

GT.M

Release Notes

V6.3-004

Empowering
the Financial World

FIS

Contact Information

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

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

Legal Notice

Copyright ©2018-2019 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 1.2	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 1.1	19 June 2018	Made corrections for GTM-8874 and GTM-1042.
Revision 1.0	28 March 2018	V6.3-004

Table of Contents

V6.3-004	1
Overview	1
Conventions	1
Platforms	2
Platform support lifecycle	4
32- vs. 64-bit platforms	4
Call-ins and External Calls	4
Internationalization (Collation)	5
Environment Translation	5
Additional Installation Instructions	5
.....	5
Upgrading to GT.M V6.3-004	7
Stage 1: Global Directory Upgrade	7
Stage 2: Database Files Upgrade	8
Stage 3: Replication Instance File Upgrade	10
Stage 4: Journal Files Upgrade	10
Stage 5: Trigger Definitions Upgrade	10
Downgrading to V5 or V4	11
Managing M mode and UTF-8 mode	12
Setting the environment variable TERM	13
Installing Compression Libraries	13
Change History	14
V6.3-004	14
Database	17
Language	18
System Administration	19
Other	20
Error and Other Messages	21
CRYPTJNLMISMATCH 	21
ENCRYPTCONFLT2 	21
INVSEQNOQUAL 	21
JNLACCESS 	21
JNLBADRECFMT 	21
JNLCYCLE 	22
JNLDBERR 	22
JNLDISABLE 	22
JNLEXTEND 	22
JNLEXTR 	22
JNLEXTRCTSEQNO 	22
JNLFILOPN 	23
JNLFLUSHNOPROG 	23
JNLFSYNCLSTCK 	23
JNLINVALID 	23
JNLNAMLEN 	23
JNLNOCREATE 	24
JNLORDBFLU 	24
JNLREADEOF 	24
JNLRECFMT 	24

JNLSPACELOW 	24
JNLSWITCHSZCHG 	24
JNLSWITCHTOOSM 	25
JNLWRERR 	25
MSTACKSZNA 	25
PREMATEOF 	25
SYSUTILCONF 	25

V6.3-004

Overview

V6.3-004 represents the first of our effort to provide releases as more frequent smaller delta. 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.	replicating instance.	-updnok

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
	-propagateprimary for propagating 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.	
N/A	N/A	supplementary instance. For example, in an A->P->Q replication configuration, P is the supplementary instance. Both A and P are originating instances.	-updok

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

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

Platforms

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

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

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

Platform	Supported Versions	Notes
IBM Power Systems AIX	7.1 TL 4, 7.2	Only 64-bit versions of AIX with POWER7 as the minimum required CPU architecture level are Supported.

Platform	Supported Versions	Notes
		<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 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</p>

Platform	Supported Versions	Notes
		for d2dc317d564a46dfc683978a2e5a4f91434e9711 at https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3). The Red Hat Bugzilla identifier for the bug is 1213487. With NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs.
x86 GNU/Linux	Debian 9 (Stretch)	This 32-bit version of GT.M runs on either 32- or 64-bit x86 platforms; we expect the x86_64 GNU/Linux version of GT.M to be preferable on 64-bit hardware. Running a 32-bit GT.M on a 64-bit GNU/Linux requires 32-bit libraries to be installed. The CPU must have an instruction set equivalent to 586 (Pentium) or better. Please also refer to the notes above on x86_64 GNU/Linux.

Platform support lifecycle

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

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

32- vs. 64-bit platforms

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

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

Call-ins and External Calls

Parameter type	32-Bit	64-bit	Remarks
gtm_long_t	4-byte (32-bit)	8-byte (64-bit)	gtm_long_t is much the same as the C language long type.
gtm_ulong_t	4-byte	8-byte	gtm_ulong_t is much the same as the C language unsigned long type.
gtm_int_t	4-byte	4-byte	gtm_int_t has 32-bit length on all platforms.
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
<code>gtm_descriptor</code> in <code>gtm_descript.h</code>	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
<code>gtm_string_t</code> type in <code>gtmxc_types.h</code>	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 V6.3-004 in a Filesystem Hierarchy Standard compliant location such as `/usr/lib/fis-gtm/V6.3-004_arch` (for example, `/usr/lib/fis-gtm/V6.3-004_x86` on 32-bit Linux systems). A location such as `/opt/fis-gtm/`

V6.3-004_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 V6.3-004

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 V6.3-004 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 V6.3-004 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 V6.3-004.
- * 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 V6.3-004.
- * Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

Stage 2: Database Files Upgrade

To upgrade from GT.M V6*:

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

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

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

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



Important

A V5 database that has been automatically upgraded to V6 can perform all GT.M V6.3-004 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

V6.3-004 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.3-004 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 V6.3-004 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 V6.3-004 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 V6.3-004 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/V6.3-004.

To extract and reapply the trigger definitions on V6.3-004 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 V6.3-004, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.3-004 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 V6.3-004.
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 V6.3-004 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_V6.3-004_i686`, then `gtmprofile` sets `$gtm_dist` to `/usr/lib/fis-gtm/gtm_V6.3-004_i686/utf8`).
 - * On platforms where the object files have not been placed in a `libgtmutil.so` shared library, the last element of `$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.

Change History

V6.3-004

Fixes and enhancements specific to V6.3-004:

Id	Prior Id	Category	Summary
GTM-1042	S9712-000627	Other	The gtm_mstack environment variable can control the M stack size ✔
GTM-3146	C9A06-001514	Other	✔ Protect against alias and path settings subverting GT.M calls to POSIX interfaces ✔
GTM-5374	S9E04-002453	Admin	Please see GTM-5730
GTM-5730	S9F07-002559	Admin	✔ Update Process logs show record type descriptions ✔
GTM-6747	C9L02-003361	Admin	MUPIP SET -JOURNAL does not adjust sequence numbers on replicated regions ✔
GTM-7483	-	Other	Improve message from a MUPIP INTEG when a Directory Tree issue is detected
GTM-7872	-	Admin	Improve message from a MUPIP INTEG directed to a non-existent database file
GTM-7915	-	DB	Do not let ^#t record size or keysize influence trigger installation success/failure ✔
GTM-8202	-	Admin	Journal extract for a particular sequence number ✔
GTM-8576	-	Admin	✔ A Source Server logs errors to its log file rather than stderr ✔
GTM-8643	-	Language	SOCKET device listen queue depth left to OS configuration; better retry management for local connects ✔
GTM-8699	-	Language	Optional region argument for \$VIEW("STATSHARE") ✔
GTM-8752	-	Other	gtmpcat doesn't require /etc/issue
GTM-8777	-	Other	%GCE, %GSE, %RCE, and %RSE support QUIET entrypoint. %RCE and %RSE support QCALL entrypoint.
GTM-8791	-	Other	Prevent certain control character input to LKE from causing a SIG-11

Change History

Id	Prior Id	Category	Summary
GTM-8859	-	Admin	Correct turn-around point calculation for MUPIP JOURNAL -ROLLBACK when dealing with "idle" regions
GTM-8860	-	Admin	Prevent multiple slashes (/) from appearing in MUPIP JOURNAL -EXTRACT output
GTM-8870	-	Other	Clean up some rough edges
GTM-8874	-	Language	🔴 VIEW "[NO]STATSHARE accepts an optional region-list 🟢
GTM-8883	-	DB	Online Freeze/Journal switch cleanup
GTM-8891	-	Language	Prevent process failure from a certain pattern of \$SELECT() errors
GTM-8893	-	Language	🔴 CTRAP takes precedence over CENABLE (as documented)
GTM-8894	-	Language	\$ZRELDAT ISV provides the UTC data and time of the GT.M build 🟢
GTM-8895	-	Other	%PEEKBYNAME protects the variable from inappropriate modification
GTM-8899	-	Other	Work-around for bugs in the GnuPG agent 🟢
GTM-8900	-	Admin	🔴 MUPIP SET -NOENCRYPTABLE works without valid encryption setup 🟢
GTM-8903	-	Language	Prevent process failure from a certain pattern of \$SELECT() using a lead-in literal and a later global
GTM-8906	-	Admin	MUPIP JOURNAL -ROLLBACK and -RECOVER handle a larger number of records
GTM-8908	-	Other	gtmpcat deals with a symlink for \$gtm_dist
GTM-8909	-	Other	Online help does not report an inappropriate error when exiting after the user typed <CTRL-C>
GTM-8914	-	Language	🔴 \$VIEW("GVSTATS"), ZSHOW "G" and ZSHOW "T" flag results that are probably not current 🟢
GTM-8916	-	Admin	Prevent Receiver Server hang with glibc 2.25 or later
GTM-8919	-	DB	MUPIP REORG -ENCRYPT does not induce CRYPTOPFAILED errors in concurrent processes
GTM-8922	-	Language	VIEW region subarguments can be a list of regions 🟢
GTM-8923	-	Language	UTF-16 READ * and WRITE * Fixes
GTM-8924	-	Language	Fix for unusual conditions with a \$PRINCIPAL SOCKET device for a JOB'd process

Change History

Id	Prior Id	Category	Summary
GTM-8925	-	DB	32-bit Linux uses kernel managed asynchronous database I/O
GTM-8926	-	Other	Flush Timer Deferred During External Call
GTM-8927	-	DB	Prevent inappropriate JNLCTRL errors

Database

- * GT.M limits the key size of a trigger to 1019 bytes. Previously, it limited the key size to the configured limit of regions that used the trigger, which caused a need to increase the maximum key limit and/or change the name of the trigger. This change also allows GT.M to store triggers more compactly in cases where regions had a small record size limit. (GTM-7915) 🟢
- * GT.M handles journal switch errors encountered during a MUPIP FREEZE -ONLINE -ON correctly. Previously it could leave the specified region with a very large Epoch Interval, which could cause a replication source server to issue SRVLCKWT2LNG errors. In addition, GT.M only sends a PREVJNLLINKCUT message to the system log when the links have been cut. Previously the message could be reported in rare cases where a journal switch was deferred, then later switched without cutting links. Also, the source server reports the SRVLCKWT2LNG message correctly. Previously the values for PID and the wait time were reversed and the message reported minutes instead of seconds as the unit of time. (GTM-8883)
- * GT.M processes work correctly when a concurrent MUPIP REORG -ENCRYPT changes the encryption key. Previously GT.M processes could fail with CRYPTOPFAILED errors unaccompanied by other errors explaining the reason for the failure. This issue was only observed in the GT.M development environment, and was never reported by a user. MUPIP RESTORE issues an encryption setup error when attempting to restore an encrypted backup without the correct encryption setup. Previously MUPIP RESTORE issued a CRYPTOPFAILED when the restore instance was not configured for encryption. (GTM-8919)
- * Asynchronous database I/O on 32-bit Linux now uses the kernel to manage the I/Os. Previously, GT.M used the user space glibc version which limited performance. FIS recommends using 64-bit versions for production. [Linux] (GTM-8925)
- * GT.M avoids issuing JNLCNTRL errors inappropriately. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8927)

Language

- * The listen queue depth specified with WRITE /LISTEN for a listening socket is limited only by the limit imposed by the OS; previously, GT.M enforced an artificial limit of 5. A SOCKET device retries connection attempts on local sockets that fail due to possible transient issues on the other end of the attempted connection, (for example an insufficiently large listen queue). Previously such attempts could sometimes appear to succeed, when they actually did not, leading to subsequent errors from READ and WRITE on the socket. (GTM-8643) 
- * VIEW("STATSHARE",<region>) returns TRUE (1) if the process is currently sharing database statistics for the region and FALSE (0) if it is not. For the process to be sharing the database must be enabled for sharing and the process must have opted in to share. \$VIEW("STATSHARE") with no region argument indicates the process has enabled sharing. (GTM-8699) 
- * VIEW "[NO]STATSHARE"[:<region-list>] enables or disables database statistics sharing for listed regions which permit such sharing. Without the region-list, the command acts on all regions enabled for sharing. When a targeted region has sharing disabled, STATSHARE has no effect. Note: a VIEW "[NO]STATSHARE" with no region sub-argument opens any unopened mapped regions and any enabled associated statsDB regions; the \$gtm_statshare environment variable applies to databases as the application first uses them. When the last VIEW "[NO]STATSHARE" had no region sub-argument, regions implicitly share when the process first references them, but after a VIEW specifies selective sharing, regions don't implicitly share as they open. Previously the VIEW command only supported enabling or disabling sharing for all enabled regions. (GTM-8874)  
- * A <side-effect-expression><pure-Boolean-operator>\$SELECT(0:side-effect-expression)) sequence produces a SELECTFALSE run-time error; a regression introduced with the literal optimizations in V6.3-002 caused this combination to produce a SIG-11 (segmentation violation) at compilation. (GTM-8891)
- * GT.M gives CTRAP=3 precedence over CENABLE. This aligns with the documentation, and with design expectation that CENABLE serves as a programmer's tool and CTRAP serves as a means for an application to manage responses to user "interrupts." Due to a change in V6.2-000 the precedence was reversed. (GTM-8893) 
- * The \$ZRELDATE Intrinsic Special Variable provides the UTC date / time of the build GT.M build in the form YYYYMMDD 24:60 (using \$ZDATE() notation). While \$ZVERSION is probably a better identifier for most uses, \$ZRELDATE may be a helpful alternative for those testing pre-release builds. (GTM-8894) 
- * \$SELECT() deals with cases where the first value is evaluated to a literal TRUE (1) and later arguments to the function contain one or more global references. Due to a regression in V6.3-002 associated with compiler optimizations caused this combination to produce a SIG-11 (segmentation violation) at compilation. (GTM-8903)
- * \$VIEW("GVSTATS",<region>), ZSHOW "G" and ZSHOW "T" return a question-mark (?) at the end of their output strings when the process does not have access to the current shared statistics; they did not do this previously. Note that it is possible, although unlikely, the flag character could disrupt code that parses the results of these facilities. (GTM-8914)  
- * VIEW commands accepting a region sub-argument also accept a comma (,) delimited string listing of regions. As part of deadlock prevention, GT.M sorts the regions in an internal order, eliminating any duplicates from the list. Note: a VIEW with no region sub-argument opens any unopened mapped regions in the current global directory, while one with a list only opens the listed regions. If the VIEW argument has a corresponding environment variable to set the default state, the state applies to databases as the application implicitly opens them with references. Previously such commands accepted either one argument or an asterisk ("*") for all regions, but if supplied with a region string tended to do more than the specified region. (GTM-8922) 
- * READ * and WRITE * operate correctly on files and sockets with a CHSET of UTF-16, UTF-16BE, or UTF-16LE. Previously READ * returned an incorrect codepoint and WRITE * produced an incorrect character or a BADCHAR error. (GTM-8923)
- * GT.M handles socket errors on JOB process startup correctly. Previously if the INPUT and/or OUTPUT for a JOB were in a bad state (e.g. LISTEN) the JOB process could terminate with a KILLBYSIGSINFO1 (signal 11) and a core dump or hang indefinitely. (GTM-8924)

System Administration

- * Please see GTM-5730. (GTM-5374)
- * The Update Process logs record types with their corresponding type description; previously it only logged the integer type value. Note that it is possible, the change could disrupt code that parses the modified results. (GTM-5730) 🍷🍏
- * MUPIP SET -JOURNAL in a replicated environment respects existing region sequence numbers, which aligns with the behavior of MUPIP BACKUP, which can also create new journal files. Previously, MUPIP SET -JOURNAL updated the region sequence numbers to the maximum of the sequence numbers for all regions specified for the command. (GTM-6747) 🍏
- * MUPIP INTEG produces a MUNOACTION error when the command specifies non-existent database; previously it produced an INTEGERRS message in this situation. (GTM-7872)
- * MUPIP JOURNAL -EXTRACT recognizes the -SEQNO=<sequence_number_list> qualifier. <sequence_number_list> is a comma separated list of sequence number(s) in decimal form. It specifies a list of sequence numbers to include or exclude in the journal extract. When a sequence number has a (~) prefix, -SEQNO excludes it from the journal extract. For replicated regions, EXTRACT -SEQNO uses replication sequence numbers, which may select records from multiple regions. For unreplicated regions, EXTRACT uses journal sequence numbers, but specifying sequence number selection with more than one regions produces a JNLEXTRECTSEQNO error. When the sequence number list contains a sequence number involved in a TP transaction, EXTRACT reports it in a broken transaction file when the result does not contain all regions, which is commonly the case without replication, and may be the case with replication when not all regions are available to the utility. Previously EXTRACT did not support record selection by sequence number. (GTM-8202) 🍏
- * The Source Server directs errors to the Source Server log file. Operators should be sure to check the Source Server log; previously it directed errors to stderr, which meant that if it was started from a terminal and the terminal session closed, a subsequent error logged an inappropriate NOPRINCIO error and terminated the Server. (GTM-8576) 🍷🍏
- * MUPIP JOURNAL -ROLLBACK does not inappropriately adjust the ROLLBACK time coordination turn-around point when one, or more, current generation journal file(s) were updated a long time ago. Such a journal file belongs to a so called "idle" region that receives infrequent updates. GT.M V6.3-000 introduced a regression whereby idle regions were not appropriately excluded from determining the ROLLBACK turn-around point. The result was that ROLLBACK could adjust time so far back that the journal files are not present (say due to periodic removal to conserve space) which terminated the ROLLBACK. The operational work-around for this issue was to cut new journal files when deleting old journal files. (GTM-8859)
- * GT.M appropriately removes extra slashes (/) from journal files and output files provided to the MUPIP JOURNAL -EXTRACT command. Previously, when extracting journal files, GT.M kept any extra slashes in the journal file name or output file name, causing multi-slash paths to appear in the output of the extract. The workaround was to avoid using journal files and output files with successive slashes in their paths. (GTM-8860)
- * MUPIP SET -NOENCRYPTABLE can set the database encryptable flag to FALSE. Previously, after invoking MUPIP SET -ENCRYPTABLE attempting to undo that action failed unless the GNUPGHOME pointed to a valid directory and gtm_passwd was defined in the environment. This issue was only observed in the GT.M development environment, and was never reported by a user. MUPIP SET -ENCRYPTABLE now performs some basic encryption setup checks. To execute this command, GNUPGHOME must point to a valid directory and gtm_passwd must be defined in the environment. Previously, the command did not require these environment variables. (GTM-8900) 🍷🍏
- * MUPIP JOURNAL -ROLLBACK and -RECOVER handle large amounts of journal data; while there are still limits, previously when they attempted to handle over around half a billion updates they inappropriately failed with a GTM-F-MEMORY error. (GTM-8906)
- * The receiver process recovers after its update process dies. Previously, on Linux systems with glibc 2.25 or newer, the receiver process could hang indefinitely, requiring manual cleanup of the process and shared memory. This issue was only observed in the GT.M development environment, and was never reported by a user. [Linux](GTM-8916)

Other

- * GT.M supports setting the M stack size using the `gtm_mstack_size` environment variable. This specifies the size of the M stack in KiB. No setting or a setting of 0 uses the default (272KiB). The minimum supported size is 25 KiB; GT.M reverts values smaller than this to 25 KiB. The maximum supported size is 10000 KiB; GT.M reverts values larger than this to 10000 KiB. (GTM-1042) 🟢
- * GT.M ensures that alias and path settings do not interfere with the intentions of the actions it takes using invocations of the `POSIX system()` function. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-3146) 🟢
- * MUPIP INTEG issues a DBKEYMX error in case of a long key name stored in the DT (Directory Tree); previously, it issued a INVSPECREC with no context. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-7483)
- * `gtmpcat` skips scanning `/etc/issue` if the file doesn't exist. Previously, `gtmpcat` gave an error message and exited. (GTM-8752)
- * When invoked at their QUIET or QCALL entrypoints, the `%GCE`, `%GSE`, `%RCE`, and `%RSE` utility routines report only globals or routines in which they find a match, for example: `do QUIET^%GCE` or `do QCALL^%RSE`. The `QCALL` entryref only exists in `%RCE` and `%RSE`. Previously these routines always reported every item they processed regardless of whether it contained a match. (GTM-8777)
- * LKE handles certain control characters appropriately; previously, for example, `<CTRL-Z>` caused a segmentation violation (SIG-11). (GTM-8791)
- * GT.M addresses some issues identified by static analysis tools which appear to be obscure, and, in most cases, harmless. Previously an error with a source file could leak a file descriptor, and, in odd circumstances, trigger management could fail with a segmentation violation. These issues were only observed in the GT.M development environment, and were never reported by a user. (GTM-8870)
- * `%PEEKBYNAME` protects the variable `str` against inappropriate modification in the user symbol space; previously an error in `%PEEKBYNAME`, possibly caused by user input, would change the value of this variable. (GTM-8895)
- * The GT.M reference encryption plugin library retries a failing decryption request once to mask a bug in GnuPG that caused spurious `CRYPTKEYFETCHFAILED` errors during process startup or re-encryption. GnuPG version 2.2.4 fixes the underlying bug, so upgrading to GnuPG 2.2.4 and `libcrypt` 1.8.2 fixes the underlying issue. These problems were only seen in development and not reported by a customer. (GTM-8899) 🟢
- * `gtmpcat` handles `$gtm_dist` values pointing to relative symlinks correctly; previously, `gtmpcat` failed with an `ASSERTFAIL` in some cases. (GTM-8908)
- * The help facility for the utility programs MUPIP, DSE and LKE ignores control-C; previously if a user pressed `<CTRL-C>` while using help, the facility could exit with a harmless, but inappropriate, `ENO256` error. (GTM-8909)
- * MUMPS processes which attempt to process a flush timer while executing an external call defer the timer processing until after the external call is complete. Previously/ flush timers could interfere with non-reentrant routines used in the external call with undesired results. In particular, memory allocation operations in the external call can interfere with `ASYNCIO` setup in the flush timer, resulting in process hangs. (GTM-8926)

Error and Other Messages

CRYPTJNLMISMATCH

CRYPTJNLMISMATCH, Encryption settings mismatch between journal file jjjj and corresponding database file dddd

All GT.M Components Error: Encryption settings in the header of database file dddd do not match those stored in the header of journal file jjjj. This prevents access to the database. The most likely cause is inappropriate operator action such as replacing the current journal file with an older journal file.

Action: Correct the error that caused the incorrect journal file to be pointed to by the database file. If the correct journal file has been inadvertently deleted, create new journal files with the `-noprevjnl` switch. Take a backup as soon as possible thereafter. Depending on your situation, you may need to refresh secondary instances.

ENCRYPTCONFLT2

ENCRYPTCONFLT2, Message: A concurrent MUPIP REORG -ENCRYPT changed the encryption key for RRRR before the process could initialize it

Run Time Warning: Due to a concurrent MUPIP REORG -ENCRYPT, a process was forced to defer encryption key initialization for region RRRR.

Action: None. This information message is only important when followed by other encryption errors.

INVSEQNOQUAL

INVSEQNOQUAL, Invalid SEQNO qualifier value xxxxx

MUPIP Error: This error indicates that MUPIP JOURNAL -EXTRACT -SEQNO command could not extract a journal file because an invalid SEQNO format was specified.

Action: Enter a comma separated list of valid sequence numbers ('0' or positive integers) as value for the SEQNO qualifier. The format of the -SEQNO qualifier is `-SEQNO=seqno1[,seqno2,seqno3,....]` where seqno is the region sequence number in decimal format.

JNLACCESS

JNLACCESS, Error accessing journal file jjjj

Run Time Error: GT.M sends this message to the system log followed by other messages detailing the failure. jjjj is the file-specification for the inaccessible journal. In most situations, this error occurs when the journal file storage runs out of disk space.

Action: Review the accompanying message(s) for additional information. This means an error while trying to write to the journal file.

JNLBADRECFMT

JNLBADRECFMT, Journal Record Format Error encountered for file jjjj at disk address yyyy

MUPIP/Run Time Error: This indicates that an attempt to open a journal file encountered an invalid record.

Action: Report the entire incident context to your GT.M support channel.

JNLCYCLE

JNLCYCLE, Journal file jjjj causes cycle in the journal file generations of database file dddd

MUPIP Error: This indicates that MUPIP encountered journal file jjjj causing cycle in the journal file generations of database file dddd; that is following the back-pointers in the journal files can wind up repeatedly finding the same journal file.

Action: Contact your GT.M support channel with appropriate log messages.

JNLDBERR

Last used version: v4.4-002

JNLDBERR, Journal file jjjj does not correspond to database dddd

Run Time Error: This indicates that GT.M could not open journal file jjjj for database file dddd because the journal file header identifies itself as belonging to a different database file that does not exist in the system.

Action: Use a MUPIP SET command with the qualifier JOURNAL to create a journal file that matches the database.

JNLDISABLE

JNLDISABLE, Specified journal option(s) cannot take effect as journaling is DISABLED on database file dddd

MUPIP Warning: This indicates that none of the specified journal option(s) in MUPIP SET -JOURNAL or MUPIP BACKUP command took effect, because journaling was found DISABLED on database file dddd.

Action: Revise the selection qualification to exclude the DISABLED region(s) or, if appropriate, enable journaling on those regions.

JNLEXTEND

JNLEXTEND, Journal file extension error for file jjjj.

Run Time/MUPIP Error: Journal file jjjj failed to extend. If the environment is not configured for instance freeze, this causes journaling to be turned off for the region.

Action: Review the accompanying message(s) and take appropriate action. If the environment is not configured for instance freeze, perform a MUPIP BACKUP, that turns journaling on again, to reestablish durability.

JNLEXTR

Last used version: V4.4-002

JNLEXTR, Error writing journal extract file: xxxx

MUPIP Error: This indicates that an error was encountered while trying to write to either the JNL EXTRACT file or lost-transaction file or broken-transaction file as part of a MUPIP JOURNAL command.

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

JNLEXTRCTSEQNO

JNLEXTRCTSEQNO, Journal Extracts based on sequence numbers are restricted to a single region when replication is OFF

Error and Other Messages

MUPIP Error: When replication is enabled GT.M applies a uniform set of sequence numbers across regions, but when it is not in use each region has its own set of sequence numbers, and, in that case, MUPIP only works on a region at a time.

Action: If you need cross region sequence numbers, start replication with at least a passive Source Server; otherwise use one MUPIP JOURNAL -EXTRACT command for each region when using the -SEQNO qualifier.

JNLFILOPN

JNLFILOPN, Error opening journal file jjjj for database file dddd

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

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

JNLFLUSHNOPROG

JNLFLUSHNOPROG, No progress while attempting to flush journal file jjjj

Run Time Warning: Indicates processes needing space in the journal buffers were unable to write journal jjjj because even though multiple processes have controlled the resource, this process has not been able to flush records. JNLPROCSTUCK means one process is hogging, while this message means more than one process has tried but none have succeeded. Might indicate a clogged disk subsystem on which journal file JJJJ resides.

Action: Check the log file for other journaling related messages. Consider balancing disk subsystem load.

JNLFSYNCLSTCK

JNLFSYNCLSTCK, JNLFSYNCLSTCK, Journaling fsync lock is stuck in journal file jjjj

Run Time Error: A resource controlling journal file actions has remained unavailable for a long period.

Action: Check on the condition of the process identified in the associated messages.

JNLINVALID

JNLINVALID, jjjj is not a valid journal file Region: rrrr

MUPIP Error: This indicates that GT.M could not open journal file jjjj, due to an error that is detailed in the accompanying previous message(s). While trying to create a new journal file for the same region it encountered errors. rrrr is the region name associated with the journal.

Action: Review the accompanying error message(s) to determine the cause of the failure of the new journal file creation. After the cause is resolved, to reestablish durability, perform a MUPIP BACKUP that turns journaling back on.

JNLNAMLEN

Last used version: V4.4-002

JNLNAMLEN, Journal file jjjj: for database file dddd exceeds maximum of MMMM

MUPIP Error: This indicates that the file-specification jjjj of the journal for database file dddd exceeds the maximum supported length of MMMM.

Action: Modify the journal file-specification to adhere to the file length restrictions.

JLNOCREATE

JLNOCREATE, Journal file jjjj not created

MUPIP/Run Time Error: This indicates that GT.M could not create journal file jjjj.

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

JNLORDBFLU

JNLORDBFLU, Error flushing database blocks to dddd. See related messages in the operator log

MUPIP Error: This message indicates that hardening journal or database records could not be completed due to an error. The operator log should contain one or more accompanying messages indicating the cause of the error.

Action: Verify the normal state of the file system and appropriate permissions of the database and journal files. Report the entire incident context to your GT.M support channel along with any GT.M operator log messages within the same time frame.

JNLREADEOF

JNLREADEOF, End of journal file encountered for jjjj

MUPIP/Run Time Error: This indicates that MUPIP JOURNAL or a run-time journal operation encountered the end-of-file for the journal file jjjj, before it completed processing.

Action: This error indicates an improperly closed journal file. Restart journaling with a MUPIP BACKUP -NEWJNLFILES or a MUPIP SET - JOURNAL and report all available circumstance to those responsible for supporting your database operations.

JNLRECFMT

JNLRECFMT, Journal file record format error encountered

MUPIP Error: This indicates that MUPIP JOURNAL encountered an invalid record in the journal file.

Action: In the event of GT.M issuing this error message, use MUPIP BACKUP to ensure durability by creating a fresh set of journals consistent with the database. Else, to resume operation, restore the database from the last backup and play forward the updates using the appropriate MUPIP JOURNAL command. As soon as possible, report the entire incident context with information from the operator log and any other relevant information to your GT.M support channel.

JNLSPACELOW

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

Run Time Information: This indicates that the journal file jjjj is approaching the maximum size specified for it. The system creates a new journal file when the limit is reached.

Action: None required except as part of monitoring journaling space requirements or when operational practice uses this as a trigger to intervene in journal file management.

JNLSWITCHSZCHG

JNLSWITCHSZCHG, Journal AUTOSWITCHLIMIT [aaaa blocks] is rounded down to [bbbb blocks] to equal the sum of journal ALLOCATION

Error and Other Messages

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

Action: If the rounded value is inappropriate examine the alignsize, allocation and extension values and choose a more suitable value.

JNLSWITCHTOOSM

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

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

Action: Specify a higher value of AUTOSWITCHLIMIT.

JNLWRERR

JNLWRERR, Error writing journal file xxxx. Unable to update header Region: yyyy

Run Time/MUPIP Error: This indicates that GT.M encountered an error while updating the journal file header as part of trying to open the journal file.

Action: Review the accompanying message(s) for detail on the cause of the error. GT.M automatically closes the current journal file and creates a new one. To reestablish durability, perform MUPIP BACKUP to create a fresh set of journals consistent with the database.

MSTACKSZNA

MSTACKSZNA, User-specified M stack size of SSSS KiB not appropriate; must be between LLLL KiB and MMMM KiB; reverting to VVVV KiB

Run Time Information: The gtm_mstack environment variable species an M stack size outside the range GT.M supports, where LLLL and MMMM are the lower and upper bounds respectively; VVVV is the value actually used.

Action: None required immediately as the process operates with the reported size M stack, however it would be preferable to eliminate such messages by setting gtm_mstack to a value in the supported range.

PREMATEOF

PREMATEOF, Premature end of file detected

MUPIP/Run Time Error: A file read or write detected an end-of-file when it was expecting additional records.

Action: Analyze accompanying messages for the type of file on which the operation failed. If the operation was a MUPIP LOAD, refer to the About this Manual section on MUPIP LOAD errors earlier in this manual. If the circumstances warrant, group responsible for database integrity at your operation with all the diagnostic context you can gather.

SYSUTILCONF

SYSUTILCONF, Error determining the path for system utility. tttt

Run Time Error: tttt represents text describing details of an issue finding a POSIX function that it needed.

Action: Check for aliases or environment variables related to paths that might be interfering with GT.M's ability to invoke functions.