GT.M

Release Notes

V7.1-001

Empowering
the Financial World

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	26 June 2023	V7.1-001

Table of Contents

This page is intentionally left blank.

V7.1-001

Overview

V7.1-001 makes MUPIP REORG -UPGRADE available to run concurrently with other activity with the exception of any other MUPIP REORG, as well as addressing a number of issues.

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

Effective V6.0-000, GT.M documentation adopted IEC standard Prefixes for binary multiples. This document therefore uses prefixes Ki, Mi and Ti (e.g., 1MiB for 1,048,576 bytes). 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.

Δ denotes revised messages.

⊕ denotes added messages.

Platforms

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

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, 7.3 TL 1	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.** 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	To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware. As of V7.1-001, GT.M on x86-64 requires hardware/virtualized support for AVX instructions. 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).

Platform	Supported Versions	Notes
	LTS; Amazon Linux 2	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 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. 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. 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. 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

Platform	Supported Versions	Notes
		specific to SSE2. Note, depending on how CPU virtualization is configured, that virtual environments may not support AVX2 even if the underlying hardware does. Ubuntu 21.10 and Red Hat Enterprise Linux 9 are Supportable. **Note** ● To use TLSv1.3 with OpenSSL 1.1.1 and up, you must recompile the reference encryption plugins ● 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 V7.1-001 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V7.1-001_arch (for example, /usr/lib/fis-gtm/V7.1-001_x86_64 on Linux systems). A location such as /opt/fis-gtm/V7.1-001_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 V7.1-001

Before you begin

GT.M supports upgrade from V5*, V6.* and V7.* versions to V7.1-001.

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 V7.1-001 consists of 5 stages:

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

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V7.1-001 depends on your GT.M upgrade history and your current version.

Stage 1: Global Directory Upgrade

FIS strongly recommends you back up your Global Directory 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 V7.1-001.
- 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 V7.1-001.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

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 V7.1-001.

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 V7.1-001 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 V7.1-001. 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 V7.1-001 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_V7.1-001_i686, then gtmprofile sets $gtm_dist to /usr/lib/fis-gtm/gtm_V7.1-001_i686/utf8).
    - On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

For more information on gtmprofile, 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.

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

V7.1-001

Fixes and enhancements specific to V7.1-001:

Id	Prior Id	Category	Summary
GTM-DE476408	-	Other	Prevent failing JOB command from damaging shared data belonging to the issuing process
GTM-DE476835	-	DB	Prevent a rare spurious NUMOFLOW from $INCREMENT(gvn,expr)
GTM-DE500856	-	Other	^%RANDSTR limits range arguments
GTM-DE500860	-	Other	The GT.M compiler accepts files with extensions other than .m ✓
GTM-DE503394	-	Other	%YGBLSTAT issues warnings for defective command lines
GTM-DE506257	-	Language	Prevent errors in global variable subscript shifting, and prevent all shifting under EXTENDED_BOOLEAN ✓
GTM-DE506361	-	Other	GTMSECSHR handles rare race condition at its startup and shutdown
GTM-DE507982	-	Language	Ensure zero (0) derived from multiplying a non-integer by zero equates to zero
GTM-DE508852	-	Language	Correctly report NUMOFLOW errors when evaluating operations on literals at compile time
GTM-DE510902	-	Language	Prevent literal operation failures in XECUTE blocks from improperly affecting the surrounding execution environment
GTM-DE511969	-	Other	Direct Mode command RECALL restored
GTM-DE512004	-	Language	SET @expr supports long exprs and %ZSHOWVTOLCL uses them for alias containers
GTM-DE513737	-	Language	Protect the truth-value of subscripted local variables in Boolean expressions from subsequent side effects when gtm_boolean >= 1
GTM-DE513980	-	Language	$ZMAXTPTIME can interrupt a transaction holding a database critical section
GTM-DE519525	-	Language	✗ $ZTIMEOUT deferred during a TP transaction ✓
GTM-DE525624	-	Language	$ZTRANSLATE() does not issue a BADCHAR when operating on UTF-8 strings

Id	Prior Id	Category	Summary
GTM-DE527213	-	Language	On AIX, correctly handle large arguments to the HANG command
GTM-DE530712	-	DB	Proactive Block Split appropriately handles blocks with no records in them
GTM-DE531077	-	DB	Prevent multi-region trigger load operations from possibly terminating with an ASSERTPRO error
GTM-DE531078	-	DB	Online Rollback terminates active KILL In Progress (KIP) activity
GTM-DE532295	-	DB	GT.M disables proactive block splitting within TP transactions and by default
GTM-F135040		Language	$VIEW("JNLPOOL") with multiple instances
GTM-F225097	-	Admin	Second phase of in-place conversion from V6 to V7 database formats supports operation with concurrent activity ✓

Database

- $INCREMENT(gvn,expr) manages concurrency to avoid a rare spurious NUMOFLOW; previously, this the function could inappropriately issue that error, but never committed an incorrect result. (GTM-DE476835)

- Proactive block split appropriately handles blocks with no records in them. Previously, under certain circumstances, proactive block split could try to induce a block split when the block had no records in it. This was only seen during development testing and not reported by a customer. (GTM-DE530712)

- Trigger load operations involving many regions gracefully restart when a concurrent ONLINE ROLLBACK forces the process to restart transaction processing in each region. Previously, in such a situation, GT.M processes could terminate with an ASSERTPRO error. This error was only seen in development and not reported by a customer. (GTM-DE531077)

- GT.M processes working on the second phase of a KILL operation (KILL-In-Progress), issue a DBROLLEDBACK error in the event an ONLINEROLLBACK changes the state of the database. Previously, GT.M processes would restart until the fourth retry acquiring the critical section on all participating regions before issuing such an error. This error was only seen in development and not reported by a customer. (GTM-DE531078)

- GT.M disables proactive block splitting by default as well as within a TP transaction. FIS made this change after observing that, under certain conditions, the setting could incur spurious restarts and split blocks which were not subject to contention. To enable the feature outside of transaction processing, use a MUPIP SET -PROBLKSPLIT=n, where n is the number of nodes at which GT.M considers based on the number of restarts whether to split a block in advance of its becoming full. Previously, starting in V7.0-001, the default threshold for a proactive block split was 5 nodes and the feature applied to TP as well as non-TP. The performance issue was only ever observed in testing and not reported by a customer; it was not associated with any correctness or integrity concerns. (GTM-DE532295)

This page is intentionally left blank.

Language

- GT.M appropriately evaluates subscripts containing or depending on side effects in lock resource and global variable names. Previously, due to shifting of evaluations to avoid unnecessary global access while appropriately maintaining $REFERENCE, rare constructs with such subscripts could produce out-of-order side effects. In addition, the EXTENDED_BOOLEAN setting prevents such shifting entirely; previously, EXTENDED_BOOLEAN behavior was identical to that of the more efficient FULL_BOOLEAN. (GTM-DE506257) ✓

- GT.M produces an appropriate result from an equality test between zero (0) and a multiplicand of zero value that is the result of multiplying a non-integer value with zero (0). Previously, this odd combination could inappropriately indicate inequality. (GTM-DE507982)

- GT.M appropriately reports a NUMOFLOW error for literal values when those literals are operated on at compile-time and the run-time variable equivalent would trigger the same error. This affects the unaryoperators ',+, and -. The effect of these operators on variables is unchanged. Previously, these operators sometimes did not correctly report a NUMOFLOW when called for. In the case of the negation operator, this could lead to reporting an incorrect result. (GTM-DE508852)

- GT.M prevents compile-time errors in operations on literals within an XECUTE block from terminating the XECUTE without properly cleaning up the surrounding compilation environment. Previously, this could cause termination of compilation of the routine containing the XECUTE and failure to compile subsequent routines passed to the same GT.M process (GTM-DE510902)

- SET @($ZWRTAC=expr) works for strings approaching 1MiB; previously such an indirect expression was limited to the maximum length of a source line. Also, the %ZSHOWVTOLCL utility routine now deals with more cases, such as such longer strings. (GTM-DE512004)

- GT.M returns the correct value for a Boolean expression containing a subscripted local variable when that variable is affected by side-effects later in the expression and gtm_boolean>=1. Previously, GT.M evaluated the boolean using the value of the subscripted local after all side-effects. This issue did not affect unsubscripted local variables or globals of any kind. (GTM-DE513737)

- GT.M correctly executes the $ETRAP/$ZTRAP exception handler at the time of expiry of $ZMAXTPTIME when the process holds a database critical section. Previously, due to a regression in V7.0-001, the $ZMAXTPTIME timer did not execute $ETRAP/$ZTRAP exception handler until the process released all database critical sections which could allow a transaction to materially exceed the specified $ZMAXTPTIME. (GTM-DE513980)

- GT.M defers $ZTIMEOUT when its expiry happens during a TP transaction. $ZMAXTPTIME may interrupt a transaction, as optionally may MUPIP INTRPT initiated $ZINTERRUPT processing, but $ZTIMEOUT acts after a TROLLBACK or master TCOMMIT. Previously, GT.M allowed a $ZTIMEOUT expiry within a TP transaction. (GTM-DE519525) ✓✓

- Because it is a byte-oriented function, $ZTRANSLATE() does not issue a BADCHAR when operating on UTF-8 strings. Due to a regression in V6.3-013, this function incorrectly issued a BADCHAR error when the input string contained non-UTF-8 characters. There are two possible workarounds for this issue: 1) A targeted approach requiring code changes to enclose each use of $ZTRANSLATE() with VIEW "NOBADCHAR" and VIEW "BADCHAR"; ensures all UTF-8 data is handled appropriately. 2) A broad approach of defining gtm_badchar as zero or FALSE to disable BADCHAR checking throughout the application which disables detection of any improperly formatted UTF-8 strings. (GTM-DE525624)

- GT.M on AIX correctly handles HANG arguments specifying more than 2 million seconds, or more than 24 days. Previously, inappropriate integer overflow caused larger values to hang for less time than expected, and capped the possible hang length on AIX at roughly 4300 seconds or approximately one hour and ten minutes. [AIX] (GTM-DE527213)

- Specifying a second expression for $VIEW("JNLPOOL") provides a means of iterating through active Journal Pools. If the second expression is an empty string, the function returns the replication instance file name associated with the instance first attached by the process or the string "*" if the process has not previously engaged with any instance. If the file name specified in the second expression does not match the replication instance file name for any of the active Journal Pools the string "*" is returned. Otherwise the file name of the Journal Pool attached after the Journal Pool with the file name given in the second expression is returned. Note the two argument form of $VIEW("JNLPOOL") does not change the current Replication Instance. (GTM-F135040)

System Administration

- MUPIP REORG -UPGRADE, which completes the second phase of a V6 to V7 database transition, can run concurrently with other processing excepting other MUPIP REORG [-UPGRADE] processes. REORG -UPGRADE can work either by region or by file allowing administrator to run concurrent upgrade operations on different regions/files.

  > FIS does not recommend running MUPIP REORG -UPGRADE with concurrent activity for MM database files on AIX due to a rare concurrency conflict.

  MUPIP DOWNGRADE -VERSION=V63000A allows the current GT.M to downgrade a V6 database to the pre-V63000A EOF block format. Previously the downgrade function reported an unsupported error for a V7 versions. (GTM-F225097) ✓

This page is intentionally left blank.

Other

- A process created by a JOB command that for any reason exits before becoming fully functional and independent avoids interfering with any data it may share with the process issuing the JOB command. GT.M also issues a PIDMISMATCH warning message to the operator log. Previously under rare conditions, such an "infant" process could inappropriately release resources belonging to its "parent," causing symptoms including damage to a statsdb database or to a routine repository. (GTM-DE476408)
- ^%RANDSTR limits the range argument upper limit to the actual number of characters available in the current character set - 256 for M mode and 65,536 for UTF-8 mode. Previously, a missing or defective upper limit caused the routine to perform unproductive processing that could consume unreasonable amounts of time. The workaround was to avoid inappropriate range arguments. (GTM-DE500856)
- The GT.M compiler accepts source files with arbitrary extensions. FIS recommends using the .m extension for source files as our testing of that is very extensive, however there may be cases where other extensions serve a purpose. Previously, the compiler enforced an explicit or implicit .m extension for source files. (GTM-DE500860) ✓
- %YGBLSTAT warns about incorrect command line usage. Previously, the utility silently ignored command lines containing errors. (GTM-DE503394)
- GTMSECSHR appropriately handles a rare condition when two processes attempt to start a GTMSECSHR process at a coincident time. Previously, this could start more than one GTMSECSHR process, and, although a single GTMSECSHR process handled all the requests, their shutting down produced syslog error messages. (GTM-DE506361)
- The Direct Mode only RECALL command functions as documented; starting in V7.0-002 it always gave an error. (GTM-DE511969)

This page is intentionally left blank.

Error and Other Messages

MUTRUNCNOSPACE

MUTRUNCNOSPACE, Region rrrr has insufficient space to meet truncate target percentage of yyyy

MUPIP Information: MUPIP REORG -TRUNCATE produces this message in the following conditions when it determines that there is insufficient free space at the end of the database file to meet the requested percentage, the region is almost full or there are globals not mapped to the region, or there is a concurrent MUPIP EXTEND.

Action: Specify a less aggressive threshold. If a global is not mapped to the region, REORG cannot perform any operation on that global. Create a separate global directory for the global mapped to the region and then perform the REORG.

PIDMISMATCH

PIDMISMATCH, Global variable with PID=pppp does not match specified PID=qqqq

Run Time Warning: pppp is the process_id global variable while qqqq is produced by getpid(). This message indicates we skip exit handling after fork() to avoid a child process from removing the statsdb entry of its parent, which causing symptoms including damage to a database or to a routine repository.

Action: No action is required, unless we see a REQRUNDOWN or REQRLNKCTLRNDWN shortly afterwards.

This page is intentionally left blank.