GT.M

Release Notes

V6.3-006

Empowering
the Financial World

Contact Information

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

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

Legal Notice

Revision History		
Revision 1.3	31 May 2019	Add the release note for GTM-9015.
Revision 1.2	16 April 2019	Updated release note for GTM-4263.
Revision 1.1	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 1.0	14 November 2018	V6.3-006

Table of Contents

V6.3-006

Overview

V6.3-006 provides a new Intrinsic Special Variable (ISV) that allows an application to schedule (and cancel) timed events. In addition, if your applications make significant use of M LOCKs, FIS recommends you adopt this release in place of V6.3-003[A], V6.3-004 or V6.3-005. It also provides improvements to the external call interfaces. The release brings numerous smaller enhancements, and fixes. See the Change History below. Please pay special attention to the items marked with the symbols ✓ or ✓

Note

Messages are not part of the GT.M API whose stability we strive to maintain. Make sure that you review any automated scripting that parses GT.M messages.

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 and GNU/Linux on x86 (32- and 64-bits).

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	-updok (recommended) -rootprimary (still accepted)

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
		configuration, A is the source instance for B and C.	
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). Over time, we'll update all GT.M documentation 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.

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 may 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 Group 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
IBM Power Systems AIX	7.1 TL 4, 7.2	Only 64-bit versions of AIX with POWER7 as the minimum required CPU architecture level 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. 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.** AIX 7.1 TL 5 is Supportable. Only the AIX jfs2 filesystem is Supported. Other filesystems, such as jfs1 are Supportable, but not Supported. FIS strongly recommends use of the jfs2 filesystem on AIX; use jfs1 only for existing databases not yet migrated to a jfs2 filesystem.
x86_64 GNU/Linux	Red Hat Enterprise Linux 7.3; Ubuntu 16.04 LTS	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). Due to build optimization and library incompatibilities, GT.M versions older than V6.2-000 are incompatible with glibc 2.24 and up. This incompatibility has not been reported by a customer, but was observed on internal test systems that use the latest Linux software distributions from Fedora (26), Debian (unstable), and Ubuntu (17.10). In internal testing, processes either hung or encountered a segmentation violation (SIG-11) during operation. Customers upgrading to Linux distributions that utilize glibc 2.24+ must upgrade their GT.M version at the same time as or before the OS upgrade. 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: * 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. Although GT.M itself does not require libelf, the geteuid program used by the GT.M installation script requires libelf (packaged as libelf1 on current Debian/ Ubuntu distributions and elfutils-libelf on RHEL 6 & 7).

Platform	Supported Versions	Notes
		Only the ext4 and xfs filesystems are Supported. Other filesystems are Supportable, but not Supported. Furthermore, if you use the NODEFER_ALLOCATE feature, FIS strongly recommends that you use xfs. If you must use NODEFER_ALLOCATE with ext4, you *must* ensure that your kernel includes 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. With NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs.
x86 GNU/Linux	Debian 9 (Stretch)	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. Running a 32-bit GT.M on a 64-bit GNU/Linux requires 32-bit libraries to be installed. The CPU must have an instruction set equivalent to 586 (Pentium) or better. Please also refer to the notes above on x86_64 GNU/Linux.

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.

32- vs. 64-bit platforms

The same application code runs on both 32-bit and 64-bit platforms; however there are operational differences between them (for example, auto-relink and the ability to use GT.M object code from shared libraries exist only on 64-bit platforms). Please note that:

* You must compile the application code separately for each platform. Even though the M source code is the same, the generated object modules are different - 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.

Parameter type	32-Bit	64-bit	Remarks
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.

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 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 V6.3-006 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.3-006_arch (for example, /usr/lib/fis-gtm/V6.3-006_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.3-006_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 appropriate MUPIP command (e.g. ROLLBACK, RECOVER, RUNDOWN) 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 or later. Also, modify your environment scripts to include the use of gtmcrypt_config environment variable.

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.

* If your application is not using object code shared using GT.M's auto-relink functionality, please consider using it.

Compiling the Reference Implementation Plugin

If you plan to use database encryption and TLS replication, you must compile the reference implementation plugin to match the shared library dependencies unique to your platform. The instructions for compiling the Reference Implementation plugin are as follows:

1. Install the development headers and libraries for libgcrypt, libgpgme, libconfig, and libssl. On Linux, the package names of development libraries usually have a suffix such as -dev or -devel 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
   ```

   Note that the package names may vary by distribution / version.

2. 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
   ```

3. Follow the instructions in the README.

   * Open Makefile with your editor; review and edit the common header (IFLAGS) and library paths (LIBFLAGS) in the Makefile to reflect those on your system.

* Define the gtm_dist environment variable to point to the absolute path for the directory where you have GT.M installed
* Copy and paste the commands from the README to compile and install the encryption plugin with the permissions defined at install time

Caution

There are separate steps to compile the encryption plugin for GT.M versions V5.3-004 through V6.3-000 when OpenSSL 1.1 is installed and OpenSSL 1.0.x libraries arestill available.

* Download the most recent OpenSSL 1.0.x version
* Compile and install (default installs to /usr/local/ssl)

```
./config && make install
```

* Adjust the configuration : Move the newly installed libraries out of the way

```
mv /usr/local/ssl/lib /usr/local/ssl/lib.donotuse
```

* Adjust the configuration : Create another /usr/local/ssl/lib and symlink the existing 1.0.x library into it as the default. This ensures that the encryption plugin is compiled using the compatible OpenSSL 1.0.x library. Adjust the path below as necessary.

```
mkdir /usr/local/ssl/lib && ln -s /path/to/existing/libssl.so.1.0.x /usr/local/ssl/libssl.so
```

* Recompile the encryption plugin following existing directions above
* Remove /usr/local/ssl to avoid future complications

Upgrading to GT.M V6.3-006

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 differs for 32-bit and 64-bit GT.M releases for the x86 GNU/Linux platform.

GT.M upgrade procedure for V6.3-006 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 V6.3-006 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 file before upgrading. There is no one-step method for downgrading a Global Directory file 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 V6.3-006.

* 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 the specified file.

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 V6.3-006.

* 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 V6*:

There is no explicit procedure to upgrade a V6 database file when upgrading to a newer V6 version. After upgrading the Global Directory, opening a V6 database with a newer V6 GT.M process automatically upgrades fields in the database fileheader.

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 V6.3-006 operations. However, that database can only grow to the maximum size of the version in which it was originally created. A database created on V5.0-000 through V5.3-003 has maximum size of 128Mi blocks. A database created on V5.4-000 through V5.5-000 has a maximum size of 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).

Important

In order to perform a database downgrade you must perform a MUPIP INTEG -NOONLINE. If the duration of the MUPIP INTEG exceeds the time allotted for an upgrade you should rely on a rolling upgrade scheme using replication.

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 syslog 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 and does not itself span regions; 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

V6.3-006 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.3-006 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 V6.3-006 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 V6.3-006 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 V6.3-006 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/V6.3-006.

To extract and reapply the trigger definitions on V6.3-006 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 V6.3-006, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.3-006 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 V6.3-006.

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 in 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.

To verify that your database meets all of the above requirements, execute MUPIP INTEG -NOONLINE. Note that the integrity check requires the use of -NOONLINE to ensure no concurrent updates invalidate the above requirements. Once assured that 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 V6.3-006 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

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 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 UTF-8 mode in the utf8 subdirectory, and one compiled without support for UTF-8 mode 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 UTF-8 mode support. Note that on 64-bit versions of GT.M, the object code is in shared libraries, rather than individual files in the directory.

* 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_V6.3-006_i686, then gtmprofile sets $gtm_dist to /usr/lib/fis-gtm/gtm_V6.3-006_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, refer to the Basic Operations chapter of GT.M Administration and Operations Guide.

Although GT.M uses ICU for UTF-8 operation, ICU is not FIS software and FIS does not support ICU.

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 as that provided by zlib.

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

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 (GNU/Linux) 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.

V6.3-006

Although GT.M uses a library such as zlib for compression, such libraries are not FIS software and FIS does not support any compression libraries.

Change History

V6.3-006

Fixes and enhancements specific to V6.3-006:

Id	Prior Id	Category	Summary
GTM-4263	C9C04-001965	Language	MUMPS commands accept prompted names in response to "What file: "
GTM-6135	S9I03-002673	Language	$ZTIMEOUT manages a process wide timed interrupt ✓
GTM-7952	-	Language	Improve memory and signal management for external calls ✓
GTM-8017	-	Other	^%TRIM accepts characters other than <SP> and <TAB> to trim ✓
GTM-8178	-	Language	✓ Normalize GT.M compiler invocations from ZCOMPILE, ZLINK, auto-ZLINK and the MUMPS command ✓
GTM-8518	-	Admin	✓ MUPIP REPLICATE -EDITINSTANCE requires standalone access and supports -CLEANSLOTS ✓
GTM-8933	-	DB	GT.M limits the number of errors from processes attempting to open a statsDB
GTM-8947	-	Language	Performance enhancement for $TRANSLATE() when arguments two and three are literals
GTM-8993	-	DB	A process that fails to open a statsDB does not establish the location it used ✓
GTM-8998	-	Language	External calls can return all available types ✓
GTM-9005	-	Admin	Approrpriate exit status from MUPIP LOAD
GTM-9009	-	DB	✓ LOCK Fixes ✓
GTM-9011	-	Admin	MUPIP SET accepts -KEY_SIZE or -RESERVED_BYTES in the same command as -RECORD_SIZE ✓
GTM-9015	-	DB	Unexpected JNLBADRECFMT and REPLJNLCLOSED errors
GTM-9017	-	Other	Prevent segmentation violation when invoking $gtm_procstuckexec
GTM-9024	-	DB	Improve LOWSPC reporting

Change History

Id	Prior Id	Category	Summary
GTM-9025	-	Other	Restore conversion performance in percent routines for smaller numbers
GTM-9031	-	Other	Update Cmake build scripts to be compatible with current Cmake releases
GTM-9038	-	Other	✓ GT.M excludes TLS 1.0/1.1 unless configured to use older SSL/TLS protocols

Database

* If misconfigured processes get the same error when opening a statsDB, GT.M throttles the messages such that every hundredth message goes to the operator log; previously every process reported the issue. (GTM-8933)

* A process which cannot open a statsDB disables itself from maintaining the shared statistics, but does not disable subsequently starting processes, which better enables changes to an incorrectly configured environment. Previously, an initializing process that could not access a statsDB could also effectively require all processes using that database to restart in order to enable statistic sharing for the region. (GTM-8993) ✓

* GT.M handles LOCK requests appropriately. Previously, under certain circumstances, LOCK requests could be granted to multiple processes simultaneously, or handle certain conditions inappropriately, leading to segmentation violations (SIG-11). This change requires additional memory per lock slot, so administrators should monitor lock slots (LKE SHOW) to determine whether their usage indicates a need to increase lock space.

  GT.M handles rare issues with LOCK structures by allocating an additional shared memory segment to contain them. GT.M removes this shared memory segment along with the primary database shared memory automatically.

  LKE supports the CLNUP command, which specifies that LKE process the lock space to remove any abandoned artifacts left by processes that exited without releasing their LOCKs. This processing also checks for evidence any entry has been misplaced by an "overflow" condition; it if finds any, it attempts to correct it, and, if it cannot, it produces a MLKHASHTABERR warning message. Such a message indicates the need to stop all access to (at least) the affected region, use MUPIP SET to set, and, if appropriate raise, the -LOCK_SPACE; because MUPIP SET -LOCK_SPACE is a standalone operation, using it, even with the current value, ensures the database is completely quiescent. Then resume operations. LKE CLNUP supports the -PERIODIC=n qualifier which specifies that LKE perform a cleanup every n seconds, which, if you desire active cleanup, is much lighter weight than repeated invocations of LKE from a shell script. FIS suggests running LKE CLNUP -PERIODIC=n with a value of n that appears to prevent growth in the elements in the lock space as reported by LKE SHOW over substantial periods of time. You stop When LKE CLNUP -PERIODIC with a MUPIP STOP <pid>. LKE CLNUP supports the -INTEG qualifier, which specifies that it validate the integrity of the lock space and reports any issues. These qualifiers are compatible with the -REGION or -ALL qualifiers.

  LKE SHOW -NOCRIT displays the PID of any process currently holding the LOCK critical section, and all invocations of LKE SHOW include utilization information, in the form of available/total space, about shared subscript data space related to LOCK commands. Previously, LKE SHOW reported detailed information about two out of three of the elements in the LOCK control structures, with the third element reporting "full" or "not full."(GTM-9009) ✓ ✓

* When the database reaches the 88% size threshold, and for every 1% increase in size and beyond, GT.M reports the blocks used in the LOWSPC warning as the sum of the data blocks and the local bit map blocks. Previously, GT.M attempted to report the total blocks used as just the data blocks to match the 'total' field outputted by MUPIP INTEG. Additionally, GT.M prints an accurate message about the percent usage when one of these threshold sizes is reached. (GTM-9024)

Language

* MUMPS commands without an argument accept an appropriate terminal response after the "What file: " prompt. The "What file: " prompt may be more appropriate to a mumps -run than for compilation only. When driven from a HEREDOC or script, the MUMPS command and MUPIP commands that can prompt for “File or Region:” require a specification on the same line as the command. Previously, GT.M appeared to ignore all terminal input at this MUMPS prompt and optionally allowed MUPIP to accept file or region on a separate line. (GTM-4263)

* The $ZTIMeout=([timeout][:labelref]) Intrinsic Special Variable (ISV) controls a single process wide timer. The optional timeout in seconds specifies with millisecond accuracy how long from the current time the timer interrupts the process. If the specified timeout is negative, GT.M cancels the timer. If the timeout is zero, GT.M treats it as it would a DO of the vector. The optional labelref specifies a code vector defining a fragment of M code to which GT.M transfers control as if with a DO when the timeout expires. If the timeout is missing, the assignment must start with a colon and only changes the vector, and in this case, if the vector is the empty string, GT.M removes any current vector. Note that GT.M only recognizes interrupts, such as those from $ZTIMEOUT at points where it can properly resume operation, for example, at the beginning of a line, when waiting on a command with a timeout, or when starting a FOR iteration. When a ztimeout occurs, if the last assignment specified no vector, GT.M uses the current $ETRAP or $ZTRAP with a status warning of ZTIMEOUT. GT.M rejects an attempted KILL of $ZTIMeout with an error of %GTM-E-VAREXPECTED, and an attempted NEW of $ZTIMeout with an error of %GTM-E-SVNONEW.

  Example:

```
GTM>zprint ^ztimeout
ztimeout
  ; Display $ztimeout
    write !,$ztimeout               ; display $ZTIMeout - in this case the initial value -1
  ; set with a vector (do ^TIMEOUT)
    set $ztimeout="60:do ^TIMEOUT"  ; timeout of 1 minute. After timeout expires, XECUTEs do ^TIMEOUT
    write !,$ztimeout               ; displays the remaining time:vector until timeout
  ; set without a vector
    set $ztimeout=120               ; set the timeout to 2 minutes without changing the vector
    set $ztimeout="1234do ^TIMEOUT" ; missing colon creates a timeout for 1234 seconds
    set $ztimeout="10:"             ; set the timeout to 10 seconds and vector to current etrap or ztrap
    set $ztimeout=-1                ; set cancels the timeout
  ; Note that set to 0 triggers an immediate timeout
    set $ztimeout=0                 ; triggers the current vector
    set $ztimeout="0:DO FOO"        ; this has the same effect as DO FOO

GTM>
```

  (GTM-6135) ✓

* Name-level $ORDER(,-1) and $ZPREVIOUS() return an empty string when they reach the trigger definitions (stored in ^#t) as it is not a normally accessible global. Since the introduction of triggers in V5.4-000 if there were trigger definitions, these functions could return ^#t. (GTM-7433)

* GT.M protects buffers used for external calls and produces an EXTCALLBOUNDS error if the external call attempts to exceed the space requested by the call table definition. Previously GT.M did not provide this protection and used a less efficient strategy for managing the space. Additionally, when an external call exceeds its specified preallocation (gtm_string_t * or gtm_char_t * output), GT.M produces an EXCEEDSPREALLOC error. Previously GT.M did not immediately detect this condition, which could cause subsequent hard to diagnose failures.

  GT.M supports call-specific options in external call tables by appending a colon to the end of the line followed by zero or more space separated, case-insensitive keywords. The SIGSAFE keyword attests that the specific call does not create its own signal handlers, which

allows GT.M to avoid burdensome signal handler coordination for the external call. Previously, and by default, GT.M saves and restores signal setups for external calls.
(GTM-7952) ✓

* When the name of a source file is not a valid routine name, GT.M issues a NOTMNAME error and does not produce an object file. Previously, GT.M eventually gave an error when attempting use the routine. The -OBJECT compilation qualifier used without the -NAMEOFRTN qualifier implicitly names the first routine to match the name of the object qualifier. Note that, as before, listing files take on the name of the source rather than the name of the routine, and the -NAMEOFRTN or -OBJECT qualifiers in $ZCOMPILE are problematic to use with ZLINK commands as they apply to every ZLINK argument. Previously the qualifier applied to all files specified by the same MUMPS or ZCOMPILE command such that all sources received the same object name, which meant the last file was the only one that endured. Also, ZCOMPILE, as documented, accepts qualifiers in its argument prior to the routine list; previously it did not. ZCOMPILE with a wildcard works reliably; previously it stopped compiling routines after encountering a large source file In addition, the MUMPS and ZCOMPILE commands default file specifications without a .m file extension to have one and they only compile files with a .m extension; previously they did not, although some other facilities did require a .m file extension for source files. As before, explicit ZLINK of a source (.m) file always places the object in the same directory as the specified source. While we are not aware of customers with a practice of using non .m extensions or module names that are not valid M names at compilation, but subsequently rename the object modules, this change requires revision of such practices. (GTM-8178) ✓ ✓

* When the second and third argument of $TRANSLATE are literals, the GT.M compiler calculates the tables used by the translation. Previously thetables were always prepared at run-time. (GTM-8947)

* GT.M can return any of the types documented in the external calls API. For C external calls, to prevent memory leaks when returning any pointer types, GT.M requires the application to allocate returns of these types using gtm_malloc. Note that using the standard malloc for these types (or not explicitly allocating at all) produces a GTM-F-ASSERT. For Java external calls, the plugin manages any necessary allocations. Previously, GT.M had the ability to return any of the documented types, however the GT.M Programmer's Guide did not document the ability to return anything other than an integer status or a long (for Java calls). Additionally, even though GT.M had the ability to return a range of types, it did not perform the appropriate check for available space in the string pool when returning gtm_char_t*, gtm_char_t**, or gtm_string_t*, which could lead to unpleasant symptoms including a segmentation violation (SIG-11) or incorrect results. Also, attempting to return null values from a C external call results in a %GTM-E-XCRETNULLREF error, and attempting to return null from a Java external call results in a %GTM-E-JNI error. (GTM-8998) ✓

* Address a regression in V6.3-005 caused by GTM-8989 which in certain circumstance results in an out-of-design situation that turns off journaling and/or results in corrupted journal entries. While this failure was first noticed in development as a rare debug-only edge case, it has since been identified as repeatable under specific workloads. (GTM-9015)

System Administration

* MUPIP REPLICATE -EDITINSTANCE supports a -CLEANSLOTS qualifier. When specified, MUPIP goes through all slots (currently 16) in the replication instance file, identifies the slots that are inactive, and clears them to make them available for reuse. Also, except in the case where an originating primary instance has crashed, MUPIP REPLICATE grabs a "standalone" resource lock for processing -EDITINSTANCE. Previously, MUPIP did not use a resource lock when acting on an -EDITINSTANCE. (GTM-8518) ✓ ✓

* MUPIP LOAD returns non-zero exit status for load errors. Previously in some cases it inappropriately returned a 0 (Zero) exit status when it had been unable to load one or more records. (GTM-9005)

* MUPIP SET accepts -KEY_SIZE or -RESERVED_BYTES and -RECORD_SIZE in the same command; beginning with V6.0-000 they are not incompatible, but MUPIP SET continued to give an error when they were combined. (GTM-9011) ✓

Other

* The ^%TRIM() utility allows the specification of what character(s) to trim from either the left and/or right hand side of a given string. The default trim characters are $CHAR(32,9) (<SP> and <TAB>), these can be overridden by passing a string consisting of the desired characters in the optional second parameter.This functionality has existed for some time but was undocumented and not regularly tested. (GTM-8017) ✓

* GT.M appropriately invokes $gtm_procstuckexec when it encounters a situation, such as BUFOWNERSTUCK, when another process abnormally holds some resource for too long. Previously, under a rare sequence of events the invocation of gtm_procstuckexec could result in a segmentation violation (SIG-11). (GTM-9017)

* The following utilities: %DH, %DO, %HD %OD and %UTF2HEX have optimizations for the sizes most likely to be used in the GT.M environment. The In V6.3-005, GTM-5574 extended the maximum size supported by the conversion utilities but that caused a performance reduction for the most common cases. (GTM-9025)

* The GT.M Cmake build scripts now work with Cmake v3; previously they used a feature (the debug property) deprecated in that Cmake release. (GTM-9031)

* GT.M TLS encrypted sockets disallow TLS 1.0 and TLS 1.1 protocols. Previously,GT.M disallowed only SSLv2 and SSLv3 protocols. If you need to selectively re-enable these protocols, please refer to the ssl_optionsconfiguration option in "Appendix H: Creating a TLS Configuration File" (GTM-9038) ✓

ERRWZTIMEOUT

ERRWZTIMEOUT, Error while processing $ZTIMEOUT

Run Time Error: This indicates a problem invoking the current $ZTIMEOUT vector and usually accompanies other error messages

Action: Examine and correct the code vector specified by $ZTIMEOUT, or if there is none, examine the current value for $ETRAP or $ZTRAP. Unlike $ETRAP and code values for $ZTRAP, which are evaluated when they are assigned, compilation of $ZTIMEOUT vectors occurs when the vector is invoked by the expiration of the specified time.

EXCEEDSPREALLOC

EXCEEDSPREALLOC, Preallocated size ssss for M external call label LLLL exceeded by string of length SSSS

Call out Error: The code invoked as externroutinename LLLL returned a string of length SSSS, but the call table specified a maximum length of ssss for the return.

Action: Revise the external routine to abide by the call table size or change the call table to preallocate a suitably larger size.

EXTCALLBOUNDS

EXTCALLBOUNDS, Wrote outside bounds of external call buffer. M label: LLLL

Call out Fatal: The code invoked as externroutinename LLLL violated the bounds of its allocated buffers.

Action: Ensure the non-GT.M code uses appropriate allocations, pointer management logic and bounds checking.

FILEEXISTS

FILEEXISTS, File xxxx already exists

MUPIP Error: This indicates that MUPIP discovered a file with the filename xxxx already existing and did not overwrite it while executing the specified command(s). In many cases, this is an expected outcome when the action has an explicit or implicit target of multiple database files which may be in differing states.

Action: If appropriate, rename the already existing file xxxx and reissue the MUPIP command(s), or modify the MUPIP command to name (explicitly/implicitly) a file different from xxxx. If you encountered this error with MUPIP BACKUP, use the -REPLACE qualifier if you want to replace the existing backup files.

FILERENAME

FILERENAME, File xxxx is renamed to yyyy

Run Time Information: This indicates that an existing file xxxx has been renamed to yyyy so that a new file created with the original name does not overwrite the existing one. GT.M renames files during an automatic journal switch in case no explicit journal file name is specified, in which case the message is sent to the operator log. The utilities (MUPIP, GT.CM) rename files while opening log files or journal extract files and they send the message to the terminal. GT.M or utilities rename files only if the new file name specified already exists.

Action: This information messages confirms the success of the file rename operation. No futher action is necessary unless there are other warning, fatal, and/or error category messages.

GTMSECSHRPERM∆

GTMSECSHRPERM, The gtmsecshr module in $gtm_dist (DDDD) does not have the correct permission and uid (permission: PPPP, and UID: UUUU)

Run Time Error: This indicates that a client did not start the GTMSECSHR, installed to DDDD, because the executable was not owned by root (UUUU is the actual owner) and/or did not have setuid and/or exceute permissions (actual permissions are PPPP).

Action: Arrange to provide the GTMSECSHR executable with the proper characteristics. The executable must be SETUID root with execute permissions for the current user.

JNLCREATE∆

JNLCREATE, Journal file xxxx created for <database/region> yyyy with aaaa

MUPIP Information: This indicates that a journal file xxxx is created for database/region yyyy with NOBEFORE_IMAGES or BEFORE_IMAGES journaling option (aaaa).

Action: This informational message confirms the success of the new journal file creation operation for a region. No futher action is necessary unless there are other warning, fatal, and/or error category messages.

JNLFILOPN

JNLFILOPN, Error opening journal file jjjj for database file dddd

Run Time/MUPIP Error: This indicates that GT.M was unable to open journal file jjjj for the specified database file dddd. The Source Server exits with a JNLFILOPN message after six failed attempts to open journal files.

Action: Check the authorizations for the user of the process and the health of the file system holding the journal file.

JNLSPACELOW∆

JNLSPACELOW, Journal file jjjj nearing maximum size, nnnn blocks to go

Run Time Information: Depending on your settings for ALLOCATION, AUTOSWITCHLIMIT, and EXTENSION journaling options, you may see one to three JNLSPACELOW messages for each generation of a journal file. When the difference between AUTOSWITCHLIMIT and ALLOCATION is an exact multiple of EXTENSION, GT.M attempts to write the JNLSPACELOW message to the operator log three times as a journal file reaches its maximum size. The first JNLSPACELOW message appears in the operator log when the available free space (blocks) in a journal file is equal to twice the EXTENSION, the second appears when the available free space is equal to EXTENSION, and the third appears when the journal file reaches the maximum size (AUTOSWITCHLIMIT). With EXTENSION=0 or EXTENSION=AUTOSWITCHLIMIT, GT.M logs the JNLSPACELOW message only once per journal file to the operator log.

Action: The JNLSPACELOW message is an information message and requires no action. However, you can use the JNLSPACELOW messages as part of monitoring journaling space requirements or as an operational practice to a trigger to intervene in journal file management. Use the frequency of JNLSPACELOW messages to proactively monitor how fast a journal file grows and as part of a monitoring alorithm that helps predict how soon the disk is likely to hit a quota limit.

JNLSTATE∆

JNLSTATE, Journaling state for <database/region> xxxx is now yyyy

MUPIP Information: This indicates that journal state for the database/region xxxx is now yyyy.

Action: This information message confirms the sucess of journal state change operation. No futher action is necessary unless there are other warning, fatal, and/or error category messages.

JNLSWITCHSZCHG

JNLSWITCHSZCHG, Journal AUTOSWITCHLIMIT [aaaa blocks] is rounded down to [bbbb blocks] to equal the sum of journal ALLOCATION [cccc blocks] and a multiple of journal EXTENSION [dddd blocks]

MUPIP Information: This indicates that the specified AUTOSWITCHLIMIT value was rounded down as little as possible to make it aligned to the ALLOCATION + a multiple of EXTENSION. Any subsequently created journal file will use this value for AUTOSWITCHLIMIT.

Action: If the automatically rounded value for AUTOSWITCHLIMIT is inappropriate, specify an appropriate value for ALIGNSIZE, ALLOCATION, and/or EXTENSION .

JNLSWITCHTOOSM

JNLSWITCHTOOSM, Journal AUTOSWITCHLIMIT [aaaa blocks] is less than journal ALLOCATION [bbbb blocks] for database file dddd

MUPIP Error: This indicates that the specified value or the automatically calculated value for AUTOSWITCHLIMIT from a MUPIP SET JOURNAL command is less than the default or specified value of ALLOCATION. This error also indicates that the AUTOSWITCHLIMIT value specified was greater or equal to the ALLOCATION but in turn got rounded down, and this rounded down value is less than the ALLOCATION.

Action: Specify a higher value of AUTOSWITCHLIMIT or specify an ALLOCATION value that is less than the AUTOSWITCHLIMIT.

LOCKCRITOWNER

LOCKCRITOWNER, LOCK crit is held by: PPPP

Run Time/LKE Information: This shows any current owner of the resource managing M LOCKs.

Action: If a process persists in this state investigate what it's doing and, if appropriate, consider terminating it.

LOCKSPACEINFO

LOCKSPACEINFO, Region: rrrr: processes on queue: pppp/qqqq; LOCK slots in use: llll/kkkk; SUBSCRIPT slot bytes in use: ssss/tttt

Run Time Error: This indicates that the environment attempted more concurrent M LOCKs than the configured LOCK_SPACE for region rrrr can support. pppp processes are waiting on a lock. llll locks are in use. qqqq and kkkk indicate maximum number of process queue entries, and maximum number of locks respectively.

Action: Analyze the LOCK protocol for efficiency. Use mupip set -region -lock_space=size "rrrr" to increase the lock space for region rrrr. To avoid the same problem the next time you recreate the database, use GDE to make the analogous change to lock_space for the segment mapped to the ffff file in the global directory used to MUPIP CREATE this region.

LOWSPC

LOWSPC, WARNING: Database DDDD has less than PPPP% of the total block space remaining. Blocks Used: UUUU Total Blocks Available: AAAA

Operator log Information: The database has UUUU block in use and is appoaching its current limit of AAAA blocks. When the database reaches the 88% size threshold, and for every 1% increase in size and beyond, GT.M reports the blocks used in the LOWSPC warning as the sum of the data blocks and the local bit map blocks.

Action: Purge data if possible. Consider a MUPIP REORG to compact the remaining data. Investigate whether migrating to a database created by a current version has a higher limit. Move some data to another, possibly new, region and delete it from this one.

MLKCLEANED

MLKCLEANED, LOCK garbage collection freed aaaa lock slots for region rrrr

LKE Information: LKE CLNUP was able to free lock slots when requested.

Action: No action required.

MLKHASHRESIZE

MLKHASHRESIZE, LOCK hash table increased in size from aaaa to bbbb and placed in shared memory (id = mmmm)

Operator log Information: GT.M needed to expand a hash table used for managing LOCK information.

Action: No user action is required, but shared memory monitoring will show an additional shared memory segment with id mmmm.

MLKHASHRESIZEFAIL

MLKHASHRESIZEFAIL, Failed to increase LOCK hash table size from aaaa to bbbb. Will retry with larger size.

Operator log Warning: GT.M needed to expand a hash table used for managing LOCK information needed to be expanded, but the initial attempt failed, necessitating a retry.

Action: A subsequent MLKHASHRESIZE indicates that the retry succeeded and no user action is required.

MLKHASHTABERR

MLKHASHTABERR, A LOCK control structure is damaged and could not be corrected. Lock entry for LLLL is invalid.

LKE Error: LKE CLNUP -INTEG encountered an out-of-design situation for LOCK LLLL and was unable to repair it automatically.

Action: Immediately report the entire incident context with information from the operator log and any other relevant information to your GT.M support channel.

MLKHASHWRONG

MLKHASHWRONG, A LOCK control structure has an invalid state; LOCK table failed integrity check. TTTT

LKE Error: MLK CLNUP -INTEG encountered damage to the data structures related to LOCK management. The text in TTTT describes whether LKE was able to correct the error or not.

Action: If LKE was not able to correct the error, immediately report the entire incident context with information from the operator log and any other relevant information to your GT.M support channel as soon as possible.

MUNOACTION

MUNOACTION, MUPIP unable to perform requested action

MUPIP Error: This indicates that MUPIP encountered an error, which prevented the requested action.

Action: Review the accompanying message(s) to identify the cause that prevented MUPIP from performing the requested operation.

NOTMNAME

NOTMNAME, XXXX is not a valid M name

Compile Time Error: M names must be ASCII, start with a "%" or an alpha and thereafter contain only alphanumeric characters. In GT.M M, names are currently functionally limited to 31 characters, in most cases, by truncation.

Action: Correct the, typically routine, name to comply with the supported format. Names are also used for labels and both global and local variables. Note that GT.M usually truncates names longer than its supported maximum, which, FIS recommends against as, while it can provide embedded information, can lead to ambiguity or other unintended behavior.

RESRCWAIT

RESRCWAIT, Waiting briefly for the tttt semaphore for region rrrr (ffff) was held by PID pppp (Sem. ID: ssss)

Run Time Information: A process started a three (3) second wait for an FTOK or access control semaphore. If process with PID pppp does not release the semaphore before the timeout expires, the waiting process bypasses acquiring the semaphore. tttt identifies the semaphore type: "FTOK" or "access control"; rrrr is the region; ffff is the database file corresponding to region rrrr; ssss is the semaphore ID.

Action: None required.

XCRETNULLREF

XCRETNULLREF, Returned null reference from external call LLLL

Call out Error: The code invoked as externroutinename LLLL returned a NULL pointer. While GT.M accepts returns of a zero (0) value or an emptry string, it does not support the return of a NULL pointer.

Action: Revise the external call code to return a pointer to an appropriate value.

ZTIMEOUT

ZTIMEOUT, ZTIMEOUT Time expired

Run Time Warning: This warning message appears when $ZTIMEOUT expires and there were no vectors defined. If no error handlers are defined, GT.M invokes the default trap which puts the control to Direct Mode .

Action: Check the message(s) for more information on where the timer expired in the current process. If needed, set an appropriate error handler to specify an action associated with $ZTIMEOUT expiry or define a $ZTIMEOUT with a vector.