] (GTM7894)(GTM-6524)

* $$ZHOROLOG^%POSIX returns the correct number of seconds and microseconds. Previously, when the number of microseconds was 0, the function reported ten times the seconds count with no fractional part. This functionality is actually provided by the POSIX plugin rather than core GT.M, and was release in version r1 of the GT.M POSIX plugin, which is released independently. This release note appears here as a matter of record. Note also that the new intrinsic special variables $ZHOROLOG and $ZUT may provide you with comparable functionality at a lower computational cost. (GTM-7725)

* See GTM-6524(GTM-7894)

* $ZUT (UNIX time, or universal time) returns the number of microseconds since January 1, 1970 00:00:00 UTC, which provides a time stamp for directly comparing different timezones. $ZUT accuracy is subject to the precision of the system clock. $ZH[OROLOG] returns four comma-separated numbers (e.g. "63638,39194,258602,14400"). The first two numbers are identical to $HOROLOG, and for existing application code of the form $PIECE($HOROLOG,",",...), $ZHOROLOG is a drop-in replacement. The third is the number of microseconds in the current second, and the fourth is the number of seconds to be added to the time reported in the first two numbers to determine the UTC time. Note that the fourth number can be positive or negative, and because the time zone of the computer can have a different date from UTC, generating a UTC time from it may require adjustment to both first and second pieces. (GTM-5428) (GTM-7949)

* ZSHOW "C" provides the list of loaded external call packages and their routines. Packages that are accessible but have not been accessed are not reported. When a local or global variable is targeted for output, the package and routine names are placed as subscripts for ease of programmatic manipulation. ZSHOW "C" is included in ZSHOW "*". (GTM-8087)

* The JOB command accepts up to 28 routine arguments, each of which may be up to 8192 bytes long. If any argument exceeds the limit, the JOB command issues a JOBPARTOOLONG error message. Previously, the limit was 8190 bytes, but arguments exceeding 1024 may have been truncated, included extraneous data, or resulted in a SIGABRT, and a JOB command exceeding the former 8190 byte limit incorrectly issued a JOBFAIL error message along with EBADF indicating: Job error in socketpair. (GTM-8167)

* GT.M limits trigger names used as arguments to ZBREAK, ZPRINT and $TEXT() to 30 characters in order to maintain the trailing pound-sign (#), which acts as an identifier that the object named is a trigger. Previously, attempts to provide trigger name arguments of 31 or more characters to these features caused them to issue a ZLINKFILE error rather than a TRIGNAMENF error. In addition, GT.M

recognizes the region-name selector on long trigger names in regions with long region names; previously if the trigger existed but the combination of the trigger name and selector exceeded 31 characters, GT.M inappropriately produced a TRIGNAMENF error. (GTM-8197)

* The optional fourth expression (after the tlsid) in the argument of a WRITE /TLS command specifies an obfuscated password for the private key in the tlsid section of a configuration file. This key overrides any existing password. Previously GT.M required an environment variable of the form gtmtls_passwd_<tlsid> to specify an obfuscated password. The optional fifth expression in the argument of a WRITE /TLS command specifies configuration file options that add to or replace existing options in the section labeled tlsid. If there is no section with the label tlsid, GT.M creates a new virtual section, which persists for the life of the process. These options use the same format as the configuration file including the terminating semi-colon (";"). The format for WRITE /TLS is:

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

When the second argument of $ZSOCKET() specifies "TLS" as the keyword expression, $ZSOCKET() returns a string of the form "1,<SERVER|CLIENT>[,<tlsid>]" when the SOCKET is using TLS or the empty string if it is not. The optional tlsid comes from the WRITE /TLS which enabled TLS on the specified socket. The optional fourth argument specifies that $ZSOCKET() return additional information where the argument may contain (case-insensitive) "CIPHER" and/or "OPTIONS" separated by a comma. The returned information for "CIPHER" is the TLS protocol prefixed by "P:" and the algorithm prefixed by "C:" and the information for OPTIONS is prefixed by "O:" delimited by vertical-bars. For example to return the TLS options and cipher on the current socket in the device specified by the local variable S:

```
WRITE $ZSOCKET(S,"TLS",,"options,CIPHER")
```

The missing third argument specifies the current socket. (GTM-8219)

* JOBLVN2LONG error message has been changed, and it no longer includes the maximum and encountered ZWRITE representation lengths. Previously, it was reporting both sizes as 0 from the parent process. The correct sizes were visible from the child process only. The child process now reports an additional informational message, JOBLVNDETAIL, which includes those two sizes. (GTM-8221)

* GT.M handles interactions between ZBREAK and recursive relink appropriately. Previously, recursively linking a new version of an active routine, inserting ZBREAKs, relinking it again, could cause a segmentation violation (SIG-11) or illegal instruction (SIG-4) when a routine which had linked a pre-ZBREAK version of the routine with the ZBREAKs attempted to invoke it anew after the ZBREAK version had disappeared from the stack. In addition, GT.M issues a more appropriate REQRLNKCTLRNDWN error message in case an actively used rtnobj shared memory segment gets deleted (by say an "ipcrm -m") and a new process tries to attach to that; previously one would get a RELINKCTLERR error message. Also, in case an actively used relinkctl shared memory segment gets deleted, the REQRLNKCTLRNDWN message has more detail indicating the shmid and the actual error returned by the "shmat()" system call; previously the additional detail was absent. As result of this effort GT.M produces somewhat smaller object files with a small uptick in performance. (GTM-8224)

* On exit, processes release their M locks in a fast operation; previously it was possible for the release to delay process termination. (GTM-8228)

* GT.M correctly handles I/O errors when its stdout and stderr are conjoined to a file, FIFO, or PIPE device. Previously, when a process's stdout and stderr were conjoined to a file, FIFO, or PIPE device, a failure to write to either of the streams, for instance in a broken pipe situation, could trigger a segmentation violation (SIG-11). (GTM-8229)

* $VIEW("POOLLIMIT") with a missing size argument defaults to 0 (100% availability) and with a missing or asterisk (*) region argument, applies to all regions; previously such inputs produced a segmentation violation (SIG-11). In addition, VIEW("POOLLIMIT") for database regions with access methods other than BG returns a zero (0); in V6.2-001, it could inappropriately return a non-zero value. Also, error messages that contain VIEW/$VIEW() problematic arguments use ZWRITE format to display non-graphic characters, and GT.M appropriately handles errors with VIEW/$VIEW() region names; in V6.2-001 such errors could damage local storage of the bad name. (GTM-8231)

* $ZSEARCH() skips symbolic links that result in loops; previously it gave an error when the search encountered such a link. (GTM-8237)

* ZRUPDATE rejects file-name arguments that are symbolic links or start with a percent-sign (%); previously it accepted file names starting with '%' as well as symbolic links with a '.o' extension that resolved to any file. In addition, ZRUPDATE recognizes question-mark (?) as a single character wild-card; previously it did not. Also, ZRUPDATE always updates the existing shared memory relinkctl information for a

file with an existing entry; previously if the process had never previously accessed the directory containing the entry and the directory no longer existed, ZRUPDATE inappropriately failed to maintain the relinkctl memory. (GTM-8238)

* GT.M does not send extraneous newlines to principal devices that are files, PIPEs, or FIFOs. Previously, in certain situations, such as during a ZSYSTEM command, or when processing an error, GT.M could send extraneous newlines to file, pipe or FIFO principal devices (but not to terminal principal devices). Additionally, if the stderr and stdout of a process are conjoined to a terminal or a single file (e.g., with shell redirection), $X and $Y are correctly maintained, and application output to stdout and error messages to stderr are correctly merged. Previously, under such circumstances $X and $Y did not reflect the output to stderr, and lines written to stdout and stderr could be commingled. Finally, receiving a signal, such as SIGTERM, while processing a READ command does not result in an indeterminate hang. Previously, there was a small window in processing of the READ command, receiving a signal during which could cause GT.M to hang indefinitely. Please review scripting that captures and compares GT.M output to determine whether it is affected by this fix. (GTM-1759)(GTM-6112)(GTM-8239)

* The environment variable gtm_autorelink_ctlmax specifies the maximum number of entries for unique routine names in the relink control file created by a process for any directory, with a minimum of 1,000, a maximum of 16,000,000 and a default of 50,000 if unspecified. If a specified value is above or below the allowed range, the process logs the errors ARCTLMAXHIGH or ARCTLMAXLOW respectively in the syslog, and uses the nearest acceptable limit instead. MUPIP RCTLDUMP and ZSHOW "A" outputs include the maximum number of unique routine names available in a relink control file. Previously (effective V6.2-001, when auto relink became production grade software), relink control files for each directory had a fixed capacity of 1,000,000 unique routine names. Note that any application code or scripting that parses the output of MUPIP RCTLDUMP or ZSHOW "A" needs to be changed. [AIX, Linux x86_64, Solaris] (GTM-8240)

* Acquisition attempts by processes of M LOCKs held by other processes are lightweight with little impact on database throughput. Previously, large numbers of processes - hundreds to thousands - waiting for LOCKs held by existing processes and mapped to the shared memory of a database region could interfere with one another and negatively impact database activity for the region. (GTM-8241)

* $ZWIDTH() respects the [NO]BADCHAR setting; previously it ignored this setting and always reported invalid UTF-8 characters. (GTM-8246)

* The JOB command accepts up to 32 arguments and issues MAXACTARG error on the child's error stream if the argument count is beyond the limit. The parent process continues to receive a JOBFAIL if the number of arguments exceeds the limit. Previously, the limit was 28 and exceeding the limit caused a GTMASSERT. (GTM-8249)

* VIEW "NOISOLATION":strexpr reports a VIEWGVN error if the strexpr is not a literal and contains only a plus-sign or minus-sign; previously this caused a segmentation violation (SIG-11). (GTM-8258)

* Setting VIEW "[NO]FULL_BOOLEAN[WARN]" works appropriately in certain rare cases involving indirection and direct mode, previously executed expressions retained the behavior from the setting in effect at the time the code was compiled for execution. Also, compilation does not result in a segmentation violation (SIG-11) in rare cases when switching back and forth between Boolean modes. (GTM-8261)

* Inside a trigger, ZPRINT and $TEXT() work correctly when using just label and offset for the entryref. Previously, neither ZPRINT nor $TEXT() would work correctly unless the trigger routine name was included in the entryref. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8273)

* The auto-relink facility appropriately manages memory when it encounters errors; previously under unusual conditions it could leak memory. In addition, the facility carefully manages the propagation of authorizations to new objects; previously in unusual circumstances it managed the permissions incorrectly. MUPIP RUNDOWN -RELINKCTL when invoked with an argument that does not resolve to an existing directory and does not match a previously existing directory with a left-over relinkctl file produces a FILEPARSE error; previously such invocations did not produce any output. MUPIP RUNDOWN -RELINKCTL for the every auto-relink-enabled directory in $gtmroutines, or specified as an argument when no corresponding relinkctl file is found, follows the RLNKCTLRNDWNSUC message with a TEXT message saying "No relinkctl file found"; previously MUPIP did not provide the supplementary message. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8280)

* Measurement of time for M LOCK timeouts is consistent with that of other timed operations, and only includes the total elapsed time, although it makes a final attempt after returning from an interrupt during which the time elapsed. Previously it included the elapsed time

plus any time spent handling interrupts, both interrupts internal to GT.M operation as well as executing $ZINTERRUPT(). This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8282)

* Recursive relink of a routine that has one or more breakpoints set works correctly. In GT.M V6.2-001, this could occasionally cause the process to abnormally terminate with a segmentation violation (SIG-11). This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8290)

* Timed commands wait the full duration of the specified time. Previously, the time interval could (if interrupted just prior to completion by a MUPIP INTRPT or operations internal to the GT.M run-time logic) be up to 1ms less than the specified time in rare occasions. The most observable effect of such a small time interval was the possibility of two accesses to $HOROLOG separated by a HANG 1 returning the same value. (GTM-8295)

* WRITE or ZWRITE of $ZC now returns the same value as $ZCstatus. Previously WRITE $ZC always returned zero and ZWRITE $ZC produced a GTMASSERT2. (GTM-8306)

* GT.M sends an INVTMPDIR error to the system log when $gtm_tmp is invalid. In addition, GT.M bypasses $gtm_linktmpdir validation for the gtmsechsr image. Previously, if $gtm_tmp was invalid and $gtm_linktmpdir was not defined, GT.M incorrectly sent an INVLINKTMPDIR error to the system log. Also, if $gtm_linktmpdir was defined, an INVLINKTMPDIR error was sent to the operator when gtmsecshr was started because the gtmsecshr wrapper converted the value to the NULL string. (GTM-8317)

* In a fatal error situation, GT.M typically creates a GTM_FATAL_ERROR*.txt file before it exits. GT.M skips the creation of the .txt file for M stack overflows or a SIG-11 addressing exceptions. The .txt extension promotes the ability to easily open such files with common editors. Previously the name of a GTM_FATAL_ERROR* file did not end with a .txt extension, and, under certain very rare circumstances, GT.M could encounter a segmentation violation (SIG-11) when creating the file, which at least partially masked the original failure. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8320)

* GT.M deals with suspended processes holding shared resources used by auto-relink in a timely fashion. Note that you should not suspend a GT.M process using shared resources, and almost all GT.M processes do so. Previously, a process attempting to auto-relink, depending on the inappropriately held resource, could wait up to approximately twenty minutes. (GTM-8330)

* GT.M protects against a SIGINT (generated by a <CTRL-C> with the default key binding) while initializing a $PRINCIPAL terminal at process start up; previously, a SIGINT in a small window could cause a segmentation violation (SIG-11). (GTM-8333)

* GT.M processes with open PIPE devices terminate promptly when interrupted by a terminal signal, such as SIGTERM. Previously, sending a terminal signal to a GT.M process that had one or more open PIPE devices could under very rare circumstances result in an indefinite hang. This was only observed in the GT.M development environment and was never reported by a user. (GTM-8339)

* GT.M manages memory better during process start up. Previously, an interruption during a small window in memory allocation could terminate a process without any error messages or dump files. This was only observed in the GT.M development environment, and was never reported by a user. (GTM-8364)

* GT.M processes correctly handle interrupts (such as a timer pop, <CTRL-C>, or a MUPIP INTRPT). Previously interrupts received in very small windows of code could in rare cases terminate processes in various ways including segmentation violations (SIG-11) or GTMASSERTs. This was only observed in the GT.M development environment and was never reported by a user. (GTM-8365)

* OPEN and USE commands on sequential devices using deviceparameters APPEND, SEEK, and TRUNCATE work for files over 4GiB in size. In V6.2-000 and V6.2-001, the TRUNCATE mispositioned for files over 4GiB since V6.1-000. (GTM-8369)

* ZSHOW works correctly after a MERGE command that invokes a trigger (that is, the target of the MERGE is a GVN) which in turn encounters a runtime error and transfers control out of the trigger code using a ZGOTO. Previously, such a ZSHOW would abnormally terminate with a segmentation violation (SIG-11). (GTM-8370)

* $QUERY(lvn) works correctly in the case lvn was previously the target of a ZSHOW "V" when no variables existed. Previously this situation would cause $QUERY(lvn) to result in a segmentation violation (SIG-11). (GTM-8371)

* V6.2-002A $SELECT() works correctly in several scenarios where it previously did not.

* Effective V6.2-002, compiling nested calls to $SELECT() with both environment variables gtm_boolean and gtm_side_effects undefined or set to zero (0) could occasionally terminate processes with a segmentation violation (SIG-11).

* Effective V6.2-002, compiling nested calls to $SELECT() with gtm_boolean set to 1 or 2 could cause a GTMASSERT2.

* Effective V6.2-002, $SELECT() incorrectly maintained $REFERENCE and the naked reference in some cases.

* Effective V6.0-001, even with gtm_side_effects set to one (1) to specify that the code has side effects, GT.M would evaluate $SELECT() out of order. KILL x WRITE x+$SELECT(1:$INCREMENT(x)) is an example of code with side effects that did not maintain standard order of evaluation even when compiled with gtm_side_effects set to 1.

Although the above cases describe the core of the previous behavior, it is likely that there are cases that overlap the above in ways that are not obvious. When gtm_boolean and gtm_side_effects are undefined (or set to zero), GT.M assumes that the code has no side effects other than $REFERENCE and indirection, and can, and does, rearrange computation to improve performance: the evaluation of operands in expressions, the order in which function call parameters are evaluated, etc. In order to be more consistent with other optimizations, when neither gtm_boolean nor gtm_side_effects are set to non-zero values, GT.M V6.2-002A rearranges computation to evaluate a $SELECT() that appears within a Boolean expression only when its arguments contain a global variable or indirection, where previously it always did. If such a $SELECT() has an argument containing an extrinsic ($$) function call with side effects, results on V6.2-002A may differ from those on previous GT.M releases. Note: as the rearranging can change from release to release, you may encounter unexpected and varying results if you compile application code that has side effects with environment variables that tell GT.M that it has no side effects. Where code contains side-effects, FIS strongly recommends compiling it with gtm_side_effects and gtm_boolean set to 1. (GTM-8376)

Utilities-MUPIP

* GT.M provides an option to preallocate blocks from the file system when creating or extending a database file; by default UNIX file systems, and GT.M, use sparse (or lazy) allocation, which defers actual allocation until blocks are first written. The GDE segment qualifier -[NO]DEFER_ALLOCATE, with a default of DEFER_ALLOCATE, determines whether MUPIP CREATE preallocates blocks on database creation, and determines whether subsequent extensions also preallocate. The MUPIP SET qualifier -[NO]DEFER_ALLOCATE provides a mechanism to control GT.M behavior when subsequently extending existing database files, whether using MUPIP EXTEND or auto-extend. To switch an existing database file so it immediately preallocates all blocks, first use MUPIP SET -NODEFER_ALLOCATE to set the switch in the database file header, followed by MUPIP EXTEND -BLOCKS=n, where n >= 0. The DSE DUMP -FILEHEADER ("Defer allocation" field) and GDE SHOW ("DALL" field) show the setting of the switch for each database file. Failures to preallocate space produce a PREALLOCATEFAIL error. On platforms where GT.M does not support preallocation (HP-UX and Solaris), although GDE accepts -NODEFER_ALLOCATE, MUPIP CREATE ignores it and sets the database file to DEFER_ALLOCATE. On those platforms, any attempt to change this flag with MUPIP SET produces a NODFRALLOCSUPP error.

  In testing this enhancement, FIS exposed a bug in the Linux ext4 file system which is present in the upstream Linux kernel since at least 3.10, and which is fixed by kernel commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3). The Red Hat Bugzilla identifier for the bug is 1213487. We are also investigating the possibility of a different bug in ext3 and jfs filesystems that may be triggered by use of NODEFER_ALLOCATE. Although the bug appears to be very rare, it manifests itself as a database block which is all zeros (ext4), or zero data bytes (ext3 and jfs) resulting in data loss. Please avoid NODEFER_ALLOCATE for database files on Linux ext3, ext4, or jfs file systems. The default behavior of DEFER_ALLOCATE avoids the bugs. On Linux, place any database files for which you wish to use use NODEFER_ALLOCATE on xfs file systems.[AIX, Linux] (GTM-5894)

* The source server sends a REPLSRCEXITERR message to the system log when the server exits abnormally; previously GT.M did not provide notification of such an event. (GTM-7783)

* MUPIP REORG TRUNCATE always relocates global root blocks if there is room to relocate them nearer to the beginning of a database file; previously, it could leave them unrelocated under certain conditions thereby limiting the amount of space returned to the file system. The workaround was to repeat the operation as soon as possible. (GTM-8187)

* MUPIP commands that produce a lot of output (e.g. MUPIP RCTLDUMP, MUPIP REORG etc.) correctly maintain terminal characteristics in case they are used in a UNIX pipeline involving other tools (e.g. "more") that also play with terminal characteristics. Previously this could result in the terminal not echoing characters once the MUPIP command returned (more likely if the command was not allowed to run to completion). A workaround was to issue a "stty sane" to the shell on completion. (GTM-8232)

* Terminating a MUPIP process with a signal, such as SIGTERM, does not result in segmentation violations (SIG-11). Previously, sending a signal to a MUPIP process in a small window shortly after its start-up could cause a segmentation violation (SIG-11). This was only encountered in the GT.M development environment and was never reported by a user. [AIX, SunOS, Linux x86_64] (GTM-8303)

* GT.M ensures that timestamps of successive records in a journal file as well as across journal files that were switched on-the-fly (e.g. auto switches) never decrease. Previously this was not ensured which meant that in rare cases a later update could have an older/lesser timestamp in the journal record even though the system time did not go backward. This in turn could affect time related MUPIP JOURNAL commands (e.g. time-based backward recovery etc.) This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8332)

* MUPIP JOURNAL - ROLLBACK and -RECOVER handle being interrupted appropriately. Previously, in a rare combination of circumstances, it was possible for a rollback/recover to lose data, or issue incorrect EPOCHTNHI errors, after an interrupted attempt. This was only encountered in the GT.M development and testing environment, and was never reported by a user. (GTM-8334)

* MUPIP EXTRACT in binary format can include global variable nodes from both encrypted and unencrypted regions. When an extract includes data from one or more encrypted regions, MUPIP EXTRACT uses a level 8 format (the first 26 characters of the label are "GDS BINARY EXTRACT LEVEL 8"), which cannot be loaded by MUPIP LOAD of earlier GT.M releases. MUPIP LOAD attempts to read extracts of level 7 format using a heuristic approach, which can fail with an error, but we have found typically successful. Previously, extracts from V6.1-000, V6.2-000 and V6.2-001 (extract format level 7) that included both encrypted and unencrypted regions caused MUPIP LOAD

to behave badly, including termination with a segmentation violation (SIG-11) or loading incorrect data (the latter, while theoretically possible, was never reported by user). The workaround for loading an extract from any of those three versions was to create separate extracts for encrypted regions and unencrypted regions. Additionally, MUPIP LOAD can read extracts of level 5 format. Previously, MUPIP LOAD failed loading extracts of that version with an error. (GTM-8360)

Utilities-Other Than MUPIP

* The configure install script compiles M object files in UTF-8 mode as long as it finds at least one UTF-8 locale. Previously, the configure script required the en_US.UTF-8 locale in order to compile object files in UTF-6 mode. Also, the gtminstall script displays any errors produced by the configure script output; previously, the gtminstall script masked all configure script output inappropriately masking error information. These two changes fix Debian Bug #775302 associated with fis-gtm-6.2-000. (GTM-8041)

* On AIX, the reference implementation of the encryption plugin works as expected when using GPG libraries compiled using instructions from Appendix D (Building Encryption Libraries) of the GT.M Administration and Operations Guide.

  The reference implementation of the encryption plugin included in GT.M versions V5.5-000 through V6.2-001 referenced a non-standard compilation of the GPGME encryption library (libgpgme). Without the solution below, the administrators setting up the database will encounter either %GTM-E-CRYPTINIT or %GTM-E-CRYPTDLNOOPEN examples below, with extended detail indicating a failure to load GPGME or the encryption reference plugin library.

```
%GTM-E-CRYPTINIT, Could not initialize encryption library while opening encrypted file gtm.dat.
Error initializing GpgME: GPGME/No such file or directory

%GTM-E-CRYPTDLNOOPEN, Could not load encryption library while opening encrypted file ffff. Could not load module
 llll.
```

  The solution to this problem required either recompilation of the reference implementation encryption plugin or a modification of the installed GPGME library. FIS strongly recommends recompiling the reference implementation encryption plugin. Please see the section "Plugin Architecture & Interface" packaging in Chapter 12 (Database Encryption) of the GT.M Administration and Operations Guide for compilation instructions. If you have a reason that requires you to modify the GPG libraries after installation, and have GT.M support from FIS, contact your support channel. [AIX] (GTM-8235)

* %RSEL handles $ZOUTINES directories designated for auto-relink; previously it ignored such directories. (GTM-8247)

* The PINENTRY routine used for unattended password entry does not create temporary directories or files when $gtm_dist and $gtmroutines are properly specified. If the routine encounters an error, it hands off the password request to the system default pinentry program. The PINENTRY routine only fails when GTMXC_gpgagent is not defined correctly. Typically users encounter this problem during database setup. Previously, the PINENTRY routine always created temporary directories and files and did not support a hand off to the system pinentry program which resulted in a CRYPTKEYFETCHFAIL error. (GTM-8272)

* See GTM-8275.(GTM-8274)

* The reference implementation of the GT.M plug-in makes appropriately efficient use of GnuPG, under all conditions, but especially when multiple databases and devices use the same key file. Previously, recent versions of GnuPG included a component that greatly benefits from this change. Because the plug-in stops processing as soon as it finds a key it is looking for, a given look up does not detect or report subsequent errors in the configuration file. Previously, such subsequent errors could stop GT.M from using keys specified even in the correctly specified part of a configuration file. Note that this reference plug-in is compatible with V6.2-000 and V6.2-001.(GTM-8274) (GTM-8275)

* DBCERTIFY CERTIFY correctly handles a split in the root level of a global variable tree that occurs after a SCAN. This was previously reported as fixed in V5.2-000A as part of GTM-5873 (C9G04-002790), but had a regression in V6.0-001 due to fixes made for GTM-7559. Also, it no longer gives occasional spurious DBPREMATEOF or DBCINTEGERR errors. Previously these (fortunately rare) conditions required reverting to a V4 format database followed by repeating the scan and upgrade. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8287)

* The gtmprofile setup script correctly handles ICU versions ending in zero, such as 50, when defining gtm_icu_versions; previously gtmprofile inappropriately defined gtm_icu_version as just an integer instead of a decimal value. The workaround was to edit the script produced by gtmprofile. This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8315)

* DSE CHANGE FILEHEADER produces a CLIERR, Unrecognized option : SPIN_SLEEP_CNT error when given the long-deprecated SPIN_SLEEP_TIME field, which is also not displayed by the DSE DUMP FILEHEADER command. Previously DSE accepted this qualifier, which was harmless. (GTM-8316)

Error and Other Messages

ARCTLMAXHIGH

ARCTLMAXHIGH, The environment variable XXXX = YYYY is too high. Assuming the maximum acceptable value of ZZZZ

Run Time Warning: The environment variable named XXXX that controls the maximum number of auto-relink routine entries is assigned a value (YYYY) that is too high. MUMPS will set maximum routine count to ZZZZ and continue normal operation.

Action: Please set environment variable XXXX to a value between 1,000 and 16,000,000.

ARCTLMAXLOW

ARCTLMAXLOW, The environment variable XXXX = YYYY is too low. Assuming the minimum acceptable value of ZZZZ

Run Time Warning: The environment variable named XXXX that controls the maximum number of auto-relink routine entries is assigned a value (YYYY) that is too low. MUMPS will set maximum routine count to ZZZZ and continue normal operation.

Action: Please set environment variable XXXX to a value between 1,000 and 16,000,000.

INVLINKTMPDIR

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

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

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

INVTMPDIR

INVTMPDIR, Value for $gtm_tmpdir is either not found or not a directory: dddd - Reverting to default value

Error: Indicates the process cannot access directory dddd, which it may need for a number of actions; the directory may not exist as a directory or the process lacks authorization to locate the directory.

Action: The directory specification comes from $gtm_tmp if it is defined, otherwise it defaults to the system temporary directory, typically /var/tmp on Solaris and /tmp in other environments. Either correct the environment variable definition or ensure directory dddd is appropriately set up. Note that all users of a particular GT.M instance normally need to use the same temporary directory to ensure proper interprocess communication.

INVZBREAK

INVZBREAK, Cannot set ZBREAK in direct mode routine (GTM$DMOD)

Run Time Error: GTM$DMOD is an embedded routine that provides direct mode and it does not permit insertion of a ZBREAK.

Action: Issue ZBREAK only for application code

JOBLVNDETAIL

JOBLVNDETAIL, The zwrite representation of a local variable transferred to a JOB'd process is too long. The zwrite representation cannot exceed XXXX. Encountered size: YYYY

Run Time Error: The length of the zwrite representation of a local, (including the quotes, the '=', concatenate operator "_", and "$[Z]C()") has the length of YYYY which exceeds the maximum limit of XXXX.

Action: Please check the sizes of locals that needs to be sent and make sure their lengths are less than XXXX. For those big locals, consider using another mechanism such as sockets.

LASTWRITERBYPAS

LASTWRITERBYPAS, The last writer for database file xxxx bypassed the rundown

All GT.M Components Warning: This indicates that the last process which had the xxxx database file open for writing bypassed the rundown while disconnecting.

Action: This may occur due to an instance freeze. If so, first ensure that the Instance Freeze is resolved, manually clearing the Instance Freeze if necessary. If there is a source server still running, a normal shutdown of the source server will perform the rundown. Otherwise, the DSE ALL -B[UFFER_FLUSH] command may be used to ensure that any changes remaining in shared memory are written to disk.

NODFRALLOCSUPP

NODFRALLOCSUPP, The NODEFER_ALLOCATE qualifier is not allowed on this operating system. Not changing the defer allocation flag

MUPIP Error: Indicates the disk space preallocation is not supported on the current operating system.

Action: Consider using an external utility, such as FALLOCATE, to fulfill the need. Currently, Linux and AIX are the supported operating systems for the NODEFER_ALLOCATE feature.

NOPRINCIO

NOPRINCIO, NOPRINCIO Unable to write to principal device

Run Time Fatal: This indicates that GT.M attempted to but could not read from or write to the principal device.

Action: The NOPRINCIO error message works differently from other messages. The first occurrence results in an error that can be caught by device and trap handlers. The second occurrence is FATAL which does not drive device or trap handlers and terminates the process. The most common causes for the principal device to cease to exist involve terminal sessions or socket connections (including those from processes started by inetd/xinetd). When the remote client terminates the connection, the underlying principal device is closed and becomes inaccessible when the process attempts to READ from, or WRITE to, it. In the case of terminals, a typical cause is users closing the window of a terminal session without cleanly exiting from the GT.M process.

PREALLOCATEFAIL

PREALLOCATEFAIL, Disk space reservation for SSSS segment has failed

MUPIP/Run Time Error: Indicates the disk space preallocation has failed due to a system call error.

Action: Please read the accompanying system message to find out why the system call error occurred and resolve that problem.

RECLOAD

RECLOAD, Error loading record number: nnnn

MUPIP Error: This message identifies a record that MUPIP could not LOAD and follows a message about the cause. If this message is Fatal, which it can be for BIN format, it produces a core file for diagnostic analysis.

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

REPLLOGOPN

REPLLOGOPN, Replication subsystem could not open log file xxxx : yyyy. Logging done to zzzz

MUPIP Error: This indicates that MUPIP could not find the log file or did not have access permission to open the log file. If there is another log file available (a previously opened file), MUPIP writes to the other log file. If there is no other log file available, MUPIP sends any remaining messages to /dev/null and terminates the replication server process.

Action: Check the log file permissions, and if permissions are correct, move the log file and specify that MUPIP should log to a log file which has appropriate access permissions.

REPLSRCEXITERR

REPLSRCEXITERR, Source server for secondary instance xxxx exited abnormally. See log file yyyy for details.

Operator log Warning: This indicates the the source server for instance xxxx exited abnormally.

Action: Check the end of the log file at yyyy for additional message(s).

RESRCINTRLCKBYPAS

RESRCINTRLCKBYPAS, xxxx with PID qqqq bypassing the ssss semaphore for region rrrr (ffff) currently held by PID pppp.

Run Time Information: LKE or DSE automatically bypassed FTOK or access control semaphore. xxxx identifies the process type: "LKE", "DSE" or "GT.M"; qqqq is the bypassing process's PID; ssss identifies the semaphore type: "FTOK" or "access control"; rrrr is the region bypassed; ffff is the file corresponding to region rrrr; pppp is the PID of the process holding the semaphore. In normal operation these messages indicate incidental conflicts with no impact. If they occur when shutting the system downthey indicate the last process may have bypassed rundown actions that normally flush all information from memory to durable storage and release shared resources like semaphores and shared memory.

Action: These messages when shutting down GT.M activity may indicate a need to complete the process by invoking MUPIP RUNDOWN to get the database to a safe state; doing so as part of every shutdown is good practice.

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, zpos: hhhh

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. hhhh is the $ZPOSITION of the line of M code that caused the restart of the transaction; utilities leave this field blank.

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.

TRIGINVCHSET

TRIGINVCHSET, Trigger tttt for global gggg was created with CHSET=cccc which is different from the current $ZCHSET of this process

Trigger/Run Time Error: TRIGINVCHSET occurs when a process invokes a trigger on a global using a $ZCHSET that is different from the $ZCHSET used at the time of loading the first trigger on that global. GT.M implicitly uses the $ZCHSET of the first trigger on a global to invoke all triggers on that global. Note that tttt is the name of the first trigger on the global gggg-not necessarily the name of the trigger being invoked. cccc is the $ZCHSET of the process at the time of loading tttt on global gggg.

Action: Ensure that the process invoking a trigger on a global uses the same $ZCHSET that was used to load the first trigger on that global. If your application requires triggers in both M and UTF-8 modes, use different globals to load M mode and UTF-8 mode triggers.

TRIGUPBADLABEL

TRIGUPBADLABEL, Trigger upgrade cannot upgrade label NNNN to CCCC on ^GGGG in region RRRR

MUPIP Error: MUPIP TRIGGER UPGRADE cannot upgrade trigger version number NNNN to the current version number CCCC for the global GGGG in region RRRR.

Action: Reload the triggers in region RRRR.

WEIRDSYSTIME

WEIRDSYSTIME, Time reported by the system clock is outside the acceptable range. Please check and correct the system clock.

Run Time Error: Time reported by the system clock is outside the acceptable range. Please check and correct the system clock

Action: GT.M requires the system time be set between January 1, 1970 00:00:00 UTC (the UNIX epoch) and September 27, 33658 01:46:40 UTC