# GT.M

## Release Notes

V7.1-000

Empowering
the Financial World

FIS

## Contact Information

GT.M Group
Fidelity National Information Services, Inc.
347 Riverside Drive
Jacksonville, FL 13220
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

### Legal Notice



| Revision History | | |
|---|---|---|
| Revision 1.0 | 14 April 2023 | V7.1-000 |

# Table of Contents

# GT.M Release Notes

## V7.1-000

## Overview

V7.1-000 adds MUPIP UPGRADE and MUPIP REORG -UPGRADE to upgrade V6 format database files in place to V7 format as well as various fixes. For more information, refer to the Change History section.

Items marked with the ✅ symbol document new or different capabilities.

Please pay special attention to the items marked with the ⚠ symbol. as those document items that have a possible impact on existing code, practice or process. Please be sure to recompile all objects to ensure all the updates are in place.

### Note

While FIS keeps message IDs and mnemonics quite stable, messages texts change more frequently as we strive to improve them, especially in response to user feedback. Please ensure 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:<br>mupip backup -database ACN,HIST /backup |
| **Reference Number** | A reference number is used to track software enhancements and support requests.<br>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 x86_64.

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. | -updok (recommended) |

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

Effective V6.0-000, GT.M documentation 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, we refer to the combination of operating system and hardware architecture as a platform. We deem this set of specific versions: 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. We deem this larger set of versions: 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 and therefore deem them: 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 reference plugin has its own additional requirements.

| Platform | Supported Versions | Notes |
|---|---|---|
| IBM Power Systems AIX | 7.1 TL 5, 7.2 TL 5 | Only 64-bit versions of AIX with POWER7 as the minimum required CPU architecture level are Supported.<br><br>While GT.M supports both UTF-8 mode and M mode on this platform, there are problems with the AIX ICU utilities that prevent FIS from testing 4-byte UTF-8 characters as comprehensively on this platform as we do on others.<br><br>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.**<br><br>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 7.9, 8.7; Ubuntu 18.04 LTS, 20.04 LTS, and 22.04 LTS; Amazon Linux 2 | To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware.<br><br>GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6.32 or later), glibc (version 2.12 or later) and ncurses (version 5.7 or later).<br><br>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.<br><br>GT.M requires a compatible version of the libtinfo library. On Red Hat, the ncurses-libs and ncurses-compat-libs packages contain the libtinfo library. On Debian/Ubuntu, libtinfo5 and libncurses5 packages contain the libtinfo library. If any of these packages is not already installed on your system, please install using an appropriate package manager.<br><br>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 or later.<br><br>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 |

| Platform | Supported Versions | Notes |
|---|---|---|
| | | the bug is 1213487. With NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs.<br><br>Our testing has shown an interaction between glibc 2.36 and all versions of GT.M on Linux/x86_64 systems without AVX2 support. This can cause segmentation violations (SIG-11) in processes performing concurrent updates to the same database block, which terminate the process, but do not damage the database. The issue is due to the way glibc performs certain memory operations when using SSE2 instructions. The glibc behavior was subsequently modified to avoid this issue, and the change was included in glibc 2.37, however, we have not yet confirmed the change resolved issue. Linux/x86_64 systems with support for AVX2 instructions are not vulnerable, as glibc chooses its AVX2 implementation, when available, over its SSE2 implementation, and the problematic behavior is specific to SSE2. Note, depending on how CPU virtualization is configured, that virtual environments may not support AVX2 even if the underlying hardware does.<br><br>Ubuntu 21.10 and Red Hat Enterprise Linux 9 are Supportable.<br><br>  <br>**Note**<br>• To use TLSv1.3 with OpenSSL 1.1.1 and up, you must recompile the reference encryption plugins<br>• RHEL 8 includes compat-openssl10.x86_64 for binaries compiled against OpenSSL 1.0.2 on RHEL 7 |

**Important**

Effective V7.0-003, GT.M is no longer Supportable on the 32 bit x86 platform. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.

### 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 attempts to provide support for customers in good standing on 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.

## 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 V7.1-000 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V7.1-000_arch (for example, /usr/lib/fis-gtm/V7.1-000_x86_64 on Linux systems). A location such as /opt/fis-gtm/V7.1-000_arch would also be appropriate.
- 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 the example / reference implementation plugin in support of database encryption, TLS replication, or TLS sockets, you must compile the reference plugin in order to match the shared library dependencies specific 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

The encryption plugin currently uses functionality that is deprecated in OpenSSL 3.0. This will be fixed in a future release.

### Re-evaluate TLS configuration options

The GT.M TLS reference encryption plugin implements a subset of options as documented in the OpenSSL 1.0.2 man page for SSL_set_options which modify the default behavior of OpenSSL. Future versions of the plugin will enable new options as and when the OpenSSL library adds them. To enable options not supported by the GT.M TLS reference plugin, it is possible to create an OpenSSL configuration for GT.M processes. See the OpenSSL man page for "config".

## Upgrading to V7.1-000

### Before you begin

GT.M supports upgrade from V5*, V6.* and V7.* versions to V7.1-000.

GT.M does not support upgrading from V4* versions. Please upgrade V4 databases using instruction in the release notes of an appropriate GT.M V6.* version.

The GT.M database consists of four types of components- database files, journal files, global directories, and replication instance files.

GT.M upgrade procedure for V7.1-000 consists of 5 stages:

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

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V7.1-000 depends on your GT.M upgrade history and your current version.

### Stage 1: Global Directory Upgrade

FIS strongly recommends you back up your Global Directory 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 V7.1-000.

- Execute the EXIT command. This command automatically upgrades the Global Directory.
- 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 V7.1-000.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

An analogous procedure applies in the reverse direction.

## Stage 2: Database Files Upgrade

Before starting the database file upgrade, use the prior GT.M version to perform an appropriate MUPIP action (i.e. ROLLBACK, RECOVER, RUNDOWN) to removes abandoned GT.M database semaphores and releases any IPC resources.

There are three upgrade paths available when you upgrade to V7.1-000.

**V7 Upgrade Path 1: In-place Upgrade**

To upgrade from GT.M V7*:

There is no explicit procedure to upgrade a V7 database file when upgrading to a newer V7 version. After upgrading the Global Directory, opening a V7 database with a newer V7 GT.M process automatically upgrades the fields in the database file header.

To upgrade from GT.M V6* (or V5*):

There are two phases to upgrade from V6 to V7:

- Phase 1: MUPIP UPGRADE phase
- Phase 2: MUPIP REORG -UPGRADE (GVT Index Block Upgrade)

Both phases operate once per region and require standalone access. Phase 1 is not restartable. Phase 2 is restartable.

While these are the basic steps, customers must integrate them with appropriate operational practice and risk mitigating procedures, such as comprehensive testing, backup, integrity checks, journal and replication management, and so on based on their environments and risk tolerance. FIS strongly recommends performing a MUPIP INTEG (-FAST), of the database and creating a backup prior to upgrade. Customers must test these utilities against copies of their own production files, using their planned procedures, before undertaking the conversion of current production files.

While FIS has done considerable testing of MUPIP UPGRADE and MUPIP REORG -UPGRADE, the duration of that testing has not reached the level FIS typically performs for work of this complexity and impact. While our goal is to allow MUPIP REORG -UPGRADE to run with concurrent activity, our testing has not reached a level to allow it to run without standalone access. Using MUPIP UPGRADE and MUPIP REORG -UPGRADE should be a significantly faster alternative to using MUPIP EXTRACT and LOAD. FIS favors using a "rolling" upgrade using a replicated instance. Whatever the approach you choose, FIS requests capturing all logs in case there are issues or questions leading to support requests.

**Phase 1: Standalone MUPIP UPGRADE**

MUPIP UPGRADE performs Phase 1 actions of upgrading a database to V7. The format of the UPGRADE command is:

```
MUPIP UPGRADE {-FILE <file name>; | [-REGION] <region list>}
```

As the GT.M version upgrade changes the journal format to support 64-bit block pointers, MUPIP UPGRADE does not maintain journal files or replication; configured journaling and replication resumes for activity after MUPIP UPGRADE.

UPGRADE:

- Requires standalone access
- Turns off journaling and replication for the duration of UPGRADE
- When encountering an error where the command specifies multiple regions, UPGRADE moves on to the next region, while for a single file/region, it terminates; avoid any unnecessary <CTRL_C> or MUPIP STOP (or kill) of an active MUPIP UPGRADE process, as such an action leaves the database region effectively unusable
- Estimates and reports the space required for its work
  - UPGRADE estimates are intended to be generous, and, particularly for small databases, they may seem unnecessarily large
  - If MUPIP is not authorized to perform a required file extension, that is, the extension amount is defined as zero (0), it produces an error before it does anything that would damage the selected database file
- Moves blocks from immediately after the existing master map to make room for a V7 master map
  - Depending on the block size and the GT.M version with which it was created, the new starting Virtual Block Number (VBN), the location of block zero for the database file, may exceed the starting VBN for a database created with V7, which causes a minor waste of space
  - UPGRADE relocates blocks in multiples of 512 to align blocks with their local bitmaps
- Eliminates any globals that previously existed, but have been KILL'd at the name level; these global variable trees (GVTs) contain only a level one (1) root block and an empty data (level zero) block and are "invisible" to the GT.M process run-time
- Stores the offset GT.M must apply to the original block pointers as a consequence of the relocation of the starting VBN
- Upgrades the directory tree (DT) block pointers from 32- to 64-bits; this requires splitting any blocks that do not have sufficient space to accommodate the larger block pointers
- Ensures that all is work is flushed to secondary storage
- Reports completion of its activity on a database file with a "MUPIP MASTERMAP UPGRADE completed" message

At this point, after a successful MUPIP UPGRADE:

- All DT blocks are in V7m format and all GVT index blocks remain in V6/V6p format
- Subsequent activity that updates index blocks for existing GVTs implicitly converts any V6 index blocks to V6p format after applying the offset
- No process other than MUPIP REORG -UPGRADE converts GVT index blocks from V6p format to V7m format; in other words, adding new nodes does not create GVT index blocks with V7 format - adding new nodes splits existing index blocks and such block splits retain the pre-existing block format
- Newly created GVTs, storing new global names, have V7m format
- Data blocks, at level zero (0), and local bit map blocks have the same format in V6 and V7, so, for consistency, normal updates also give those blocks a V7m format designation

These database changes are physical rather than logical, and thus do not require replication beyond noting the increase in transaction numbers.

**Phase 2: MUPIP REORG -UPGRADE** (GVT Index Block Upgrade)

MUPIP REORG -UPGRADE performs Phase 2 actions of upgrading a database to V7 format. The format of MUPIP REORG -UPGRADE is:

```
MUPIP REORG -UPGRADE {-FILE <file_name> | [-REGION] <region_list>}
```

Before image journaling with MUPIP REORG upgrade provides maximum resiliency. MUPIP REORG -UPGRADE reports it has completed its actions for a region with a MUPGRDSUCC message, at which point all index blocks have V7m format with 64-bit block pointers. You can resume and complete a MUPIP REORG -UPGRADE stopped with a MUPIP STOP (or <Ctrl-C>); avoid a kill -9, which carries a high risk of database damage.

MUPIP REORG -UPGRADE:

- Requires standalone access
- Runs on an entire region; as a result, MUPIP REORG -UPGRADE prevents multiple concurrent REORG -UPGRADE runs per region
- Stops execution when a concurrent Online ROLLBACK is detected because that operation changes the block content of the database
- Can be subject to stopping and restarting at any point
- Processes the GVTs within a database file
  - Splitting any index blocks that do not have sufficient space to accommodate the block pointer upgrade from 32 to 64 bits
  - Updating the block pointers from 32 to 64 bits, also changing the version of the block to V7m
  - Journaling its work as before images (if so configured) and INCTN records

**Phase 3: Optional GVT Data and Local Bit Map Block Upgrade**

While it makes no operational or processing difference, GT.M does not consider the database "fully upgraded" until the block version format of all data blocks becomes V7m. Any of the following operations upgrade some or all of the remaining data blocks:

- MUPIP REORG

  Because this operation may not visit every block in the database it may fail to upgrade static/unchanging blocks
- MUPIP REORG -ENCRYPT
- MUPIP INTEG -TN_RESET

  This operation requires standalone access and resets the transaction number on all blocks in the database.

Failure to perform Phase 3 has **NO** implications for V7.1-000 but might be an issue for any as-yet unplanned further enhancement.

**V7 Upgrade Path 2: EXTRACT and LOAD**

Two commonly used mechanisms are as follows. We recommend you use replication to stage the conversion and minimize down time.

- MUPIP EXTRACT -FREEZE followed by a MUPIP LOAD

  Using MUPIP EXTRACT with -FREEZE ensures that the V6 database files are frozen at the point of the extract, preventing updates without administrative action to unfreeze the database. MUPIP LOAD the extracts into newly created V7 database files

  Use this operation when there is insufficient space to make a database extract
- MERGE command with two global directories and Extended References

  Using this approach to transfer data from a V6 database file to a V7 database, administrators must take some action to prevent updates during the transfer

  This operation consumes less disk space and disk I/O. As a result the operation is faster than an EXTRACT and LOAD.

If you are using triggers, extract the triggers from the V6 database and load them in the new V7 database.

**V7 Upgrade Path 3: No change**

Continue using your V6 databases with GT.M V7.1-000. In case you do not wish to operate with files of differing format, specify the -V6 qualifier when invoking MUPIP CREATE.

**Choosing the right upgrade path**

Choose V7 Upgrade Path 1 or 2 if you anticipate a database file to grow to over 994Mi blocks or require trees of over 7 levels as V7.1-000 supports 16Gi blocks and 11 levels. Note that the maximum size of a V7 database file having 8KiB block size is 114TiB (8KiB*16Gi).

Choose the V7 Upgrade Path 3 if you do not anticipate a database file to grow beyond the V6 database limit of 994Mi blocks or a tree depth limit of 7 levels. Note that the maximum size of a V6 database file having 8KiB block size is 7TiB (8KiB*992Mi).

Other than the new maximum database file size and greater tree depth that comes with V7 Upgrade Path 1 and 2, there is no difference between V7 Upgrade Path 1 and 2 and V7 Upgrade Path 3. You can choose V7 Upgrade Path 3 first and then later choose V7 Upgrade Path 1 or 2 if a need arises.

For additional details on differences in factors involved in the V6 to V7 upgrade refer to Appendix G in the GT.M Administration and Operations Guide.

## 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 the V5 version of 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 through V6.3-014 have a maximum size of 1,040,187,392 (992Mi) blocks.
- Database created with V7.0-000 and up have a maximum size of 17,179,869,184 (16Gi) blocks.

## Stage 3: Replication Instance File Upgrade

GT.M V7.1-000 does not require new replication instance files when upgrading from any version after V6.0-000.

## 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 e.g. RECOVER, ROLLBACK, or EXTRACT.

MUPIP UPGRADE temporarily disables journaling and replication settings for the duration of its activity. Once complete, MUPIP UPGRADE restores prior settings.

### Stage 5: Trigger Definitions Upgrade

GT.M V7.1-000 does not require trigger definition upgrade when upgrading GT.M from any version after V6.3-000. If upgrading from a prior GT.M release, please see the instructions in the release notes for V6.3-014.

## 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. During installation, GT.M provides an option that allows placing the object code in shared libraries in addition to 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_V7.1-000_i686, then gtmprofile sets $gtm_dist to /usr/lib/fis-gtm/gtm_V7.1-000_i686/utf8).
    - On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

For more information on gtmprofile, refer to the Basic Operations sect1 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.

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

## V6.3-014

Fixes and enhancements specific to V6.3-014:

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-DE201212 | GTM-9331 | Language | Prevent signal interference from external calls |
| GTM-DE201213 | GTM-9308 | Admin | Better messages from MUPIP RUNDOWN when in FREEZE -ONLINE |

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-DE201302 | GTM-9285 | Admin | MUPIP REPLICATE -SOURCE -CONNECTPARAMS changes |
| GTM-DE201318 | GTM-9322 | DB | More care to release the replication jounral pool when not doing so might cause a deadlock |
| GTM-DE201319 | GTM-9328 | Language | No interrupting an already interrupted process |
| GTM-DE201320 | GTM-9329 | Language | $ZTIMEOUT issue corrections |
| GTM-DE201324 | GTM-9335 | DB | Prevent unusual case in which an abnormal process exit might cause an apparent database hang when using AIO |
| GTM-DE201325 | GTM-9336 | Admin | Correct an issue with MUPIP SIZE occasionally failing with a damaged stack |
| GTM-DE201326 | GTM-9337 | DB | Prevent spurious error in a process that contributed to the snapshot facility used by MUPIP INTEG |
| GTM-DE201327 | GTM-9338 | Admin | Allow the $gtm_tmp environment variable to control the placement of files used in trigger compilation |
| GTM-F134422 | GTM-7628 | Admin | Up to 64GiB for Source and receiver buffers |
| GTM-F135088 | GTM-8800 | Admin | MUPIP FTOK enhancements and MUPIP SEMAPHORE replace separate utilities |
| GTM-F135114 | GTM-8863 | DB | Provide GT.M statistics indicating use of various critical sections |
| GTM-F135240 | GTM-9102 | Admin | MUPIP FREEZE is consistent across regions except for AIO |
| GTM-F135346 | GTM-9266 | Admin | Correction to replication connection failure message |
| GTM-F135362 | GTM-9320 | Admin | Suppress inapropriate EN022 syslog messages associated with use of READ_ONLY databases |
| GTM-F135363 | GTM-9321 | Language | Correct $REFERENCE maintainance in $ORDER(<indirection>,<literal>) |
| GTM-F135368 | GTM-9332 | Language | Prevent high LOCK activity over a region instantiation from causing moment of LOCK "confusion" |

## Database

- GT.M processes appropriately handle releasing the replication journal pool resource; previously in rare circumstances, processes might not release the resource as intended leading to a dead-lock that can only be cleared by stopping the process inappropriately continuing to hold the resource. (GTM-DE201318)

- GT.M deals with the loss of a process involved with AIO database writes more promptly; previously it could take material time for it to recognize and deal with the issue, which manifested as a database hang. This behavior was seen in in-house testing and was never reported by a customer.(GTM-DE201324)

- GT.M process disengaging from the snapshot mechanism used by MUPIP INTEG ignores an error indicating the associated shared memory is already gone. Previously, such a process could occasionally and inappropriately get a SYSCALL error associated with shmdt and ENO22. (GTM-DE201326)

- GT.M provides toggle statistics to indicate when a process is in a critical code section, and the general characterization of that critical section. When a process enters a critical section, it sets the appropriate toggle statistic to '1' and when it leaves the critical section,

sets it back to '0'. GT.M provides toggle statistics designated DEXA, GLB, JNL, MLK, PRC, TRX, ZAD, JOPA, AFRA, BREA, MLBA & TRGA, which are documented in the "ZSHOW" section of the GT.M Programmers Guide. All of the standard GT.M tools and facilities for examining statistics operate on the toggle statistics, and the only operational difference between the toggle statistics and the counter statistics previously provided by GT.M is that, since toggle statistics provide a near real-time report on GT.M process status rather than a cumulative state, they are neither saved on process exit nor loaded on process initialization. Note that we are still considering feedback on these added statistics, so they may be the subject of future changes. (GTM-F135114)

## Language

- The GT.M external call and timer logic ensure that GT.M recovers from external calls that ignore GT.M timers or disrupt GT.M signal handling. Note that GT.M exposes a timer mechanism for use by external calls that integrates external timer needs with those of GT.M. Note also that GT.M external calls have a configuration option that claims an external call does not change the GT.M signal handling; this option makes signal handling more efficient. As a result of changes in V6.3-006 associated with GTM-7952, an external call could suspend GT.M's timer mechanism for a process, which caused timed operations (HANG, JOB, LOCK, OPEN, READ, and some Z* operations) to suspend indefinitely. The workaround was to signal such a process with a SIGALARM. (GTM-DE201212)

- When handling a MUPIP INTRPT, GT.M discards any other such requests. Starting in V6.3-006, GT.M could inappropriately accept multiple invocations of $ZINTERRUPT causing the process to get a fatal STACKOFLOW. (GTM-DE201319)

- GT.M handles an edge case for $ZTIMEOUT properly. Previously, cancelling a timer by setting $ZTIMEOUT to -1 before completely processing the $ZTIMEOUT action for an expired timeout could cause the $ZTIMEOUT action to repeat indefinitely. GT.M retains the vector in $ZTIMEOUT properly. Previously, attempting to assign $ZTIMEOUT an incorrect vector routine, caused GT.M to inappropriately clear the prior vector value. These behaviors were seen in in-house testing and were never reported by a customer. (GTM-DE201320)

- Name-level $ORDER(<indirection>,<literal>), where the indirection contains a subscripted or unsubscripted gvn, correctly maintains $REFERENCE; a change in V6.3-013 caused this case to incorrectly maintain $REFERENCE, giving it the value of the last prior global reference rather than that within the indirection. (GTM-F135363)

- GT.M deals appropriately with large numbers of LOCK releases. One aspect of this change eliminates counting LOCK pending removals, mostly from timeouts, as wake ups in internal counters. Previously if the releases reached 2**32 in the active life of a region (between shutdowns), processes could, for some typically brief period, receive an incorrect indication they had received the LOCK, when they had not. (GTM-F135368)

## System Administration

- When MUPIP RUNDOWN encounters a region with an active FREEZE -ONLINE, it skips running down the region and issues an OFRZACTIVE warning message. Previously under these circumstances, MUPIP issued less appropriate messages. (GTM-DE201213)

- The -CONNECTPARAMS qualifier MUPIP REPLICATE -SOURCE -START includes the following fixes and enhancements:

  - -CONNECTPARAMS takes the fourth comma-delimited value as the approximate alert-time in seconds after which the Source Server records the REPLAERT message in the Source Server log file when it fails to establish a replication connection during soft connection attempts. Specify 0 if you want to disable the REPLALERT logging behavior. By default, the logging of the REPLALERT message for soft connection attempt failures is disabled. Previously, MUPIP only reported the REPLWARN message for such failures with an inaccurate time.

  - The Source Server sleeps for hard-tries-period when it fails to resolve the network address specified with the -SECONDARY qualifier. Also, the Source Server allows users to bypass hard connection attempts by specifying zero (0) as the hard-connection-count for more timely logging of the REPLALERT messages as the Source Server bypasses hard connection attempts. Previously, it was not possible to bypass hard connection attempts and the Source Server did not sleep for hard-tries-period when the network address specified with -SECONDARY resulted in a network resolution failure.

  - GT.M automatically adjusts the values for soft-tries-period and hard-tries-period to about half of the GT.M determined shutdown wait time. GT.M logs these adjustments in the Source Server log file. This helps prevent inadvertent timeouts of a Source Server shutdown

due to high soft-tries-period and/or hard-tries-period and prevents out-of-design conditions. The Source Server log file includes the units for hard-tries-period in milliseconds and the units of soft-tries-period and alert-time in seconds. Previously, the Source Server log file did not include the time measurement units.

- MUPIP reports a BADCONNECTPARAM if the specification is incorrect or the user specifies a value that may lead to an out-of-design situation. MUPIP also allows users to specify up to six parameters with -CONNECTPARAMS and uses the defaults for the parameters not specified. Previously, MUPIP displayed the same "Error parsing or invalid parameter in CONNECTPARAMS" message for parsing error in any -CONNECTPARAMS parameter and required all six parameters.

Please refer to the "Starting the Source Server" section of the Administration and Operations Guide for complete information on -CONNECTPARAMS. (GTM-DE201302)

- When MUPIP SIZE encounters an apparently invalid key, it either ignores it or reports a database integrity error, depending on the nature of the issue; previously some such keys could cause the utility to fail due to a damaged stack. (GTM-DE201325)
- Trigger compilation responds to the location specified by the gtm_tmp environment variable; previously it always used the default for gtm_tmp, which is /tmp. (GTM-DE201327)
- If there is enough free shared memory available, the MUPIP REPILCATE qualifier -BUFFSIZE accepts values of up to 64 GiB for source and receiver servers. Previously, this maximum was 4GiB. (GTM-F134422)
- MUPIP FTOK recognizes the -ID=<integer> qualifier to specify the signature use for the ftok; if unspecified, it defaults to the signature for a GT.M database file. MUPIP FTOK also recognizes the -ONLY qualifier to restrict the output to only the file's ftok; if a file is not valid and accessible, FTOK reports -1 values. By default, the utility provides additional information about the file, including its file ID which provides the basis for the FTOK. In addition, MUPIP FTOK recognizes the -[NO]HEADER qualifier; -ONLY output has no header, but by default other forms include a header that -NOHEADER suppresses. Any existing parsing of the output require adjustments to deal with the additional information and revised spacing. If the database file is unavailable, the utility defaults to the -ONLY behavior. In addition, the utility accepts a space (<SP>) delimited list of files, such as that provided by the use of the * and ? shell wildcard characters. With either the -JNLPOOL or -RECVPOOL qualifier, MUPIP FTOK ignores any files in the list. Note that to minimize ftok collisions, GT.M uses its own ftok generation algorithm. This added functionality in MUPIP FTOK replaces the standalone ftok utility provided in the GT.M distribution; that executable has been removed from the release package. Previously, MUPIP FTOK did not recognize these qualifiers, only processed a single file, and did not show the FTOK in its output. MUPIP SEMAPHORE reports the details of the a space delimited list of semaphores IDs. This replaces the undocumented standalone semstat2 utility in the GT.M distribution; that executable has been removed from the release package. Note the output format differs slightly from that of semstat2. Previously MUPIP did not support the SEMPHORE command. (GTM-F135088)
- MUPIP FREEZE -ONLINE institutes the FREEZE on all regions in parallel to produce a consistent snapshot for a multiple-region database. FREEZE -ONLINE momentarily suspends updates to all regions while instituting the freeze. This action could slow regular processing until all regions are frozen. When AIO is ON, FREEZE -ONLINE institutes the FREEZE serially. Previously, MUPIP FREEZE -ONLINE froze each region serially which could result in an inconsistent snapshot with the final regions have sequence numbers higher than the first regions. (GTM-F135240)
- The replication source server includes the correct error text when logging failed connection attempts. Previously, the wrong error text may have been included in the message. This was found during internal testing and has never been reported in the field. (GTM-F135346)
- GT.M does not issue warning messages to the syslog when processes close READ_ONLY databases; previously it did, which was noticeable when using ^%PEEKBYNAME. (GTM-F135362)

## Error and Other Messages

## BADCONNECTPARAM

**BADCONNECTPARAM,** Error parsing or invalid parameter. [XXXX]

MUPIP Error: MUPIP produces this message when there in an error in any connection parameter specified with -CONNECTPARAMS. XXXX contain a brief description of the parameter and its a valid value range.

Action: Specify valid values for the -CONNECTPARAM parameter. Refer to the -CONNECTPARAM documentation in the Administration and Operations Guide for more information.

## BADPARAMCOUNT

**BADPARAMCOUNT,** -CONNECTPARAMS accepts one to six parameter values

MUPIP Error: MUPIP produces this message when there are more than six parameters specified for -CONNECTPARAMS.

Action: Specify one to six parameters or omit -CONNECTPARAMS from the MUPIP REPLICATE -SOURCE -START command to use the default connection parameters.

## FALLINTOFLST Δ

**FALLINTOFLST,** Fall-through to a label with formallist is not allowed

Run Time/Compile Time Error: This error indicates that M code reached a label with a formallist by falling through from the previous label. When issued as a warning, it indicates the compiler determined such an error could happen and may have inserted an implicit QUIT to prevent the run-time error

Action: Revisit your code to ensure that all invocations of labels with a formallist occur using a DO command or extrinsic function ($$).

## NOPRINCIO Δ

**NOPRINCIO,** Unable to dddd principal device: DDDD at LLLL due to: SSSS

Run Time Fatal: This indicates that GT.M attempted to, but could not, READ from, or WRITE to (direction indicated by dddd), the PRINCIPAL device, and therefore attempted to issue an appropriate error, for example, an IOEOF, TERMHANGUP, or TERMWRITE at location LLLL, with a status of SSSS. However, if the error handling does not prevent any and all subsequent READs and WRITEs to the no longer available PRINCIPAL device, the next subsequent I/O error shuts down the process immediately to prevent mysteriously lost output, or, worse, an indefinite loop. The NOPRINCIO message appears in the operator log

Action: The NOPRINCIO error message is FATAL which does not drive device or trap handlers and terminates the process. This termination does not allow any application level orderly shutdown and, depending on the application, may lead to out-of-design application state. Therefore FIS recommends appropriate application level error handling that recognizes the preceding error and performs an orderly shutdown without issuing any additional READ or WRITE to the principal device. 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 becomes inaccessible making any subsequent attempt to READ from, or WRITE to, it hopeless. In the case of terminals, a user closing the window of a session without cleanly exiting from the GT.M process sets up the case that can drive this error. GT.M does not issue NOPRINCIO errors from Direct Mode, because it is a developer tool, or at the completion of a HEREDOC in a shell script. However, this means a HEREDOC must use ZHALT to return a specific status to the shell, and that a $ETRAP that bounces a process into Direct Mode terminates without evidence.

## REPLALERT

**REPLALERT,** Source Server could not connect to replicating instance [XXXX] for [NNNN] seconds

MUPIP Warning: The Source Server records this warning message when the Source Server fails to establish a replication connection with the secondary instance [XXXX] for [NNNN] seconds. The frequency of recording this warning message can be adjusted with the soft connection attempt period (the fourth -CONNECTPARAM).

Action: Use the REPLALERT message as an mechanism to alert operations about replication network issues. Specify 0 as the REPLALERT period parameter (the fourth -CONNECTPARAM) to disable logging this message. The REPLALERT messages are disabled by default (that is, without specifying -CONNECTPARAM).

## SHMHUGETLB

**SHMHUGETLB,** Could not back shared memory with huge pages, using base pages instead

All GT.M Components Warning: When the gtm_hugetlb_shm environment variable is defined and evaluates to a non-zero integer or any case-independent string or leading substring of ""TRUE"" or ""YES"" in a process creating shared memory, GT.M attempts to back all such shared memory segments with huge pages, using the default huge pages size. If huge pages cannot be used, GT.M outputs the SHMHUGETLB warning and tries to back the shared memory with base pages instead. The warning message specifies the operation of the caller along with the relevant file path for the process requesting shared memory. The warning message also includes either an ENOMEM or an EPERM error, depending on why the request for huge pages failed.

Action: If the warning includes an ENOMEM error, consider allocating more huge pages. If the EPERM error is specified, make sure the caller is privileged (i.e. has the CAP_IPC_LOCK capability) and is a member of the sysctl_hugetlb_shm_group group.