

GT.M

Release Notes

V6.3-003

Contact Information

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

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

Legal Notice

Copyright © 2017-2019 Fidelity National Information Services, Inc. and/or its subsidiaries. All Rights Reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts.

GT.M™ is a trademark of Fidelity National Information Services, Inc. Other trademarks are the property of their respective owners.

This document contains a description of GT.M and the operating instructions pertaining to the various functions that comprise the system. This document does not contain any commitment of FIS. FIS believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. FIS is not responsible for any errors or defects.

Revision History		
Revision 1.5	5 February 2019	Updated the Platforms section to add AIX 7.1 TL 4 and AIX 7.2 as supported versions; Correct the maximum V6 database size.
Revision 1.4	20 August 2018	Add GTM-9020 and GTM-9023.
Revision 1.3	19 June 2018	* Remove GTM-8859 as it was fixed in GT.M V6.3-004. * Make corrections to GTM-7986, GTM-8732, GTM-8832 and GTM-8880.
Revision 1.2	17 April 2018	Change category of GTM-8769 to   .
Revision 1.1	22 January 2018	* Updated for V6.3-003A. Added GTM-8880, GTM-8887, and GTM-8889. * Added GTM-8846 and GTM-8873 which were missed in the V6.3-003 Release Notes due to a documentation error.
Revision 1.0	12 December 2017	V6.3-003

Table of Contents

V6.3-003A	1
Overview	1
Conventions	1
Platforms	3
Platform support lifecycle	5
32- vs. 64-bit platforms	5
Call-ins and External Calls	5
Internationalization (Collation)	6
Environment Translation	6
Additional Installation Instructions	7
.....	7
Upgrading to GT.M V6.3-003A	9
Stage 1: Global Directory Upgrade	9
Stage 2: Database Files Upgrade	10
Stage 3: Replication Instance File Upgrade	12
Stage 4: Journal Files Upgrade	13
Stage 5: Trigger Definitions Upgrade	13
Downgrading to V5 or V4	14
Managing M mode and UTF-8 mode	15
Setting the environment variable TERM	17
Installing Compression Libraries	17
.....	19
V6.3-003B	19
V6.3-003A	19
V6.3-003	19
Database	23
Language	27
System Administration	31
Other	35
Error and Other Messages	37
DBFREEZE ⁺ OFF	37
DBFREEZE ⁺ ON	37
DBNONUMSUBS ^Δ	37
DBNULCOL ^Δ	37
GBLOFLOW ^Δ	38
LSINSERTED ⁺	38
MUTEXFRCDTERM ^Δ	38
NULSUBSC ^Δ	38
READONLYNOBG ⁺	39
REPLINSTACC ^Δ	39
REPLINSTMISMATCH ^Δ	39
REPLMULTINSTUPDATE ⁺	40
STACKCRIT ^Δ	40

STACKOFLOW 	40
STPCRIT 	40
STPOFLOW 	40

V6.3-003A

Overview

V6.3-003B adds some capabilities to the restriction facility introduced in V6.3-002. It improves performance when you have a large number of concurrent M LOCKs. While we have designated it as field test grade, perhaps the biggest new thing is the ability for a process to access multiple instances during its execution (GTM-8182). We are interested in your explorations of this and are ready to answer questions you encounter as you explore the facility.

As always, the release bring numerous smaller enhancements, and fixes. See the Change History below. Please pay special attention to the items marked with the symbols  or .



Note

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

Conventions

This document uses the following conventions:

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



Note

The term UNIX refers to the general sense of all platforms on which GT.M uses a POSIX API. As of this date, this includes: AIX and GNU/Linux on x86 (32- and 64-bits).

The following table summarizes the new and revised replication terminology and qualifiers.

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
originating instance or primary instance	-rootprimary	originating instance or originating primary instance. Within the context of a replication connection between two instances, an originating instance is referred to as source instance or source side. For example, in an B<-A->C replication 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.	-updnok
N/A	N/A	supplementary instance. For example, in an A->P->Q replication configuration, P is the supplementary instance. Both A and P are originating instances.	-updok

Effective V6.0-000, GT.M documentation adopted IEC standard Prefixes for binary multiples. This document therefore uses prefixes Ki, Mi and Ti (e.g., 1MiB for 1,048,576 bytes). Over time, we'll update all GT.M documentation to this standard.

- ✔ denotes a new feature that requires updating the manuals.
- ⊕ denotes a new feature or an enhancement that may not be upward compatible and may affect an existing application.
- ⊖ denotes deprecated messages.
- ⚠ denotes revised messages.
- ⊕ denotes added messages.

Platforms

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

Each GT.M release is extensively tested by FIS on a set of specific versions of operating systems on specific hardware architectures (the combination of operating system and hardware architecture is referred to as a platform). This set of specific versions is considered Supported. There may be other versions of the same operating systems on which a GT.M release may not have been tested, but on which the FIS GT.M Group knows of no reason why GT.M would not work. This larger set of versions is considered Supportable. There is an even larger set of platforms on which GT.M may well run satisfactorily, but where the FIS GT.M team lacks the knowledge to determine whether GT.M is Supportable. These are considered Unsupported. Contact FIS GT.M Support with inquiries about your preferred platform.

As of the publication date, FIS supports this release on the hardware and operating system versions below. Contact FIS for a current list of Supported platforms. The reference implementation of the encryption plugin has its own additional requirements, should you opt to use it as included with GT.M.

Platform	Supported Versions	Notes
IBM Power Systems AIX	7.1 TL 4, 7.2	<p>Only 64-bit versions of AIX with POWER7 as the minimum required CPU architecture level are Supported.</p> <p>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.</p> <p>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.</p> <p>AIX 7.1 TL 5 is Supportable.</p> <p>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.</p>
x86_64 GNU/Linux	Red Hat Enterprise Linux 7.3; Ubuntu 16.04 LTS	<p>To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware.</p> <p>GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6.32 or</p>

Platform	Supported Versions	Notes
		<p>later), glibc (version 2.12 or later) and ncurses (version 5.7 or later).</p> <p>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.</p> <p>GT.M requires the libtinfo library. If it is not already installed on your system, and is available using the package manager, install it using the package manager. If a libtinfo package is not available:</p> <ul style="list-style-type: none"> * Find the directory where libncurses.so is installed on your system. * Change to that directory and make a symbolic link to libncurses.so.<ver> from libtinfo.so.<ver>. Note that some of the libncurses.so entries may themselves be symbolic links, for example, libncurses.so.5 may itself be a symbolic link to libncurses.so.5.9. <p>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.</p> <p>Although GT.M itself does not require libelf, the geteuid program used by the GT.M installation script requires libelf (packaged as libelf1 on current Debian/Ubuntu distributions and elfutils-libelf on RHEL 6 & 7).</p> <p>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 <i>must</i> 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</p>

Platform	Supported Versions	Notes
		NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs.
x86 GNU/Linux	Debian 9 (Stretch)	This 32-bit version of GT.M runs on either 32- or 64-bit x86 platforms; we expect the x86_64 GNU/Linux version of GT.M to be preferable on 64-bit hardware. Running a 32-bit GT.M on a 64-bit GNU/Linux requires 32-bit libraries to be installed. The CPU must have an instruction set equivalent to 586 (Pentium) or better. Please also refer to the notes above on x86_64 GNU/Linux.

Platform support lifecycle

FIS usually supports new operating system versions six months or so after stable releases are available and we usually support each version for a two year window. GT.M releases are also normally supported for two years after release. While FIS will attempt to provide support to customers in good standing for any GT.M release and operating system version, our ability to provide support diminishes after the two year window.

GT.M cannot be patched, and bugs are only fixed in new releases of software.

32- vs. 64-bit platforms

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

- * You must compile the application code separately for each platform. Even though the M source code is the same, the generated object modules are different - the object code differs between x86 and x86_64.
- * Parameter-types that interface GT.M with non-M code using C calling conventions must match the data-types on their target platforms. Mostly, these parameters are for call-ins, external calls, internationalization (collation) and environment translation, and are listed in the tables below. Note that most addresses on 64-bit platforms are 8 bytes long and require 8 byte alignment in structures whereas all addresses on 32-bit platforms are 4 bytes long and require 4-byte alignment in structures.

Call-ins and External Calls

Parameter type	32-Bit	64-bit	Remarks
gtm_long_t	4-byte (32-bit)	8-byte (64-bit)	gtm_long_t is much the same as the C language long type.

Parameter type	32-Bit	64-bit	Remarks
gtm_ulong_t	4-byte	8-byte	gtm_ulong_t is much the same as the C language unsigned long type.
gtm_int_t	4-byte	4-byte	gtm_int_t has 32-bit length on all platforms.
gtm_uint_t	4-byte	4-byte	gtm_uint_t has 32-bit length on all platforms



Caution

If your interface uses gtm_long_t or gtm_ulong_t types but your interface code uses int or signed int types, failure to revise the types so they match on a 64-bit platform will cause the code to fail in unpleasant, potentially dangerous, and hard to diagnose ways.

Internationalization (Collation)

Parameter type	32-Bit	64-bit	Remarks
gtm_descriptor in gtm_descript.h	4-byte	8-byte	Although it is only the address within these types that changes, the structures may grow by up to 8 bytes as a result of compiler padding to meet platform alignment requirements.



Important

Assuming other aspects of code are 64-bit capable, collation routines should require only recompilation.

Environment Translation

Parameter type	32-Bit	64-bit	Remarks
gtm_string_t type in gtmxc_types.h	4-byte	8-byte	Although it is only the address within these types that changes, the structures may grow by up to 8 bytes as a result of compiler padding to meet platform alignment requirements.



Important

Assuming other aspects of code are 64-bit capable, environment translation routines should require only recompilation.

Additional Installation Instructions

To install GT.M, see the "Installing GT.M" section in the GT.M Administration and Operations Guide. For minimal down time, upgrade a current replicating instance and restart replication. Once that replicating instance is current, switch it to become the originating instance. Upgrade the prior originating instance to become a replicating instance, and perform a switchover when you want it to resume an originating primary role.



Caution

Never replace the binary image on disk of any executable file while it is in use by an active process. It may lead to unpredictable results. Depending on the operating system, these results include but are not limited to denial of service (that is, system lockup) and damage to files that these processes have open (that is, database structural damage).

- * FIS strongly recommends installing each version of GT.M in a separate (new) directory, rather than overwriting a previously installed version. If you have a legitimate need to overwrite an existing GT.M installation with a new version, you must first shut down all processes using the old version. FIS suggests installing GT.M V6.3-003A in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.3-003A_arch (for example, /usr/lib/fis-gtm/V6.3-003A_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.3-003A_arch would also be appropriate. Note that the **arch** suffix is especially important if you plan to install 32- and 64-bit versions of the same release of GT.M on the same system.
- * Use the appropriate MUPIP command (e.g. ROLLBACK, RECOVER, RUNDOWN) of the old GT.M version to ensure all database files are cleanly closed.
- * Make sure gtmsecshr is not running. If gtmsecshr is running, first stop all GT.M processes including the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP pid_of_gtmsecshr**.
- * Starting with V6.2-000, GT.M no longer supports the use of the deprecated \$gtm_dbkeys and the master key file it points to for database encryption. To convert master files to the libconfig format, please click  to download the CONVDBKEYS.m program and follow instructions in the comments near the top of the program file. You can also download CONVDBKEYS.m from <http://tinco.pair.com/bhaskar/gtm/doc/articles/downloadables/CONVDBKEYS.m>. If you are using \$gtm_dbkeys for database encryption, please convert master key files to libconfig format immediately after upgrading to V6.2-000 or later. Also, modify your environment scripts to include the use of gtmcrypt_config environment variable.

Recompile

- * Recompile all M and C source files.

Rebuild Shared Libraries or Images

- * Rebuild all Shared Libraries after recompiling all M and C source files.
- * If your application is not using object code shared using GT.M's auto-relink functionality, please consider using it.

Compiling the Reference Implementation Plugin

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

1. Install the development headers and libraries for libgcrypt, libpgpme, 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 libpgpme11-dev libconfig-dev libssl-dev
```

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

2. Unpack \$gtm_dist/plugin/gtmcrypt/source.tar to a temporary directory.

```
mkdir /tmp/plugin-build
cd /tmp/plugin-build
cp $gtm_dist/plugin/gtmcrypt/source.tar .
tar -xvf source.tar
```

3. Follow the instructions in the README.
 - * Open Makefile with your editor; review and edit the common header (IFLAGS) and library paths (LIBFLAGS) in the Makefile to reflect those on your system.
 - * Define the gtm_dist environment variable to point to the absolute path for the directory where you have GT.M installed
 - * Copy and paste the commands from the README to compile and install the encryption plugin with the permissions defined at install time



Caution

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

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

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

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

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

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

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

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

Upgrading to GT.M V6.3-003A

The GT.M database consists of four types of components- database files, journal files, global directories, and replication instance files. The format of some database components differs for 32-bit and 64-bit GT.M releases for the x86 GNU/Linux platform.

GT.M upgrade procedure for V6.3-003A consists of 5 stages:

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

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V6.3-003A depends on your GT.M upgrade history and your current version.

Stage 1: Global Directory Upgrade

FIS strongly recommends you back up your Global Directory file before upgrading. There is no one-step method for downgrading a Global Directory file to an older format.

To upgrade from any previous version of GT.M:

- * Open your Global Directory with the GDE utility program of GT.M V6.3-003A.

* Execute the EXIT command. This command automatically upgrades the Global Directory.

To switch between 32- and 64-bit global directories on the x86 GNU/Linux platform:

1. Open your Global Directory with the GDE utility program on the 32-bit platform.
2. On GT.M versions that support SHOW -COMMAND, execute SHOW -COMMAND -FILE=file-name. This command stores the current Global Directory settings in the specified file.
3. On GT.M versions that do not support GDE SHOW -COMMAND, execute the SHOW -ALL command. Use the information from the output to create an appropriate command file or use it as a guide to manually enter commands in GDE.
4. Open GDE on the 64-bit platform. If you have a command file from 2. or 3., execute @file-name and then run the EXIT command. These commands automatically create the Global Directory. Otherwise use the GDE output from the old Global Directory and apply the settings in the new environment.

An analogous procedure applies in the reverse direction.

If you inadvertently open a Global Directory of an old format with no intention of upgrading it, execute the QUIT command rather than the EXIT command.

If you inadvertently upgrade a global directory, perform the following steps to downgrade to an old GT.M release:

- * Open the global directory with the GDE utility program of V6.3-003A.
- * Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

Stage 2: Database Files Upgrade

To upgrade from GT.M V5.0*/V5.1*/V5.2*/V5.3*/V5.4*/V5.5:

A V6 database file is a superset of a V5 database file and has potentially longer keys and records. Therefore, upgrading a database file requires no explicit procedure. After upgrading the Global Directory, opening a V5 database with a V6 process automatically upgrades fields in the database fileheader.

A database created with V6 supports up to 992Mi blocks and is not backward compatible. V6 databases that take advantage of V6 limits on key size and records size cannot be downgraded. Use MUPIP DOWNGRADE -VERSION=V5 to downgrade a V6 database back to V5 format provided it meets the database downgrade requirements. For more information on downgrading a database, refer to Downgrading to V5 or V4.



Important

A V5 database that has been automatically upgraded to V6 can perform all GT.M V6.3-003A operations. However, that database can only grow to the maximum size of the version in which it was originally created. A database created on V5.0-000 through V5.3-003 has maximum size of 128Mi blocks. A database created on V5.4-000 through V5.5-000 has a maximum size of 224Mi blocks. A database file created with V6.0-000 (or above) can grow up to a maximum of 992Mi blocks. This means that, for example, the maximum size of a V6 database file having 8KiB block size is 7936GiB (8KiB*992Mi).



Important

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

If your database has any previously used but free blocks from an earlier upgrade cycle (V4 to V5), you may need to execute the MUPIP REORG -UPGRADE command. If you have already executed the MUPIP REORG -UPGRADE command in a version prior to V5.3-003 and if subsequent versions cannot determine whether MUPIP REORG -UPGRADE performed all required actions, it sends warnings to the syslog requesting another run of MUPIP REORG -UPGRADE. In that case, perform any one of the following steps:

- * Execute the MUPIP REORG -UPGRADE command again, or
- * Execute the DSE CHANGE -FILEHEADER -FULLY_UPGRADED=1 command to stop the warnings.



Caution

Do not run the DSE CHANGE -FILEHEADER -FULLY_UPGRADED=1 command unless you are absolutely sure of having previously run a MUPIP REORG -UPGRADE from V5.3-003 or later. An inappropriate DSE CHANGE -FILEHEADE -FULLY_UPGRADED=1 may lead to database integrity issues.

You do not need to run MUPIP REORG -UPGRADE on:

- * A database that was created by a V5 MUPIP CREATE
- * A database that has been completely processed by a MUPIP REORG -UPGRADE from V5.3-003 or later.

For additional upgrade considerations, refer to Database Compatibility Notes.

To upgrade from a GT.M version prior to V5.000:

You need to upgrade your database files only when there is a block format upgrade from V4 to V5. However, some versions, for example, database files which have been initially been created with V4

(and subsequently upgraded to a V5 format) may additionally need a MUPIP REORG -UPGRADE operation to upgrade previously used but free blocks that may have been missed by earlier upgrade tools.

- * Upgrade your database files using in-place or traditional database upgrade procedure depending on your situation. For more information on in-place/traditional database upgrade, see Database Migration Technical Bulletin.
- * Run the MUPIP REORG -UPGRADE command. This command upgrades all V4 blocks to V5 format.



Note

Databases created with GT.M releases prior to V5.0-000 and upgraded to a V5 format retain the maximum size limit of 64Mi (67,108,864) blocks.

Database Compatibility Notes

- * Changes to the database file header may occur in any release. GT.M automatically upgrades database file headers as needed. Any changes to database file headers are upward and downward compatible within a major database release number, that is, although processes from only one GT.M release can access a database file at any given time, processes running different GT.M releases with the same major release number can access a database file at different times.
- * Databases created with V5.3-004 through V5.5-000 can grow to a maximum size of 224Mi (234,881,024) blocks. This means, for example, that with an 8KiB block size, the maximum database file size is 1,792GiB; this is effectively the size of a single global variable that has a region to itself and does not itself span regions; a database consists of any number of global variables. A database created with GT.M versions V5.0-000 through V5.3-003 can be upgraded with MUPIP UPGRADE to increase the limit on database file size from 128Mi to 224Mi blocks.
- * Databases created with V5.0-000 through V5.3-003 have a maximum size of 128Mi (134, 217,728) blocks. GT.M versions V5.0-000 through V5.3-003 can access databases created with V5.3-004 and later as long as they remain within a 128Mi block limit.
- * Database created with V6.0-000 or above have a maximum size of 1,040,187,392(992Mi) blocks.
- * For information on downgrading a database upgraded from V6 to V5, refer to: Downgrading to V5 or V4.

Stage 3: Replication Instance File Upgrade

V6.3-003A does not require new replication instance files if you are upgrading from V5.5-000. However, V6.3-003A requires new replication instance files if you are upgrading from any version prior to V5.5-000. Instructions for creating new replication instance files are in the Database Replication chapter of the GT.M Administration and Operations Guide. Shut down all Receiver Servers on other instances that are to receive updates from this instance, shut down this instance Source Server(s), recreate the

instance file, restart the Source Server(s) and then restart any Receiver Server for this instance with the -UPDATERESYNC qualifier.



Note

Without the -UPDATERESYNC qualifier, the replicating instance synchronizes with the originating instance using state information from both instances and potentially rolling back information on the replicating instance. The -UPDATERESYNC qualifier declares the replicating instance to be in a wholesome state matching some prior (or current) state of the originating instance; it causes MUPIP to update the information in the replication instance file of the originating instance and not modify information currently in the database on the replicating instance. After this command, the replicating instance catches up to the originating instance starting from its own current state. Use -UPDATERESYNC only when you are absolutely certain that the replicating instance database was shut down normally with no errors, or appropriately copied from another instance with no errors.



Important

You must always follow the steps described in the Database Replication chapter of the GT.M Administration and Operations Guide when migrating from a logical dual site (LDS) configuration to an LMS configuration, even if you are not changing GT.M releases.

Stage 4: Journal Files Upgrade

On every GT.M upgrade:

- * Create a fresh backup of your database.
- * Generate new journal files (without back-links).



Important

This is necessary because MUPIP JOURNAL cannot use journal files from a release other than its own for RECOVER, ROLLBACK, or EXTRACT.

Stage 5: Trigger Definitions Upgrade

If you are upgrading from V5.4-002A/V5.4-002B/V5.5-000 to V6.3-003A and you have database triggers defined in V6.2-000 or earlier, you need to ensure that your trigger definitions are wholesome in the older version and then run MUPIP TRIGGER -UPGRADE. If you have doubts about the wholesomeness of the trigger definitions in the old version use the instructions below to capture the definitions delete them in the old version (-*), run MUPIP TRIGGER -UPGRADE in V6.3-003A and then reload them as described below.

You need to extract and reload your trigger definitions only if you are upgrading from V5.4-000/V5.4-000A/V5.4-001 to V6.3-003A or if you find your prior version trigger definitions have problems. For versions V5.4-000/V5.4-000A/V5.4-001 this is necessary because multi-line XECUTEs for triggers require a different internal storage format for triggers which makes triggers created in V5.4-000/V5.4-000A/V5.4-001 incompatible with V5.4-002/V5.4-002A/V5.4-002B/V5.5-000/V6.0-000/V6.0-001/V6.3-003A.

To extract and reapply the trigger definitions on V6.3-003A using MUPIP TRIGGER:

1. Using the old version, execute a command like **mupip trigger -select="*" trigger_defs.trg**. Now, the output file `trigger_defs.trg` contains all trigger definitions.
2. Place `-*` at the beginning of the `trigger_defs.trg` file to remove the old trigger definitions.
3. Using V6.3-003A, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.3-003A replicating instance using `$ZTRIGGER()`:

1. Shut down the instance using the old version of GT.M.
2. Execute a command like **mumps -run %XCMD 'i \$ztrigger("select") > trigger_defs.trg**. Now, the output file `trigger_defs.trg` contains all trigger definitions.
3. Turn off replication on all regions.
4. Run **mumps -run %XCMD 'i \$ztrigger("item","-*")** to remove the old trigger definitions.
5. Perform the upgrade procedure applicable for V6.3-003A.
6. Run **mumps -run %XCMD 'if \$ztrigger("file","trigger_defs.trg")** to reapply your trigger definitions.
7. Turn replication on.
8. Connect to the originating instance.



Note

Reloading triggers rennumbers automatically generated trigger names.

Downgrading to V5 or V4

You can downgrade a GT.M V6 database to V5 or V4 format using MUPIP DOWNGRADE.

Starting with V6.0-000, MUPIP DOWNGRADE supports the `-VERSION` qualifier with the following format:

```
MUPIP DOWNGRADE -VERSION=[V5|V4]
```

-VERSION specifies the desired version for the database header.

To qualify for a downgrade from V6 to V5, your database must meet the following requirements:

1. The database was created with a major version no greater than the target version.
2. The database does not contain any records that exceed the block size (spanning nodes).
3. The sizes of all the keys in database are less than 256 bytes.
4. There are no keys present in database with size greater than the Maximum-Key-Size specification in the database header, that is, Maximum-Key-Size is assured.
5. The maximum Record size is small enough to accommodate key, overhead, and value within a block.

To verify that your database meets all of the above requirements, execute MUPIP INTEG -NOONLINE. Note that the integrity check requires the use of -NOONLINE to ensure no concurrent updates invalidate the above requirements. Once assured that your database meets all the above requirements, MUPIP DOWNGRADE -VERSION=V5 resets the database header to V5 elements which makes it compatible with V5 versions.

To qualify for a downgrade from V6 to V4, your database must meet the same downgrade requirements that are there for downgrading from V6 to V5.

If your database meets the downgrade requirements, perform the following steps to downgrade to V4:

1. In a GT.M V6.3-003A environment:
 - a. Execute MUPIP SET -VERSION=v4 so that GT.M writes updates blocks in V4 format.
 - b. Execute MUPIP REORG -DOWNGRADE to convert all blocks from V6 format to V4 format.
2. Bring down all V6 GT.M processes and execute MUPIP RUNDOWN -FILE on each database file to ensure that there are no processes accessing the database files.
3. Execute MUPIP DOWNGRADE -VERSION=V4 to change the database file header from V6 to V4.
4. Restore or recreate all the V4 global directory files.
5. Your database is now successfully downgraded to V4.

Managing M mode and UTF-8 mode

With International Components for Unicode (ICU) version 3.6 or later installed, GT.M's UTF-8 mode provides support for Unicode® (ISO/IEC-10646) character strings. On a system that does not have ICU 3.6 or later installed, GT.M only supports M mode.

On a system that has ICU installed, GT.M optionally installs support for both M mode and UTF-8 mode, including a utf8 subdirectory of the directory where GT.M is installed. From the same source

file, depending upon the value of the environment variable `gtm_chset`, the GT.M compiler generates an object file either for M mode or UTF-8 mode. GT.M generates a new object file when it finds both a source and an object file, and the object predates the source file and was generated with the same setting of `$gtm_chset/$ZCHset`. A GT.M process generates an error if it encounters an object file generated with a different setting of `$gtm_chset/$ZCHset` than that processes' current value.

Always generate an M object module with a value of `$gtm_chset/$ZCHset` matching the value processes executing that module will have. As the GT.M installation itself contains utility programs written in M, their object files also conform to this rule. In order to use utility programs in both M mode and UTF-8 mode, the GT.M installation ensures that both M and UTF-8 versions of object modules exist, the latter in the `utf8` subdirectory. This technique of segregating the object modules by their compilation mode prevents both frequent recompiles and errors in installations where both modes are in use. If your installation uses both modes, consider a similar pattern for structuring application object code repositories.

GT.M is installed in a parent directory and a `utf8` subdirectory as follows:

- * Actual files for GT.M executable programs (`mumps`, `mupip`, `dse`, `lke`, and so on) are in the parent directory, that is, the location specified for installation.
- * Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for UTF-8 mode in the `utf8` subdirectory, and one compiled without support for UTF-8 mode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install UTF-8 mode support. Note that on 64-bit versions of GT.M, the object code is in shared libraries, rather than individual files in the directory.
- * The `utf8` subdirectory has files called `mumps`, `mupip`, `dse`, `lke`, and so on, which are relative symbolic links to the executables in the parent directory (for example, `mumps` is the symbolic link `../mumps`).
- * When a shell process sources the file `gtmprofile`, the behavior is as follows:
 - * If `$gtm_chset` is "m", "M" or undefined, there is no change from the previous GT.M versions to the value of the environment variable `$gtmroutines`.
 - * If `$gtm_chset` is "UTF-8" (the check is case-insensitive),
 - * `$gtm_dist` is set to the `utf8` subdirectory (that is, if GT.M is installed in `/usr/lib/fis-gtm/gtm_V6.3-003A_i686`, then `gtmprofile` sets `$gtm_dist` to `/usr/lib/fis-gtm/gtm_V6.3-003A_i686/utf8`).
 - * On platforms where the object files have not been placed in a `libgtmutil.so` shared library, the last element of `$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(smkn), 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.

V6.3-003B

Fixes and enhancements specific to V6.3-003B:

Id	Prior Id	Category	Summary
GTM-9020	-	DB	Prevent problems when the LOCK space fills.
GTM-9023	-	DB	Fix rare, but serious issues with LOCKs by reverting GTM-8680

V6.3-003A

Fixes and enhancements specific to V6.3-003A:

Id	Prior Id	Category	Summary
GTM-8880	-	Language	Fix issue with (non-default) Standard Boolean evaluation with side-effects and certain patterns
GTM-8887	-	Other	Fix rare timer issue
GTM-8889	-	Other	Prevent UNDEF error after <CTRL-C> within ZHELP navigation

V6.3-003

Fixes and enhancements specific to V6.3-003:

Id	Prior Id	Category	Summary
GTM-4212	C9C03-001944	Admin	MUPIP better deals with over length file names
GTM-6115	C9I01-002944	Language	Please see GTM-8694
GTM-7986	-	Language	Warning on implicit wrapping of source lines exceeding maximum supported length
GTM-8182	-	DB	Allow updating globals belonging to different instances 🟢
GTM-8186	-	Language	Accept offset alone for an entryref in DO, GOTO and ZGOTO 🟢

Id	Prior Id	Category	Summary
GTM-8587	-	Language	Maintain \$DEVICE and \$KEY for all supported devices 🟢
GTM-8617	-	Admin	MUPIP SET supports N[ULL_SUBSCRIPTS] and STD[NULLCOLL] qualifiers. 🟢
GTM-8680	-	DB	🔴 LOCK Improvements
GTM-8732	-	Admin	Better validation for MUPIP REPLICATE -LOG_INTERVAL and -HELPER, and MUPIP SET -DEFER_TIME
GTM-8735	-	Admin	READ_ONLY characteristic to prevent state changes to MM databases 🟢
GTM-8754	-	Other	Prevent odd ASYNCIO deadlock
GTM-8767	-	Admin	🔴 MUPIP SET -HARD_SPIN_COUNT and -SPIN_SLEEP_MASK support 🟢
GTM-8769	-	Language	🔴 Syntax check \$ETRAP, \$ZSTEP, \$ZTRAP, and EXCEPTION when specified 🟢
GTM-8779	-	Admin	Freeze Notification
GTM-8780	-	Language	Fix \$SELECT() handling of certain syntax errors
GTM-8781	-	Other	Prevent memory leak in ZSYSTEM
GTM-8786	-	Language	🔴 \$NAME() of a naked reference returns any current extended reference
GTM-8787	-	Admin	MUPIP JOURNAL -EXTRACT='-stdout' doesn't explode at termination if stdout is gone
GTM-8788	-	Language	The compiler excludes BLKTODEEP lines from the object files
GTM-8789	-	Language	Prevent NEW \$ZGBLDIR from setting up an Update Process failure
GTM-8790	-	DB	Retain any extended first reference in \$REFERENCE when sharing statistics
GTM-8792	C9I01-002944	Language	🔴 Prevent keys that exceed the supported maximum string length

Id	Prior Id	Category	Summary
GTM-8794	-	Admin	🔴 MUPIP RUNDOWN -OVERRIDE works on a non-MUPIP backup made during an Instance Freeze
GTM-8795	-	DB	Journal Updates promptly during MUPIP FREEZE -ONLINE
GTM-8796	-	DB	Improved error handling during TP and mini transaction commits
GTM-8797	-	Admin	Installation script fixes
GTM-8798	-	Admin	MUPIP ENDIANCVT converts Mutex Queue Slots
GTM-8799	-	Other	Improve performance for a pattern of local variable creation
GTM-8801	-	Other	cmake build produces appropriate support for the ^%YGBLSTATS utility.
GTM-8804	-	Language	ZSHOW "T" option to return summary for ZSHOW "GL" 🟢
GTM-8805	-	DB	Fix to havesting of LOCKs abandoned by an abnormally terminated process
GTM-8832	-	Language	Appropriately report NUMOFLOW for string literal with a huge value when used as a number
GTM-8839	-	Language	\$DEVICE shows the full error message 🟢
GTM-8840	-	Admin	🔴 Normalized gtmsecshr message severities
GTM-8842	-	Admin	ZBREAK and ZSTEP restricted in triggers when TRIGGER_MOD is restricted 🟢
GTM-8844	-	Admin	🔴 Restriction available for HALT and ZHALT; ZGOTO 0 can return a non-zero status to the shell 🟢
GTM-8846	-	Admin	GT.M accepts multi-slash journal file names
GTM-8847	-	Language	Provide a way to detect and limit process private heap storage 🟢
GTM-8849	-	Other	Help databases built with make files have QDBRUNDOWN and NOGVSTATS characteristics

Id	Prior Id	Category	Summary
GTM-8850	-	DB	Allow process exit when MUPIP FREEZE - ONLINE is in place
GTM-8854	-	Language	Compiler handles a syntax error after a literal postconditional that's FALSE
GTM-8855	-	Other	Prevent memory leak from an error locating a global directory
GTM-8856	-	Language	Defer failing evaluations of literal pattern matches to run time
GTM-8857	-	Language	Improve error detection for certain pattern match cases
GTM-8858	-	DB	Improve available information in cases of apparent database integrity issues
GTM-8866	-	Language	Prevent timeouts with more than three decimal digits from being too long
GTM-8873	-	DB	Prevent occasional \$ORDER(,-1) problem

Database

- * GT.M allows updating globals belonging to a different source instance using extended global references or SET \$ZGBLDIR. While the replication setup remains the same, these are the main considerations:
 1. Use one of two ways to identify the current instance as specified by a replication instance file:
 - a. A global directory can define a mapping to a replication instance file as specified with a GDE CHANGE -INSTANCE -FILE_NAME=<replication_instance_file> command. When a global directory is use, if it has a mapping of an instance file, that mapping overrides any setting of the gtm_repl_instance environment variable. GDE CHANGE -INSTANCE -FILE_NAME="" removes any global directory mapping for an instance file.
 - b. The gtm_repl_instance environment variable specifies a replication instance file for utilities, and, as the default, whenever a user processes relies on a global directory with no instance file specification.
 2. In order to use multiple instances, at least one global directory must have an instance mapping.
 3. A replication instance file cannot share any region with another instance file.
 4. The Source Servers of all the instances have properly set up Replication Journal Pools.
 5. A TP transaction or a trigger, as it always executes within a TP transaction, must always restrict updates to globals in one replicating instance.



Notes

- * Like other mapping specified by a global directory, a process determines any instance mapping by a global directory at the time a process first uses the global directory. Processes other than MUPIP CREATE ignore other (non-mapping) global directory database characteristics, except for collation, which interacts with mapping.
- * When Instance Freeze is enabled (gtm_custom_errors is appropriately defined), a process attaches a region to an instance at the first access to the region; the access may be a read or a VIEW/\$VIEW(). Otherwise, the process attaches to a region at the first update to that region. When the mappings are correct, this difference does not matter.
- * A process can always update globals that are not in a replicated region.
- * Use \$VIEW("JNLPOOL") to determine the state of the current Journal Pool. \$VIEW("JNLPOOL") returns the replication instance file name for the current Journal Pool and an empty string when there is no Journal Pool. Note that the

current Journal Pool may not be associated with the last global accessed by an extended reference.

Example:

An EHR application uses a BC replication configuration (A->B) to provide continuous availability. There are two data warehouses for billing information and medical history. For research purposes, the data in these medical history warehouses is cleansed of patient identifiers. Two SI replication instances (P->Q) are setup for the two data warehouses.

The primary global directory (specified via the environment variable gtmgbldir) includes the regions needed for the application proper. It may have the instance file as specified in the global directory or via the environment variable gtm_repl_instance. Each warehouse instance would have its own global directory (e.g. p.gld and q.gld). These global directories have an instance file specified with GDE CHANGE -INSTANCE -FILE_NAME=<replication_instance_file>.

Such a replication setup may benefit from this facility in the following ways:

1. A trigger on the primary database A uses normal global references to update a staging global (say ^%BACKLOG) in a non-replicated region of A to store information meant for the warehouses. At an appropriate time, a separate batch process runs across the ^%BACKLOG staging global and applies updates using extended references to P or Q using a transaction or non-TP. If the transaction succeeds, the process removes the applied updates from ^%BACKLOG. Locks control access to ^%BACKLOG and enforce the serialization of updates to P

OR

2. The application does not use triggers but updates a global on A in a transaction. If the transaction succeeds, the application starts two more transactions for the warehouses. The second transaction uses extended references to update P. If it fails, the application updates ^%BACKLOG("P") on a non-replicated region of A. The third transaction uses extended references to update Q. If it fails, the application updates ^%BACKLOG("Q") on a non-replicated region of A. A batch process runs periodically to apply updates from ^%BACKLOG to P and Q using TP or non-TP and remove updates that have been applied. This batch process uses LOCKs to control access and enforce serialization of updates to P and Q.

Because this functionality has a wide variety of user stories (use cases) and has substantial complexity, although the code appears robust, we are not confident that we have exercised a sufficient breadth of use cases in our testing. Also, we may make changes in future releases that are not entirely backwards compatible. We encourage you to use this facility in development and testing, and to provide us with feedback. If you are an FIS customer and wish to use this in production, please contact us beforehand to discuss your use case(s). (GTM-8182) 🟢

- * GT.M LOCK commands perform better with large numbers of locks, and particularly with large numbers of processes acquiring the locks. Previously processes acquiring locks could encounter significant slowdown and lock timeouts as the number of locks and competing processes increased. This change requires additional memory per lock slot, so administrators should monitor lock slots (LKE SHOW) to determine if they need to increase lock space needs.

The GDE -LOCK_SPACE segment qualifier and MUPIP SET -LOCK_SPACE qualifier accept a maximum value of 262144 pages. Previously the maximum value was 65536 pages.

LKE SHOW includes a LOCKSPACEINFO message in its output for regions with a BG or MM access method. This message provides additional information on the use of LOCK space. Previously LKE only issued this message when the lock space was exhausted.

(GTM-8680) 🚫

- * When the first reference to a database for which a process has statistics sharing enabled is an extended reference, \$REFERENCE maintains the extended reference. A regression associated with the implementation of statistics sharing in V6.3-001[A] caused this unusual case to lose that information. This was only ever observed in the GT.M development environment and has never been reported from a customer site. (GTM-8790)
- * GT.M keeps journal files up to date while a MUPIP FREEZE -ONLINE is in place. Previously the journal files would only be updated when there was a large amount of journal activity or the freeze was removed. (GTM-8795)
- * GT.M correctly handles any errors in the middle of a transaction commit. In GT.M V6.3-002, due to a regression introduced by GTM-8436, it was possible in very rare scenarios for a critical section deadlock. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8796)
- * GT.M manages LOCK concurrency correctly when checking for abandoned LOCKs. In V6.3-002 it could prematurely or belatedly determine that a LOCK was abandoned. (GTM-8805)
- * GT.M processes detach from database files correctly when a FREEZE -ONLINE is in place. Previously a process could hang waiting on a critical resource while trying to detach, which typically occurs when the process is trying to exit. (GTM-8850)
- * Improve available information in cases of apparent database integrity issues. (GTM-8858)
- * GT.M properly handles retries involving \$ORDER(gvn,-1) or \$ZPREVIOUS(gvn) functions. Previously, with certain key combinations, the retry processing could overflow a buffer, leading to memory corruption. The workaround was for any process using \$\$ORDER(-1) in a region to previously have used a maximum length key for the region. (GTM-8873)
- * **V6.3-003B** GT.M handles out-of-lock-space conditions more gracefully. Previously a full lock table could result in corruption of the lock structures, leading to segmentation violations (SIG-11). (GTM-9020)
- * **V6.3-003B** This basically reverts GTM-8680 as the performance improvements were unreliable in circumstances reported by a customer, resulting in multiple processes occasionally holding the same LOCK. The GDE -LOCK_SPACE segment qualifier and MUPIP SET -LOCK_SPACE qualifier accept a maximum value of 65536 pages. In V6.3-003[A] the maximum value was 262144 pages. LKE SHOW does not include a LOCKSPACEINFO message in its output for regions with a BG or MM

Database

access method as it did in V6.3-003[A]; GT.M only issues this when the application exhausts the LOCK space. (GTM-9023)

Language

- * Addressed by GTM-8694 (GTM-6115)
- * When GT.M encounters a line with a length greater than 8192 bytes in a source file, it emits a %GTM-W-LSINSERTED warning. This warning identifies cases where a line greater than 8192 bytes is split into multiple lines, which causes statements beyond the character prior to the limit to execute irrespective of any starting IF, ELSE or FOR commands. Previously, GT.M split the lines with no warning. (GTM-7986)
- * GT.M accepts an offset without a label for an entryref argument to DO, GOTO and ZGOTO. FIS recommends restricting the use of offsets in entryrefs to debugging, error handling and testing. Previously GT.M required a label before any offset. (GTM-8186) 🟢
- * GT.M sets \$KEY to the characters terminating a READ, and NULL if terminated otherwise (e.g. FIX format, end of file, or timeout). When it encounters an error during an I/O, GT.M sets \$DEVICE to "1," followed by an error description. Previously, GT.M did not maintain \$KEY for sequential devices and only maintained \$DEVICE for certain I/O errors. (GTM-8587) 🟢
- * GT.M checks the syntax of code assigned to \$ETRAP, \$ZSTEP, \$ZTRAP, and EXCEPTION at the time they are specified. Note that \$ZTRAP and EXCEPTION are subject to gtm_ztrap_form, and, if that specifies entryref or adaptive, GT.M does not check the syntax. Also, the environment variables \$gtm_etrp, \$gtm_trigger_etrp, and \$gtm_zstep provide ways of setting some of the ISVs, so their values are verified at process initiation. Further, a SET \$ETRAP uses a temporary default value of "IF \$ZJOBEXAM" when shifting from \$ZTRAP to \$ETRAP in case the specified value has compilation errors. Previously GT.M detected errors in such code only for SET \$ZSTEP and when attempting to use the vector. (GTM-8769) 🟡 🟢
- * \$SELECT() compilation properly handles special cases where an omitted colon after a literal true select argument produces a syntax error; a regression introduced in V6.3-001[A], caused it to produce a GTMASSERT2 after reporting the issue. (GTM-8780)
- * \$NAME() of a naked reference returns any extended reference associated with the current \$REFERENCE; previously it did not. (GTM-8786) 🟡
- * The compiler excludes BLKTODEEP lines from the object files; due to a regression introduced by GTM-5178 in V6.3-002 they were not excluded (GTM-8788)
- * The Update Process operates correctly when a trigger issues a NEW \$ZGBLDIR while performing updates on other unreplicated instances. A regression introduced with GTM-4759 in V63000[A] caused such operations in the Update Process to terminate unexpectedly with a segmentation fault (SIG-11). (GTM-8789)
- * \$QUERY() of a local variable produces a MAXSTRLEN error when its result exceeds the supported string length. While GT.M supports very long key lengths for local variables, features that need to work with the entire key, such as \$NAME() and \$QUERY(), may not be able to handle keys with a length that exceeds the maximum supported string length (currently 1MiB). Rather than prohibit

Language

longer keys entirely, GT.M just restricts such features, so, if you need the features, avoid keys that exceed the limit. Previously \$QUERY() could give a segmentation violation (SIG-11) if it encountered an over-length key. (GTM-6115)(GTM-8792) 🍏

- * The ZSHOW "T" (where "T" can be case-insensitive) produces only the summary lines for "G" and "L" output; previously ZSHOW always showed the detail with the summary. (GTM-8804) 🍏
- * GT.M reports a NUMOFLOW error for a string literal used as a number and evaluating to a number that exceeds the supported range, as of this writing: 1E47. A compiler optimization in V6.3-001[A] caused such an evaluation to produce a very very small negative value. (GTM-8832)
- * \$DEVICE returns the complete error message. Previously, \$DEVICE truncated messages which were over 80 characters. (GTM-8839) 🍏
- * The read/write (non-NEWable) \$ZSTRPLIM ISV provides a way for a process to limit its process private memory used for local variable and scratch storage. When the value is zero (0), the default, or negative, there is no limit. A positive value specifies a byte limit. When a request for additional memory exceeds the limit, GT.M does the expansion, and then produces an STPCRIT error. By default, a later request for memory produces an STPOFLOW, unless, subsequent to STPCRIT, \$ZSTRPLIM has been set to the same or higher limit. Note that GT.M allocates memory in large blocks so the interaction of \$ZSTRPLIM with memory growth is not exact. When the gtm_string_pool_limit environment variable specifies a positive value, GT.M uses it for the initial value of \$ZSTRPLIM. Previously, process memory was only limited by operating system configuration. (GTM-8847) 🍏
- * The compiler appropriately handles a syntax error in the argument of a postconditional command when the postconditional is a literal that evaluates to FALSE. Due to a regression associated with GTM-8573 in V6.3-001[A], this combination caused an abnormal termination with a segmentation violation (SIG-11). (GTM-8854)
- * GT.M defers literal optimizations involving patterns within an XECUTE as well as evaluations that encounter issues with the pattern table. Due to a regression associated with GTM-8573 in V6.3-001[A], these combinations caused an abnormal termination with a segmentation violation (SIG-11). (GTM-8856)
- * Pattern code processing appropriately produces a PATMAXLEN error for certain patterns that exceed the size GT.M supports. Previously, some patterns produced a segmentation violation (SIG-11). This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8857)
- * GT.M appropriately handles timeout values which have more than three decimal digits; in V6.3-002 and V6.3-003, such values inappropriately had a very long timeout. The workaround was to avoid such values because GT.M only recognizes three digits after the decimal point for timeouts. (GTM-8866)
- * **V6.3-003A** When using standard Boolean evaluation (no short-circuiting enabled by \$gtm_boolean or gtm_side_effects) GT.M deals appropriately with cases where there is a side effect and a right-hand operand interior to the expression happens to evaluate to a value that causes

an incorrect result. This issue appeared with the introduction of standard Boolean evaluation in V5.5-000, and has not previously shown up in testing or been reported until a customer encountered a case. (GTM-8880)

System Administration

- * MUPIP BACKUP for directory and file name lengths equal and greater than 255, issues FILENAMETOOLONG error; previously, this produced a core. Also, backing up an Instance File to a path longer than 255 succeeds with the correct journal sequence number; previously, this issued an incorrect journal sequence number. In addition, MUPIP JOURNAL -RECOVER -REDIRECT issues more information for the INVREDIRQUAL error; previously, it provided less context for the INVREDIRQUAL error. (GTM-4212)
- * The MUPIP SET command supports the following qualifiers: -N[ULL_SUBSCRIPTS]={never, always, existing}, which controls whether GT.M accepts null subscripts for database keys. -[NO]STD[NULLCOLL], which determines whether GT.M will use standard MUMPS collation or GT.M collation for null-subscripted keys. Previously, this functionality was only available through GDE for database file creation, and DSE for existing database files. FIS strongly recommends avoiding the use of DSE when there is an alternative. (GTM-8617) 🟢
- * MUPIP REPLICATE -RECEIVER -LOG_INTERVAL= and MUPIP SET -DEFER_TIME= accept values ranging from 0 to 2**31-1, -DEFER_TIME= accepts one special value -1; otherwise they produce an error message. MUPIP REPLICATE -RECEIVER -HELPER accepts values ranging from 1 to 128 and otherwise produces an error message. Previously, all these operations accepted inappropriate values. (GTM-8732)
- * MUPIP SET -{FILE|REGION} recognizes the -[NO]READ_ONLY qualifier to indicate whether GT.M should treat an MM access method segment as read only for all users, including root. This designation augments UNIX authorizations and prevents any state updates that normally might require an operational action for a database with no current accessing (attached) processes. The GT.M help databases have -READ_ONLY set by default. Previously, a database such as the gtmhelp database in the GT.M distribution typically never received a data update but nevertheless could require a ROLLBACK, RECOVER or RUNDOWN to ensure a proper at-rest state. MUPIP emits an error on attempts to set -READ_ONLY on databases with the BG access method, or to set the access method to BG on databases with -READ_ONLY set. (GTM-8735) 🟢
- * MUPIP SET for file or region accepts -H[ARD_SPIN_COUNT]=<integer count> and -SPIN[_SLEEP_MASK]=<hexadecimal mask>; previously it did not support changes to the hard spin count and required -SPIN_SLEEP_LIMIT to change the spin sleep mask. MUPIP SET no longer supports the -SPIN_SLEEP_LIMIT qualifier. (GTM-8767) 🟡 🟢
- * MUPIP FREEZE sends a DBFREEZEON/DBFREEZEOFF message to the system log for each region whose freeze state is changed. (GTM-8779)
- * MUPIP JOURNAL -EXTRACT='-stdout' appropriately handles its termination; previously, if stdout was already closed this specification produced a segmentation violation (SIG-11). (GTM-8787)
- * Copies of a database file made while a MUPIP FREEZE -ONLINE -ON is in effect can be used on the same system by performing a MUPIP RUNDOWN -OVERRIDE and a MUPIP FREEZE -OFF on the copy. Previously, an attempt to remove the freeze on the copy would attempt to modify the journal files of the original database and fail. Note that this change moved the ^%PEEKBYNAME

item "sgmnt_data.freeze_online" to "node_local.freeze_online"; for this release (V6.3-003) only, ^%PEEKBYNAME recognizes either designation, but going forward, it will not. If you have code referencing this item, please revise it. (GTM-8794) 🚫

- * The gtminstall script has a new command line option to skip disabling RemoveIPC=Yes in systemd configuration files. This option was added to facilitate unattended installs and automated builds. Previously, if no response was provided, the script would terminate with an error in the script. This issue was only observed in the GT.M development environment, and was never reported by a user.

The configure script better handles the detection of 64 bit software. Previously, the script could mistakenly identify 32bit software as 64bit software if the output of the file command contained "64" in the sha1 binary hash. This issue was only observed in the GT.M development environment, and was never reported by a user.

Additionally the gtminstall script defaults to i586 kit for i686 platforms. Since GT.M V6.2-001, the GT.M release distribution kit has i586 in the name. Attempting to use gtminstall on an i686 platform resulted in a failure to download and install the distribution kit due to this change in name. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8797)

- * MUIP ENDIANCVT converts all numeric file header fields to the opposite endian. Previously, it did not convert the Mutex Queue Slots field. (GTM-8798)
- * gtmsecshr, and facilities that interact with it use message severities that seem appropriate to the issue. Previously, we received customer concerns that the severities were arbitrary, which complicated understanding them. Note that, should you use message parsing that depends on severity, you should review it for possible impact. (GTM-8840) 🚫
- * When TRIGGER_MOD is restricted, attempting to ZBREAK a trigger results in a RESTRICTEDOP error, and both ZBREAK and ZSTEP actions are ignored while executing code within a trigger. Previously a TRIGGER_MOD restriction did not imply these other restrictions. (GTM-8842) ✅
- * The GT.M restrictions facility recognizes HALT[:<group-name>] and ZHALT[:<group-name>]. When either is present and restriction conditions are met, the restricted command produces a RESTRICTEDOP error. In order to limit pathological looping, if A GT.M process issues a second occurrence of the restricted command within half a second, it terminates after sending a fatal error to both the principal device and the syslog, and also producing a GTM_FATAL* context file, but no core file. Note that, With these restrictions in place, a process should terminate with, for example: ZGOTO 0. with or without a restriction, executing these commands as part triggered logic on a replicating instance may cause the Update Server to terminate and thereby stop replication. As part of this change when ""=\$ZTRAP and ""!=\$ECODE, ZGOTO 0 returns a non-zero status, derived from the error code in \$ZSTATUS, to the shell. If you have an application that uses ZHALT, ZGOTO 0 and shell scripts that check returned status, you should review things in light of this change. Note: with appropriate error handling, an application can use one or both of these restrictions to perform clean up after any explicit HALT or ZHALT. Previously, the restrictions facility did not support these two restrictions, and \$ZGOTO 0 always returned a success status to the shell. (GTM-8844) 🚫 ✅
- * GT.M appropriately uses file paths with multiple adjacent forward slashes (/) when turning journaling on. Previously, when turning journaling on, GT.M appended inappropriate characters

to the end of intended journal file names whose path contained adjacent forward slashes. The workaround was to avoid specifying file paths with any adjacent forward slashes. (GTM-8846)

Other

- * GT.M defers interrupts during asynchronous database writes. Previously, such interrupts could very occasionally cause a deadlock. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8754)
- * ZSYSTEM manages memory appropriately; a regression in V6.3-002 caused it to leak small amounts of memory. This issue was only observed in the GT.M development environment, and was not reported by a user. (GTM-8781)
- * GT.M handles certain unusual cases of local storage (heap) utilization more efficiently. Previously, these cases would cause poor performance as local when adding variables with such patterns. (GTM-8799)
- * The cmake build produces appropriate support for the ^%YGBLSTATS utility; in the original V6.3-001, V6.3-001A and V6.3-002 releases, an attempt to use ^%YGBLSTATS with a cmake build produced DLLNORTN and ZCRTENOTF errors.(GTM-8801)
- * Help databases built with make files have QDBRUNDOWN and NOGVSTATS characteristics, which match the properties of help databases of the release builds. Previously, these characteristics differed depending on the build. (GTM-8849)
- * GT.M correctly cleans up buffers which were allocated prior to a runtime error due to a missing global directory. Previously, these buffers would accumulate if an error handler prevented GT.M from terminating. The workaround for this was to avoid repeated attempts to use a global directory that does not exist or to which the process does not have an authorized path. (GTM-8855)
- * **V6.3-003A** GT.M handles rare cases in timer handling correctly. Previously these cases could result in SETTIMERFAILED errors and messages in the system log. This was only ever observed in the GT.M development environment and has never been reported from a customer site. (GTM-8887)
- * **V6.3-003A** The ZHELP command does not report errors after the user presses a <CTRL-C>. Previously, when exiting after a <CTRLC>, the utility reported an UNDEF error and left a GT.M help dump file for analysis. (GTM-8889)

Error and Other Messages

DBFREEZE^{OFF}

DBFREEZE^{OFF}, Region rrrr is UNFROZEN ([NO]OVERRIDE [NO]AUTOREL)

Operator log/MUPIP Information: The database region rrrr is no longer frozen, most likely due to a MUPIP FREEZE -OFF, with the selected options. [NO]AUTOREL indicates whether an autorelease of the region occurred prior to the MUPIP FREEZE -OFF command.

Action: Confirm that this was the desired action.

DBFREEZE^{ON}

DBFREEZE^{ON}, Region rrrr is FROZEN ([NO]OVERRIDE [NO]ONLINE [NO]AUTOREL)

Operator log/MUPIP Information: The database region rrrr is frozen, most likely due to a MUPIP FREEZE -ON, with the reported options.

Action: Confirm that this was the desired action.

DBNONUMSUBS

DBNONUMSUBS, XXXX Key contains a numeric form of subscript in a global defined to collate all subscripts as strings

Run Time/MUPIP Error: The record has a numeric subscript but the collation setting for the global or region indicates all subscripts are filed as strings. The leading context (XXXX) identifies the block and offset of the problematic record. This can arise if an operator uses DSE to force a change to a collation setting or to modify a key when the global already has content.

Action: If you can determine the cause of, and reason for, the change and you may choose to reverse it. If you need to change the collation, the appropriate procedure is to EXTRACT the data, KILL the global, or remove and recreate the database file, and then LOAD the extracted data.

DBNULCOL

DBNULCOL, XXXX NULL collation representation differs from the database file header setting

DSE/MUPIP/Run Time Error: This indicates the database contains a record with an empty subscript ("Null" subscript) representation that is incompatible with the current setting database file header setting for such a representation. The leading context (XXXX) specifies the block number and offset of the problematic record. This can only arise if someone changes the setting for the database while it contains one or more such subscripts. FIS recommends against making such a change. This message can originate from MUPIP INTEG, DSE INTEG or from running with VIEW "GDSCERT"

Action: Use the record and block information to remove the problematic record with DSE and restore the data appropriately, typically with a SET command. Note that the record and block of the record may change due to ongoing updates, so this operation requires great care and familiarity with DSE.

GBLOFLOW

GBLOFLOW, Database segment is full

Run Time/MUPIP Error: This indicates that an error was encountered while extending the database file.

Action: Examine the accompanying message(s) for the cause of the error. If the error is due to insufficient authorization, address that. If the error is due to TOTALBLKMAX (refer to the explanation of that message) or a lack of enough free space on the disk to fit the size of a database file, try performing a KILL of some nodes in the database to get free blocks in the existing allocated space (you may need to KILL several subscripted nodes before you can KILL a name node).

LSINSERTED

LSINSERTED, Line YYYY, source module XXXX exceeds maximum source line length; line separator inserted, terminating scope of any prior IF, ELSE, or FOR

Compile Time Warning: Indicates that source XXXX line YYYY exceeded the maximum line length and GT.M separated it into multiple lines to allow continued parsing. Internally, GT.M represents the generated code as N lines for this source line, where N is the number of segments extracted from this source line. Be aware that as a result of this, source lines containing a command whose scope is rest of the line (IF, ELSE, FOR), are now split into multiple lines, each with a separate scope.

Action: Consider refactoring code to avoid source line lengths in excess of 8192 characters.

MUTEXFRCDTERM

MUTEXFRCDTERM, Mutual Exclusion subsystem detected forced termination of process pppp. Crit salvaged from database file dddd.

Run Time Warning: This indicates that GT.M confirmed inappropriate termination of the process pppp, while holding crit on database file dddd.

Action: Determine the cause of the termination and take appropriate action.

NULSUBSC

NULSUBSC, XXXX Null subscripts are not allowed for current region

Run Time/MUPIP Error: This indicates that a global variable specified a null subscript in a database file which does not accept null subscripts. The leading context (XXXX) specifies more about the event or location of the issue

Action: Look for the source of the null subscript(s) and consider whether they are appropriate or due to a coding error. If they are appropriate, use MUPIP SET -NULL_SUBSCRIPTS, and remember to make the same adjustment with GDE CHANGE REGION -NULL_SUBSCRIPTS to ensure the next time you recreate a database that the characteristic persists.

READONLYNOBG

READONLYNOBG, Read-only cannot be enabled on non-MM databases

MUPIP Error: This indicates an attempt to change a BG database to -READ_ONLY or to change a -READ_ONLY to MM access method; -READ_ONLY only compatible with the MM access mode.

Action: Verify whether the database should not be read only and adjust, if appropriate. Alternatively, set the database to MM access mode then mark it as read-only.

REPLINSTACC

REPLINSTACC, Error accessing replication instance file xxxx

Run Time/MUPIP Error: This indicates that some errors were encountered while accessing the specified replication instance file defined by \$gtm_repl_instance or the relevant global directory.

Action: Refer to the accompanying message(s) for additional information.

REPLINSTMISMTCH

REPLINSTMISMTCH, Process has replication instance file ffff (jnlpool shmids = ssss) open but database dddd is bound to instance file gggg (jnlpool shmids = tttt)

Run Time Error: The process attempted an update on the replicated database dddd associated with the replication instance file ffff and journal pool shared memory id ssss; however, the process has already associated the database with a different replication instance file gggg or journal pool shmids tttt.

Action: A replicated database can only accept updates by processes that have the same replication instance file (defined by the environment variable gtm_repl_instance or in the global directory) open for that database. Ensure the same replication instance file is used for all processes that update the same replicated database file. This error can also occur if the replication instance file was recreated (while processes were still accessing the replication instance). In this case, the name ffff and gggg would be the same but the corresponding journal pool shared memory ids would be different. To recover from this situation, shut down all processes accessing the instance from before and after the instance file recreate. Run an argumentless MUPIP RUNDOWN to clean up the older journal pool tttt and restart the instance. The Source Server (which is the first process to start on a replicated instance) only binds replicated databases from its global directory to the journal pool that it creates. No other replicated database file can be bound with this journal pool.

REPLMULTINSTUPDATE +

REPLMULTINSTUPDATE, Previous updates in the current transaction are to xxxx so updates to yyyy (in rrrr) not allowed

Run Time Error: Previous updates in the current TP transaction mapped to database files associated with replication instance file xxxx, so it cannot make updates to database file yyyy which is associated with replication instance file rrrr.

Action: Modify the application so all updates in a TP transaction to replicated regions are associated with a single replication instance.

STACKCRIT ⚠

STACKCRIT, Stack space critical

Run Time Error: This indicates that the process has consumed almost all of the available stack space.

Action: Look for infinite recursion. If you do not take immediate action to reduce your stack, GT.M is likely to produce a STACKOFLOW error, which terminates the process. Examine the stack with ZSHOW. Trim the stack using QUIT, ZGOTO, HALT or ZHALT.

STACKOFLOW ⚠

STACKOFLOW, Stack overflow

Run Time Fatal: This indicates that the process required more stack space than was available in memory.

Action: Reduce the stack when you get a STACKCRIT error. This error terminates the process.

STPCRIT +

STPCRIT, String pool space critical

Run Time Error: This indicates that the process has exceeded the heap (string pool) limit specified in the \$ZSTRPLIM ISV. If you do not take prompt action to reduce the process memory requirements, at the next heap expansion, GTM produces an STPOFLOW error, which terminates the process.

Action: Investigate whether the process memory usage is appropriate, and if so, increase or remove the limit. Otherwise correct the cause(s) of the excessive memory consumption. Please see the documentation for \$ZSTRPLIM for additional information.

STPOFLOW +

STPOFLOW, String pool space overflow

Run Time Fatal: This indicates that the process has previously exceeded the heap (string pool) limit specified in the \$ZSTRPLLIM ISV and still needs more memory, so GTM terminates the process.

Action: Investigate whether the process memory usage is appropriate, and if so, increase or remove the limit. Otherwise correct the cause(s) of the excessive memory consumption. Please see the documentation for \$ZSTRPLLIM for additional information.

