GT.M

Release Notes

V6.3-001

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

Revision History

Revision 1.5	04 July 2022	Added GTM-8669.
Revision 1.4	11 March 2020	Add GTM-8547.
Revision 1.3	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 1.2	25 September 2017	Correct the description of GTM-8362 and add GTM-8632.
Revision 1.1	06 April 2017	V6.3-001A
Revision 1.0	15 March 2017	V6.3-001 - First published version

Table of Contents

V6.3-001A

Overview

GT.M V6.3-001A provides timely remediation for a flaw introduced with GTM-8637 in V6.3-001. This flaw was discovered during GT.M testing immediately after the release and was never reported by a user. V6.3-001A also includes bug fixes and one enhancement and is suitable for production use. For more information, refer to Change History - V6.3-001A.

V6.3-001 brings important and useful enhancements to GT.M.

GT.M provides a fast and efficient mechanism for processes to opt-in to share their database access statistics for other processes to monitor. The statistics are the same as those available to the process itself using the ZSHOW "G" command. With almost no impact on monitored processes, a monitoring process can rapidly identify, for example, which processes are performing the most global SETs, or which ones are encountering the most database access conflicts (GTM-6699).

MUPIP FREEZE -ONLINE freezes database writes from global buffers to the file system, while allowing applications to continue database updates as long as they are able to, without requiring a write to the database file system. During this time, journal writes continue, ensuring database recoverability. A typical use is to freeze a file system to take a snapshot, or break a mirror, an operation which can take seconds to over a minute. MUPIP FREEZE without the -ONLINE enhancement freezes database updates by application processes (GTM-8362).

V6.3-001 includes multiple optimizations for performance, some applicable to all platforms, and others specific to Linux on x86_64.

Introduced as field test grade functionality in a production release, asynchronous IO is an option for databases using the BG access method. Unlike traditional database IO, which performs synchronous IO through the file system cache, asynchronous IO bypasses the file system cache. The performance characteristics of asynchronous IO are likely to be quite different from traditional sequential IO. Although asynchronous IO in theory should be more efficient than synchronous IO by eliminating the need for the UNIX file buffer cache and thereby eliminating certain file system locks (e.g., file systems mounted with AIX's CIO mount option, in practice asynchronous IO is likely to emerge from the starting gate under-performing synchronous IO because of the years that synchronous IO has been the common IO model operating systems and file systems have had used by applications. Please anticipate extensive benchmarking and tuning for your application to achieve the best performance it can with asynchronous IO. (GTM-6838).

GT.M accepts routines with <CR><LF> line terminators. FIS thanks the participants of the 2016 "Hacking GT.M" workshop for this enhancement (GTM-4283).

As always, the release bring numerous smaller enhancements, and fixes. See the Change History below.

: [illegible] 1 > [illegible]7 + : 3 [illegible] > [illegible] 1 > [illegible]7 [illegible] 7 [illegible] 1 > [illegible]7 [illegible]

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,	-updnotok

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
		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	6.1, 7.1 TL 4, 7.2	Only 64-bit versions of AIX with POWER6 as the minimum required CPU architecture level are Supported. While GT.M supports both UTF-8 mode and M mode on this platform, there are problems with the AIX ICU utilities that prevent FIS from testing 4-byte UTF-8 characters as comprehensively on this platform as we do on others. Running GT.M on AIX 7.1 requires APAR IZ87564, a fix for the POW() function, to be applied. To verify that this fix has been installed, execute **instfix -ik IZ87564.** AIX 7.1 TL 5 is Supportable. Only the AIX jfs2 filesystem is Supported. Other filesystems, such as jfs1 are Supportable, but not Supported. FIS strongly recommends use of the jfs2 filesystem on AIX; use jfs1 only for existing databases not yet migrated to a jfs2 filesystem. Effective May 1, 2017, FIS intends to require 7.1 as the minimum level of AIX, and POWER7 as the minimum required CPU architecture level.
x86_64 GNU/Linux	Red Hat Enterprise Linux 6 and 7; Ubuntu 14.04 LTS and 16.04 LTS	To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware. GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6.32 or later), glibc (version 2.12 or later) and ncurses (version 5.7 or later). To install GT.M with Unicode (UTF-8) support on RHEL 6, in response to the installation question **Should an ICU version other than the default be used? (y or n)** please respond **y** and then specify the ICU version (for example, respond 4.2) to the subsequent prompt **Enter ICU version (ICU version 3.6 or later required. Enter as major-ver.minor-ver):** GT.M requires the libtinfo library. If it is not already installed on your system, and is available using the package manager, install it using the package manager. If a libtinfo package is not available: ● Find the directory where libncurses.so is installed on your system.

Platform	Supported Versions	Notes
		• Change to that directory and make a symbolic link to libncurses.so.<ver> from libtinfo.so.<ver>. Note that some of the libncurses.so entries may themselves be symbolic links, for example, libncurses.so.5 may itself be a symbolic link to libncurses.so.5.9. To support the optional WRITE /TLS fifth argument (the ability to provide / override options in the tlsid section of the encryption configuration file), the reference implementation of the encryption plugin requires libconfig 1.4.x. As this is a higher level than that distributed with Red Hat Enterprise Linux 6, in order to use this feature of WRITE/TLS on that platform with the reference implementation, please install libconfig 1.4.x, including the header files, and recompile the reference implementation of the encryption plugin. 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). A bug in the Linux 3.13 kernels used in Ubuntu 14.04 LTS (https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1502168) affects GT.M operation. As newer kernels do not exhibit this misbehavior, FIS recommends that you follow the Ubuntu LTS Enablement Stack procedure (https://wiki.ubuntu.com/Kernel/LTSEnablementStack) and use newer kernels to avoid the behavior until such time as the bug is fixed in the 3.13 kernels. 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 ⊠⊠⊠ ensure that your kernel includes commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3) is in your kernel. 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. Effective July 1, 2017, FIS intends to require: • the then current level of 7 (e.g, 7.3) as the minimum supported level of Red Hat Enterprise Linux; and • 16.04 LTS, as the minimum supported level of Ubuntu Linux.

Platform	Supported Versions	Notes
		If these will cause you hardship, please contact your FIS account manager or your GT.M support channel.
x86 GNU/Linux	Red Hat Enterprise Linux 6	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. Effective July 1, 2017, FIS intends to consider only Debian 8 (Jessie), or the then Debian Stable, as the sole Supported platform for the 32-bit version. 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
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.

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.

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 V6.3-001A in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.3-001A_arch (for example, /usr/lib/fis-gtm/V6.3-001A_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.3-001A_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 RUNDOWN 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. Also, modify your environment scripts to include the use of gtmcrypt_config environment variable.

Compiling the Reference Implementation Plugin

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

1. Install the development headers and libraries for libgcrypt, libgpgme, libconfig, and libssl. On Linux, the package names of development libraries usually have a suffix such as -dev or -devel and are available through the package manager. For example, on Ubuntu_x86_64 a command like the following installs the required development libraries:

   ```
   sudo apt-get install libgcrypt11-dev libgpgme11-dev libconfig-dev libssl-dev
   ```

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

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

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

3. Follow the instructions in the README.

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

   - Define the gtm_dist environment variable to point to the absolute path for the directory where you have GT.M installed

   - Copy and paste the commands from the README to compile and install the encryption plugin with the permissions defined at install time

Upgrading to GT.M V6.3-001A

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 V6.3-001A 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 V6.3-001A 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 V6.3-001A.
- 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 V6.3-001A.
- 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 V6.3-001A 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

V6.3-001A does not require new replication instance files if you are upgrading from V5.5-000. However, V6.3-001A 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 V6.3-001A 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 V6.3-001A 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 V6.3-001A 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/V6.3-001A.

To extract and reapply the trigger definitions on V6.3-001A 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 V6.3-001A, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.3-001A 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 V6.3-001A.
6. Run **mumps -run %XCMD 'if $ztrigger("file","trigger_defs.trg")'** to reapply your trigger definitions.
7. Turn replication on.
8. Connect to the originating instance.

Note

Reloading triggers renumbers automatically generated trigger names.

Downgrading to V5 or V4

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

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

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

-VERSION specifies the desired version for the database header.

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

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

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

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

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

1. In a GT.M V6.3-001A 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 Unicode

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_V6.3-001A_i686, then gtmprofile sets $gtm_dist to /usr/lib/fis-gtm/gtm_V6.3-001A_i686/utf8).
    - On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

For more information on gtmprofile, refer to the Basic Operations chapter of GT.M Administration and Operations Guide.

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

Setting the environment variable TERM

The environment variable TERM must specify a terminfo entry that accurately matches the terminal (or terminal emulator) settings. Refer to the terminfo man pages for more information on the terminal settings of the platform where GT.M needs to run.

- Some terminfo entries may seem to work properly but fail to recognize function key sequences or fail to position the cursor properly in response to escape sequences from GT.M. GT.M itself does not have any knowledge of specific terminal control characteristics. Therefore, it is important to specify the right terminfo entry to let GT.M communicate correctly with the terminal. You may need to add new terminfo entries depending on your specific platform and implementation. The terminal (emulator) vendor may also be able to help.
- GT.M uses the following terminfo capabilities. The full variable name is followed by the capname in parenthesis:

```
auto_right_margin(am), clr_eos(ed), clr_eol(el), columns(cols), cursor_address(cup),
 cursor_down(cud1), cursor_left(cub1), cursor_right(cuf1), cursor_up(cuu1),
 eat_newline_glitch(xenl), key_backspace(kbs), key_dc(kdch1),key_down(kcud1),
 key_left(kcub1), key_right(kcuf1), key_up(kcuu1), key_insert(kich1),
 keypad_local(rmkx),keypad_xmit(smkx), lines(lines).
```

GT.M sends keypad_xmit before terminal reads for direct mode and READs (other than READ *) if EDITING is enabled. GT.M sends keypad_local after these terminal reads.

Installing Compression Libraries

If you plan to use the optional compression facility for replication, you must provide the compression library. The GT.M interface for compression libraries accepts the zlib compression libraries without any need for adaptation. These libraries are included in many UNIX distributions and are downloadable from the zlib home page. If you prefer to use other compression libraries, you need to configure or adapt them to provide the same API as that provided by zlib.

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

By default, GT.M searches for the libz.so shared library in the standard system library directories (for example, /usr/lib, /usr/local/lib, /usr/local/lib64). If the shared library is installed in a non-standard location, before starting replication, you must ensure that the environment variable LIBPATH (AIX) or LD_LIBRARY_PATH (GNU/Linux) includes the directory containing the library. The Source and Receiver Server link the shared library at runtime. If this fails for any reason (such as file not found, or insufficient authorization), the replication logic logs a DLLNOOPEN error and continues with no compression.

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

Change History

V6.3-001A

Fixes and enhancements specific to V6.3-001A:

Id	Prior Id	Category	Summary
GTM-8632	-	Other	On Linux, an environment variable to help manage core file generation
GTM-8685	-	DB	Forward Rollback Consistency with Journal Renaming
GTM-8700	-	DB	Fixes to issues related to statistics sharing
GTM-8702	-	DB	Always ensure before images when needed in TP allocations
GTM-8705		Language	ZSHOW/ZWRITE work correctly even if they result in a garbage collection

V6.3-001

Fixes and enhancements specific to V6.3-001:

Id	Prior Id	Category	Summary
GTM-4283	S9C04-002078	Other	GT.M accepts routines with "DOS-style" terminators
GTM-5206	S9D12-002397	DB	If possible, avoid journal file hardening when holding a database critical section
GTM-6037	C9H07-002874	DB	Reporting of numeric subscripts in globals which should not use them
GTM-6085	C9H10-002922	Other	Update Process Reader Helper avoids rare abnormal termination
GTM-6332	C9J01-003085	Other	Code generation cleanup on Linux
GTM-6598	C9K04-003268	Admin	Limit the time MUPIP BACKUP and INTEG wait for kill-in-progress to clear
GTM-6699	D9K10-002792	DB	Opt-in facility for statistics sharing
GTM-6793	C9L04-003395	Language	Limit MERGE into an lvn to 31 subscripts

Id	Prior Id	Category	Summary
GTM-6838	C9L06-003425	DB	Asynchronous database IO
GTM-7593	-	Admin	More reliable journal flushing in edge cases under replication
GTM-7729	-	Admin	LOCKs have separate resource management by default, but you can select shared resource management
GTM-7778	-	Other	More consistent timestamps on AIX gtmsecshr messages
GTM-7837	-	Admin	MUPIP REPLICATE -CHANGELOG waits for up to 25 seconds to confirm a change log request
GTM-7857	-	DB	Reporting of empty-string subscripts that conflict with database settings
GTM-7922	-	Admin	MUPIP EXTRACT protects against concurrent updates to region spanning updates
GTM-8254	-	Language	USE command accepts [I|O]CHSET for Sequential Disks, SOCKETs and terminals
GTM-8357	-	Language	GT.M initializes the value of $ZSTEP from the environment variable $gtm_zstep if it is defined
GTM-8359	-	Language	More general support for indirect arguments to TSTART commands and an explicit error for TSTART in direct mode
GTM-8362	-	Admin	Allow updates during a MUPIP FREEZE
GTM-8366	-	Language	Clear $REFERENCE after an error in a global reference
GTM-8373	-	Admin	Accept rapid changes to replication logs destination files
GTM-8385	-	Language	Computations using literals largely performed at compile time
GTM-8427	-	Language	$ZCOLLATE() converts a GVN and $ZATRANSLATE() an expression to a key representation and back
GTM-8430	-	Other	Improvement in critical section acquisition for x86_64 Linux

Id	Prior Id	Category	Summary
GTM-8447	-	Admin	MUPIP DUMPFHEAD provides lighter weight and less problematic access to database state information
GTM-8542	-	Admin	MUPIP LOAD -FORMAT=Binary validates keys in the extract records
GTM-8544	-	Other	Timer interrupt reduction
GTM-8547	-	Admin	Source Server handles reconnection errors more gracefully.
GTM-8553	-	DB	Eliminate an interaction between poollimit and extended global references
GTM-8559	-	Language	$gtm_etrap and $gtm_trigger_etrap accept 8192 byte strings
GTM-8561	-	Other	Deal appropriately with M process data exceeding 2GiB
GTM-8562	-	Other	The configure install script creates routine directories even when reusing an existing directory
GTM-8564	-	Admin	Protect an interrupted MUPIP REORG -ENCRYPT from an intervening MUPIP REORG -TRUNCATE
GTM-8567	-	Other	configure script defaults locale to allow a UTF-8 installation to proceed
GTM-8568	-	Language	Fix three rare trigger update issues
GTM-8569	-	Other	Prevent rare duplicate messages when the number of processes in an instance exceeds 32ki
GTM-8570	-	Other	Eliminate superfluous literals from object modules
GTM-8571	-	Other	Fix regression in ^%GI for GO format containing any empty data values
GTM-8572	-	Language	Prevent instances of pre-evaluation in $SELECT()
GTM-8573	-	Language	Compiler optimization for IF <literal> and command postconditional literals
GTM-8578	-	Other	Correct handling of non-char arrays in %PEEKBYNAME()

Id	Prior Id	Category	Summary
GTM-8579	-	Language	Operator optimizations at compile time
GTM-8582	-	Admin	Appropriate Source Server startup with Sync I/O and 4KiB sector sizes on an XFS filesystem
GTM-8583	-	Language	Prevent inappropriate MAXNRSUBSCRIPTS errors from MERGE
GTM-8584	-	Language	Prevent GTMASSERT caused by a <NUL> character generating a NOCANONICNAME error message
GTM-8590	-	Language	Certain timed operations deferred during $ZCONVERT()
GTM-8593	-	Language	OPEN of an existing SOCKET device can modify ZFF & delimiters; also $KEY is always in UTF-8
GTM-8595	-	Language	Fix to M mode handling of non-ASCII literals
GTM-8597	-	DB	Better protection against one type of random error
GTM-8598	-	Other	Certain operations deferred while processing character input
GTM-8599	-	Admin	Correct odd case for MUPIP JOURNAL -ROLLBACK -FORWARD
GTM-8602	-	Other	MUPIP JOURNAL accepts multiple EOF records
GTM-8609	-	Admin	Improved error messaging for errors that occur during the first access of a journal file
GTM-8610	-	Admin	The Receiver Server avoids rare situations that could cause it to exit
GTM-8612	-	Admin	MUPIP LOAD handles minimal headers and long lines gracefully
GTM-8613	-	DB	Protect against concurrent creation of global variable with differing collation characteristics
GTM-8614	-	Admin	REQROLLBACK message indicates required -ROLLBACK also requires -NOONLINE

Id	Prior Id	Category	Summary
GTM-8615	-	Other	Certain timed operations deferred during external calls
GTM-8629	-	Admin	MUPIP RESTORE treats truncated input as an error
GTM-8637	-	Other	Prevent GTMASSERT2 from very long running jobs performing TP transactions
GTM-8639	-	Language	ZSHOW "G" provides Block Transition to Dirty (BTD) statistic for BG databases
GTM-8640	-	Language	Accept <NUL> characters in literals use for indirection and XECUTE
GTM-8641	-	DB	$ORDER(,-1) of global variables that span database blocks returns correct result
GTM-8642	-	Other	Prevent rare inappropriate GTMASSERT2 from MUPIP STOP of a process while changing encryption keys
GTM-8645	-	Other	Improved (but imperfect) tracking of database reference count
GTM-8654	-	Admin	Revised permission handling when determining group membership
GTM-8655	-	DB	Improved database structural integrity protection against kill -9
GTM-8656		Admin	Support for OpenSSL 1.1.0
GTM-8657	-	Admin	Replication from BC to SI handles multiple connections with no intervening updates
GTM-8659	-	Other	Better file cleanup in case of abnormal termination during file creation
GTM-8660	-	Admin	Improved System Profiling for x86_64 Linux editions
GTM-8664	-	DB	Defer idle epoch while holding journal pool critical section lock.
GTM-8668	-	Admin	Simplify GPG Agent interaction with the encryption reference implementation
GTM-8669	-	Other	Fix a rare buffer overrun when calling to external Java libraries and programs
GTM-8672	-	Language	Adjustments to improve the memory utilization for heap space (primarily local variable storage)

Id	Prior Id	Category	Summary
GTM-8676	-	Other	Fix to DSE FIND -KEY
GTM-8679	-	Language	Global name-level $ORDER() maintains $REFERENCE
GTM-8686	-	Other	^%TI works as documented
GTM-8687	-	DB	GT.M properly handles loading successive global directories with increasing numbers of regions
GTM-8689	-	Language	Protect OPEN command against external actions

Database

- V6.3-001 GT.M defers the hardening (fsync) of a journaled database file (a potentially time consuming operation) to occur as much as possible outside the database critical section, particularly when it is time to write an EPOCH record in the journal file. Previously, this was done while holding the critical section which could affect database transaction throughput. (GTM-5206)

- V6.3-001 MUPIP INTEG, DSE INTEG and in some instances VIEW "GDSCERT" produce a NONUMSUBS error if they encounter a numeric subscript in a global variable tree that has been defined to only use string subscripts; previously, they did not report this issue. (GTM-6037)

- V6.3-001 GT.M provides a fast and efficient mechanism for processes to share their database access statistics for other processes to monitor. In addition GT.M now supports implicit instantiation of a database file. Please refer to the Additional_Information section for the details and implications of this feature. (GTM-6699)

- V6.3-001 Released as field test grade functionality in a production release, asynchronous IO is an option for database segments using the BG access method; previously GT.M performed only synchronous I/O through the file system cache. Also, when invoked from the shell, GDE returns a non-zero status in case it terminates with errors; previously it always returned a zero status, even if it encountered errors. Note: when invoked from GT.M, GDE does not return a status. MUPIP RUNDOWN appropriately manages the FTOK semaphore associated with a database to which it has read-only access; previously it inappropriately removed that semaphore if the database was quiescent but its attachment counter had ever exceeded 32Ki simultaneous processes. Please refer to the Additional Information section for details. (GTM-6838)

- V6.3-001 When MUPIP INTEG, DSE INTEG -BLOCK, or VIEW "GDSCERT":1 processing encounter an empty-string ("null") subscript that does not match current file header settings for the null characteristic they issue a DBNULCOL or NULLSUBS error. If MUPIP LOAD-FORMAT=BINARY encounters empty-string subscripts in an extract it is loading on a database that does not permit such subscripts, it produces a NULLSUBS error. Previously, these functions did not report these errors (MUPIP LOAD did not load the offending data). Also, if MUPIP LOAD -FORMAT=BINARY encounters two keys that are the same except for their representations of empty string subscripts, it produces a warning message and discards the one whose representation does not match that of the database into which the data is being loaded. (GTM-7857)

- V6.3-001 Extended references work correctly when poollimit is enabled. Previously, such a combination could produce GTMASSERT failures. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8553)

- V6.3-001 GT.M protects itself against one type of random error in a control field used for buffer management. Previously, recovery required shutting the database down. The error observed most

likely resulted from some sort of hardware malfunction. Note that FIS recommmends the use of ECC RAM in production systems. (GTM-8597)

- V6.3-001 GT.M issues a ACTCOLLMISMTCH error in case multiple processes concurrently attempt to create the first global node of a global variable using differing collation characteristics (defined through the -GBLNAME section of their respective global directories). Previously, it was possible for processes to proceed to update with conflicting views of the global's collation, resulting in global nodes with misaligned collation sequences in the same global, creating an application level data integrity error. This issue was only observed in the GT.M development environment, and was never reported by a user.(GTM-8613)

- V6.3-001 $ORDER(gvn,-1) and $ZPREVIOUS(gvn) work correctly in case of spanning nodes. Since GT.M V6.2-002, due to a regression introduced in GTM-7917, it could return the same value as the last subscript in the input key in case the global corresponding to gvn contained spanning nodes. For example if ^X(3) existed and was a spanning node, $ORDER(^X(4),-1) would return 4 instead of 3, potentially leading to infinite loops. If ^X(3) was not a spanning node, $ORDER() would return the correct value. The return value was unaffected by the existence or non-existence of a subtree of ^X(3). (GTM-8641)

- V6.3-001 GT.M does better in maintaining database integrity in the face of a kill -9 of a process in the middle of a database commit. Previously, if a process trying to update a particular GDS block was killed in the middle of a commit, it was possible to lose prior updates to just that block within the same epoch, resulting in a database file with structural damage. Note that FIS continues to strongly recommend against kill -9 of processes that have opened a database file. (GTM-8655)

- V6.3-001 GT.M defers performing an idle epoch if it concurrently holds the journal pool critical section lock. Previously, under rare conditions, performing the idle epoch in this situation could result in database deadlock. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8664)

- GT.M marks journal files as current or not current, which allows it to deal more appropriately with an operational situation in which an operator renamed an older journal file as current. Previously, GT.M MUPIP ROLLBACK -FORWARD could not detect this operational issue and consequently failed to deliver a consistent state at completion. (GTM-8685)

- V6.3-001 GT.M now properly handles loading multiple global directories with increasing numbers of regions. Previously, this could have resulted in memory corruption if a process first opened the first global directory with fewer regions AND both global directories had at least one global variable name that spanned multiple regions. This issue was never reported by users and was only seen in the GT.M development environment. (GTM-8687)

- Operations on statistics database files (e.g. VIEW "STATSHARE", VIEW "NOSTATSHARE", direct access to ^%YGS global nodes) work correctly even in the case a process accesses the same statistics database file through more than one global directory. In GT.M V6.3-001, this could cause the process to terminate abnormally with a segmentation violation (SIG-11). GT.M correctly creates statistics database files (which are always created dynamically) even in case of heavy database contention

amongst processes. In GT.M V6.3-001, if a process in a TP transaction (TSTART/TCOMMIT fence) was in its final retry (due to restarts from other concurrently running processes), it was rare but possible to encounter a deadlock. $ZPEEK issues a BADZPEEKARG error when its argument specifies a non-existent lower-case region name. In GT.M V6.3-001, this terminated the process with a segmentation violation (SIG-11). MERGE, ZWRITE, $DATA(), $ORDER(), and $QUERY() involving ^%YGS open any implicitly associated statistics database files when run outside of a TP transaction. In GT.M V6.3-001, they could ignore ^%YGS nodes corresponding to regions previously unopened or untracked by the process. Note that these operations within an explicit TP transaction do not implicitly open any not-yet-open statistics database until after the transaction commits. GT.M correctly handles a MUPIP SET -REPLICATION=ON for a region which was still replicating after journaling turned off (in the "was_on" state). In GT.M V6.3-001, due to a regression introduced by GTM-7593, an exiting process executing in a small window of instructions in database rundown logic could, in rare, cases terminate with a segmentation violation (SIG-11). MUPIP SET JOURNAL and MUPIP REPLICATE -INSTANCE_CREATE issue appropriate errors when their input attempts to create journal and/or replication-instance file names whose absolute path is more than 255 bytes long. Previously, it was possible for these commands to abnormally terminate with a "stack smashing detected" error due to GT.M-internal buffer overflows. All these issues were only observed in the GT.M development environment, and were never reported by a user. (GTM-8700)

- Updates within explicit (TSTART/TCOMMIT) or implicit (spanning nodes or regions, or triggers) transactions requiring one or more additional blocks reliably write before images (if configured) to the journal file or to a MUPIP BACKUP -ONLINE. Missing before images could cause incorrect or damaged databases after a MUPIP JOURNAL -RECOVER or -ROLLBACK, or in a backup. In GT.M V6.3-001, due to a flaw in GTM-8637, in rare cases when choosing a previously freed block, GT.M failed to appropriately write a before image. This issue were only observed in the GT.M development environment, and was never reported by a user.(GTM-8702)

This page is intentionally left blank.

Language

- V6.3-001 MERGE into a local variable (lvn) target limits the number of target subscripts to the maximum number supported by GT.M (currently 31); previously, MERGE could produce variables with 32 subscripts which could cause subsequent problems. (GTM-6793)

- V6.3-001 The USE command accepts [I|O]CHSET as valid deviceparameters. It is possible to change the character set of an open device. In addition to USE, the OPEN command also changes the character set of an already opened device including Sequential Disk, SOCKET and terminal devices. It is useful to deal with binary data intermixed with character data. Previously, there was no documented way to change the character set of an open device. (GTM-8254)

- V6.3-001 GT.M takes the initial value of $ZSTEP from the environment variable gtm_zstep, with a default value of "B" (a BREAK command) if gtm_zstep is not defined; previously, changing the default value required a SET command. (GTM-8357)

- V6.3-001 TSTART commands with indirect arguments work correctly in GT.M. Previously, they allowed only specification of a single local variable with no parentheses, SERIAL flag or transaction id parameter. Also, use of TSTART in direct mode is prohibited and generates a NODMTSTART error. Previously, TSTART/TCOMMIT sometimes worked when on the same direct mode command line but more often got strange errors and caused memory leaks at best. (GTM-8359)

- V6.3-001 GT.M assigns an empty string to $REFERENCE when there is an error while constructing a subscripted global variable. Previously, GT.M assigned the last successful subscripted global variable to $REFERENCE. (GTM-8366)

- V6.3-001 GT.M performs arithmetic operations involving only literals at compile time, with the exception of divide and integer divide (/ and \) by zero (0), which because of their use to intentionally produce an error is left to run time. Note that modulo (#) produces a compile time error. Previously, GT.M did all such calculations at run-time. (GTM-8385)

- V6.3-001 $ZCOLLATE(glvn,intexpr[,{0|1}]) returns a transformed representation of a first argument glvn using the alternative transform specified by the second argument intexpr that, by default, or if the optional third argument is zero (0), represents a normalized form that can be used as an operand to the follows (]) or sorts-after operator ([[) such that, if both operands are in the normalized form, the result is independent of alternative collation. If the optional third argument is non-zero, $ZCOLLATE() returns a reverse transform of the first argument intended to restore the normalized form to the native M glvn representation. $ZCOLLATE() replaces the "YGVN2GDS" and "YGDS2GVN" arguments to $VIEW(), which are deprecated. $ZATRANSFORM(expr,intexpr[,{0|1}] [,{0|1}]) returns a transformed representation of a first argument expr, treated as a subscripted or unsubscripted key, using the alternative transform specified by the second argument intexpr in a normalized form that can be used as an operand to the follows (]) or sorts-after (]]) operator such that, if both operands are in the normalized form, the result is independent of alternative collation. If

the optional third argument is non-zero, $ZATRANSFORM() returns a reverse transform of the first argument intended to restore a normalized form to the native M expr representation. By default, or if the optional fourth argument is zero, $ZATRANSFORM()returns the transformation of expr using standard M collation of numbers before strings, causing numbers to sort like strings; if the optional third argument is non-zero, $ZATRANSFORM() treats all expressions as strings.(GTM-8427)

- V6.3-001 $gtm_etrap and $gtm_trigger_etrap accept up to 8192 bytes and produce a LOGTOOLONG message in the syslog when ignoring a value longer than 8192. Previously, process initialization limited both environment variables to 4096 bytes and did not log any message for an over-length $gtm_trigger_etrap. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8559)

- V6.3-001 $ZTRIGGER() operations that affect triggers executed in the same transaction work as documented. Previously, in rare situations, transactions that performed $ZTRIGGER() operations in some, but not all, retries and invoked a trigger affected by the $ZTRIGGER() operation in some, but not all, retries could result in a TRIGDEFBAD or SIG-11. Also, trigger load operations in which an error occurred only perform a validity check on trigger delete by name operations. Previously, if an error occurred during a trigger load, a subsequent trigger delete by name operation resulted in a TRIGDEFBAD if the name targeted a trigger installed as part of the current load operation. In addition, concurrent MUPIP REORG works appropriately with GT.M triggers. Previously, in rare situations, a concurrent MUPIP REORG could cause GT.M trigger operations to issue TRIGDEFBAD errors. These issues were only observed in the GT.M development environment, and were never reported by a user. (GTM-8568)

- V6.3-001 $SELECT() does not pre-evaluate any expressions. A regression introduced in V6.2-002A related to GTM-8376 resulted in inappropriate pre-evaluation of arguments in some cases, especially when both FULL_BOOLEAN and gtm_side_effects modes were off. This caused inappropriate behavior such as errors in $SELECT() formulations intended to prevent execution of inappropriate indirection or $INCREMENT() acting on a global variable. (GTM-8572)

- V6.3-001 When the argument of an IF command is a literal value or expression, the GT.M compiler generates code to set $TEST appropriately and ignores the rest of the line. When the argument of a command postconditional is a literal value or expression, the GT.M compiler evaluates the postconditional and either generates code for an unconditional command or omits generation for the command. Previously, the computation was performed at run time. Note that literal postconditionals evaluating to 0 result in smaller object modules than literal postconditionals evaluating to non-zero values. (GTM-8573)

- V6.3-001 As part of compilation, GT.M optimizes unary operations, and binary operations, where both operands are literals; cases where the first operand is an empty string, divide and integer divide (/ or \) by zero (0) are exceptions. In addition, it treats $ZCHSET and $ZVERSION as compile time constants. Please observe the following cautions: ensure you compile with the same GT.M version, $gtm_chset, $gtm_local_collate, $gtm_patnumeric, $gtm_pattern_file and $gtm_pattern_table values (or lack thereof) as those used to run your application, and use variable operands, indirection or XECUTE for operands used with pattern match (?) or sorts-after (]]) if

the application changes the run time values controlled by those environment variables. Note that the compiler detects a few errors at slightly different points which may change some messages, hopefully for the better. In addition, this change prevents a possible segmentation violation (SIG-11) in V6.3-000[A] when attempting to MUMPS -RUN of a routine with no current object module when the routine uses ZWRITE. (GTM-8579)

- V6.3-001 MERGE permits its target to hold the maximum number of subscripts supported by GT.M (currently 31). Beginning with V6.1-000 the change associated with GTM-7867 could cause inappropriate MAXNRSUBSCRIPTS errors when the source and the target were both global variables, most likely when the source had many subscripts. The workaround was to MERGE the source into a local variable and then MERGE from there to the actual target. (GTM-8583)

- V6.3-001 $QLENGTH(), $QSUBSCRIPT() and $ZCOLLATE() use ZWRITE format to report the namevalue in any NOCANONICNAME error. Previously, a <NUL> byte in the input resulted in a GTMASSERT. (GTM-8584)

- V6.3-001 GT.M defers certain timed operations while performing a $ZCONVERT(); previously, the function could hang if interrupted by an timed operation that invoked non-reentrantsystem memory management services. This issue was only observed in the GT.M development environment, and was never reported by a user.(GTM-8590)

- V6.3-001 Deviceparameters on the OPEN command for SOCKET device containing open sockets can modify the ZFF and delimiters of the current socket; previously, these could only be changed with a USE command. In addition, $KEY and $ZB for SOCKET devices appropriately return UTF-8 representations of the characters; previously, if the device character set was UTF-16[BE|LE], the terminator representations were inappropriately also encoded in UTF-16[BE|LE]. (GTM-8593)

- V6.3-001 In M mode, $ASCII() of literal characters returns the correct value. In V6.3-000/-000A, this returned an incorrect value, typically -1 (it worked correctly for ASCII literal characters, for variable arguments, and in UTF-8 mode). Note that this is an edge case: instead of coding $ASCII() of a literal, one would normally just use the value, e.g., 65 instead of $ASCII("A"), and furthermore, the supported character set for literals in M programs is a subset of ASCII, whereas the issue affected literals corresponding to the non-ASCII characters $CHAR(128) through $CHAR(255). (GTM-8595)

- V6.3-001 ZSHOW "G" and $VIEW("GVSTAT") report a count BTD, which for database regions that use the BG access method is the number of times a global buffer has transitioned from an clean (unmodified) state to a dirty (modified) state. For database regions that use the MM access method, BTD is zero. (GTM-8639)

- V6.3-001 GT.M accepts NUL characters ($CHAR(0)) within literals used in indirection and XECUTE; previously, it generated errors for that character. (GTM-8640)

- V6.3-001 GT.M now reduces the active memory usage when a process uses a large amount of memory then subsequently uses a significantly reduced amount. Previously, active memory usage

was distributed across the whole memory segment. In addition, $VIEW("SPSIZE") now returns three sizes (as comma separated values): the total amount of space allocated to the heap, amount of heap space in use, and amount of heap space reserved. The reserved space is used to reduce the active memory usage as mentioned above. GT.M now extends memory used for local variables more frequently when garbage collection does not reclaim a significant amount of space.(GTM-8672)

- V6.3-001 Name-level $ORDER() on globals maintains $REFERENCE analogously to other $ORDER() invocations, that is: by reflecting the first argument unless a subsequent global reference in the second argument takes precedence; previously, it left $REFERENCE empty except when the second argument was a variable. This also applies to $ZPREVIOUS(), which is a deprecated way to do a $ORDER(,-1). (GTM-8679)

- V6.3-001 GT.M protects OPEN commands against the possibility they don't complete due to an interrupt or device failure; previously, there was a very small window where external actions such as <CTRL-C>, MUPIP INTRPT, or a device disconnect could leave a device partially set up, which could cause a subsequent segmentation violation (SIG-11). This was only encountered in the GT.M development environment and was never reported by a user. (GTM-8689)

- ZSHOW/ZWRITE work correctly even if they encounter a relatively rare heap management action. Previously, these commands could terminate with a segmentation violation (SIG-11) in these rare cases. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8705)

System Administration

- V6.3-001 MUPIP BACKUP and INTEG wait approximately one minute for any kill-in-progress to clear. Previously, the times exceeded the documented one minute time by three times for INTEG and an amount for BACKUP that was a function of the number of regions with kill-in-progress indicators; abandoned indicators showed this issue most strongly. These problems were never reported by users and were only seen in the GT.M development environment. (GTM-6598)

- V6.3-001 GT.M flushes dirty journal and database buffers to disk in a timely manner in a replicated environment. Additionally, the Source Server recovers from an unflushed journal buffer situation by taking on the task of flushing, if needed, every eight (8) seconds while waiting for a journal record in the journal file. It logs a "REPL_INFO : Source server did flush of journal file" message to record such an event. Previously, it was possible in a rare case, involving journal file switches and process exits, for the buffers to stay unflushed causing the Source Server to issue a "REPL_WARN: Check for problems with journaling" alert every 50 seconds. The workaround for this situation was to start a new process that did an update or shut the source server down. (GTM-7593)

- V6.3-001 GDE ADD, CHANGE and TEMPLATE for REGION objects recognize the -[NO]LOCK_CRIT qualifier; MUPIP SET recognizes a -[NO]LCK_SHARES_DB_CRIT qualifier. Both control whether LOCK actions share the same resource and management as the database or use a separate resource and management. The GDE choice only affects database file creation with MUPIP CREATE. GDE SHOW -ALL and -REGION, and DSE DUMP -FILEHEADER each display the choice, which defaults to Sep(arate)/FALSE. Previously LOCK actions used the same resource and manager. While we expect this to have either no effect or a positive effect on performance depending on the application use patterns, it is different by default, so you should be aware of this change. (GTM-7729)

- V6.3-001 MUPIP REPLICATE -CHANGELOG waits for up to twenty-five seconds for confirmation from Source and Receiver Server processes that the change succeeded (it may not for a variety of reasons). Previously, only MUPIP REPLICATE -RECEIVER -CHANGELOG waited; MUPIP REPLICATE -SOURCE -CHANGELOG did not. (GTM-7837)

- V6.3-001 MUPIP EXTRACT appropriately handles the case where a concurrent process updates a spanning node that MUPIP EXTRACT is processing; previously, this situation could cause a segmentation violation (SIG-11). Note that running an EXTRACT without -FREEZE and with concurrent activity produces an inconsistent output (GTM-7922)

- V6.3-001 MUPIP FREEZE -ON -ONLINE freezes updates to the database file, but allows updates to memory and journal files to continue. As for normal freezes, the Online Freeze is removed by a MUPIP FREEZE -OFF. Online Freeze may only be used on regions with the BG access method.

  In the Online Freeze state, GT.M prevents casual database updates from occurring, including background flushing and timed epochs. However, certain conditions require updates to the database

file, including full database buffers, journal file switches, and database file extensions. The -[NO]AUTORELEASE option may be used with the -ON option to select the behavior in these conditions, with -AUTORELEASE being the default. If a GT.M process autoreleases an Online Freeze, it sends an OFRZAUTOREL message to the operator log, all processes will be allowed to write to the database file, and a subsequent MUPIP FREEZE -OFF will warn that an Online Freeze had been removed. In this case any database copy or snapshot should be considered suspect and retried. If -NOAUTORELEASE is specified, memory updates will be suspended rather than release the freeze. If a process encounters this situation while holding a critical resource, it will send an OFRZCRITSTUCK message to the operator log and wait, which will prevent other operations on the region. When the Online Freeze is removed by a MUPIP FREEZE -OFF, the waiting process will send a OFRZCRITREL message to the operator log.

Some commands which cannot run with an Online Freeze, e.g., MUPIP BACKUP and MUPIP SET -JOURNAL, will either autorelease or issue an OFRZACTIVE error, depending on the -[NO]AUTORELEASE option used to set the freeze. Other commands, e.g., MUPIP EXTEND, MUPIP REORG -TRUNCATE, and MUPIP INTEG -ONLINE, will either autorelease or hang until the Online Freeze is released.

A MUPIP FREEZE -OFF must always follow a MUPIP FREEZE -ON -ONLINE, even in the case of an autorelease, to ensure that normal operations are resumed. In the case of an autorelease, the MUPIP FREEZE -OFF command will report a OFRZNOTHELD warning.

To maximize the time that updates to memory may continue, MUPIP FREEZE -ON -ONLINE flushes all dirty buffers to disk and performs a journal file switch, but it does not perform a database extension. If a database file is nearly full, the user should consider doing a database file extension before the Online Freeze. When the previous FREEZE operation was -ONLINE, a MUPIP FREEZE -OFF flushes any dirty buffers to disk and performs a journal file switch. These operations are performed in such a way as to minimize impact to processes doing memory updates, but they do involve performing epochs, so there may be some delay to other processes.(GTM-8362)

- V6.3-001 MUPIP replication servers accept changes to their log file destinations immediately after the last change; previously, they occasionally required a wait between changes. (GTM-8373)

- V6.3-001 MUPIP DUMPFHEAD [-FILE <file-name>][-REGION <region-list>] provides a way to get substantially the same information as DSE DUMP -FILEHEADER, but in the same format as provided by %PEEKBYNAME, and without connecting to database files. It is both lighter weight than DSE and avoids the need to use DSE, for which operator error can have serious consequences. The formatting is more regular than that of DSE. As MUPIP DUMPFHEAD does not open shared memory, values reported for dynamic fields that are in shared memory may be stale. Because MUPIP DUMPFHEAD is implemented in MUMPS, application code can call getfields^%DUMPFHEAD(varname,dbfilename), where varname is a local variable name passed by reference, and dbfilename is the name of a database file, to generate the records dumped by MUPIP. Note that this facility supersedes ^%DSEWRAP, which is deprecated, is not updated or tested, and eventually will be withdrawn. (GTM-8447)

- V6.3-001 MUPIP LOAD -FORMAT=BINARY validates the keys in the extract records and reports any errors it detects before skipping the bad record and any following records in the block. Previously, LOAD of a binary extract did not validate incoming keys. (GTM-8542)

- V6.3-001 The Source Server handles reconnects after errors gracefully. Previously the Source Server could exit with a REPLBRKNTRANs error after an error like REPLNOTLS that causes the Source and Receiver Servers to disconnect in middle of renegotiating the replication starting point. This issue was identified in FIS testing and has not been reported by any customers. (GTM-8547)

- V6.3-001 MUPIP REORG -ENCRYPT works correctly when reissued after a prior invocation of the same command was interrupted. Previously, if a MUPIP REORG -TRUNCATE was run in between and did truncate the database, it was possible for the reissued MUPIP REORG -ENCRYPT to incorrectly succeed even though it did not finish the (re)encryption. This was particularly evident if one ran a MUPIP SET -ENCRYPTIONCOMPLETE command on the same database which correctly indicated the encryption as incomplete. This issue was only observed in the GT.M development environment, and was never reported by a user. A workaround for this was to run a MUPIP EXTEND on the database and reissue the MUPIP REORG -ENCRYPT. (GTM-8564)

- V6.3-001 Enabling sync_io for journaling works correctly on XFS filesystems configured with 4KiB sector sizes. Previously, the Source Server could experience a REPLFILIOERR error with the message "Error in reading jfh in update_eof_addr" when reading from journal files. (GTM-8582)

- V6.3-001 MUPIP JOURNAL -ROLLBACK -FORWARD correctly rolls the database forward in the case one region has a journal file with very limited update activity (less duration than the epoch interval of that region). Previously, it was possible, in the unlikely case of a crash (where the journal files were not cleanly shutdown) where a region had only a single epoch matching the safe restore time determined across all regions, for ROLLBACK to inappropriately set the database file header as if it had restored updates beyond those it actually restored, which could cause a subsequent replication restart to miss updates. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8599)

- V6.3-001 The error messages for JNLTRANSLSS and JNLTRANSGTR now include the transaction numbers in the database file header and journal file header. The accompanying JNLOPNERR prints the name of the database and journal files. Previously, when issued alongside a JNLSENDOPER message, JNLOPNERR missed information about the database file, and there were no transaction numbers for JNLTRANSLSS or JNLTRANSGTR. The error JNLREADEOF omits the journal file name as the accompanying JNLEXTEND message prints this information.[/p][p]The error messages JNLBADRECFMT, JNLVSIZE, and CRYPTJNLMISMATCH include context information when a process fails to open a journal file for thefirst time. Previously, these error messages did not include some or all of the context information. (GTM-8609)

- V6.3-001 The Receiver Server handles unusual protocol messages appropriately. Previously, in extremely rare circumstance, the Receiver Server could mishandle such a message and terminate with a segmentation violation or REPLTRANS2BIG error preceded by numerous

"Received UNKNOWN message" messages. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8610)

- V6.3-001 MUPIP LOAD does not terminate abnormally with a segmentation fault (SIG-11) when delivering the MAXSTRLEN error nor does MUPIP LOAD treat a 12 byte header as a MAXSTRLEN error. Previously, a 12 byte header line in a GO or ZWRITE extract could cause GT.M to terminate abnormally with a segmentation fault (SIG-11). The workaround for the short header was to edit it in order to pad the length. (GTM-8612)

- V6.3-001 The REQROLLBACK message indicates that required -ROLLBACK also requires the -NOONLINE qualifier.(GTM-8614)

- V6.3-001 MUPIP RESTORE issues an IOEOF error and exits with a non-zero status when supplied with a truncated backup file. Previously, it used to prompt for the next volume to be mounted and later issue a SYSTEM-E-UNKNOWN error and incorrectly exit with a zero status indicating normal exit. Additionally, it cleans up database semaphore ipcs in case of errors; previously left two semaphores per database in case of errors.(GTM-8629)

- V6.3-001 GT.M considers available process groups when determining permissions based on group membership for IPCs and files, like journals and snapshot files. Previously, when GT.M failed to determine group membership, GT.M inappropriately removed owner access to the IPCs that it created resulting either a PERMGENFAIL error or DBFILERR error followed by the supplemental text "Error with database control semctl SETVAL". (GTM-8654)

- V6.3-001 The GT.M reference encryption plugin is compatible with OpenSSL 1.1.0. Previously, the plugin would not compile with OpenSSL 1.1.0. (GTM-8656)

- V6.3-001 GT.M replication appropriately handles the case where a receiving Supplementary Instance (P) connects to a non-supplementary Originating Instance (A) for the first time and then reconnects with no intervening updates. Previously, if the A->P connection occurred twice with no intervening update, replication from P to another receiving Supplementary Instance (Q) failed with a STRMSEQMISMTCH error. (GTM-8657)

- V6.3-001 GT.M makes more information available to system profiling tools such as perf. [x86_64 Linux] (GTM-8660)

- V6.3-001 The GT.M reference encryption plugin Makefile copies the pinentry.m routine into $gtm_dist/plugin/gtmcrypt and the GT.M reference encryption plugin pinentry program includes $gtm_dist/plugin/r in the gtmroutines search path. Previously, if the encryption plug-in source archive was not extracted in $gtm_dist/plugin/gtmcrypt, the custom pinentry program failed to load the pinentry routine and the user would be prompted via the system default pinentry program.The GT.M Encryption plugin properly compiles when GT.M is installed without Unicode support. Previously, this would result in the Makefile exiting with an error. The GT.M encryption plugin

includes support for loopback pinentry mode (available starting with GnuPG 2.1.12) which simplifies unattended passphrase handling.(GTM-8668)

This page is intentionally left blank.

Other

- V6.3-001 The GT.M compiler accepts input with <CR><LF> line termination (common on some non-POSIX Operating Systems); previously, it did not. Our thanks to the membership of the May 2016 "Hacking GT.M" workshop for this change. (GTM-4283)
- V6.3-001 Update Process Reader Helpers operate correctly. Previously, under rare conditions the Reader Helper would encounter a segmentation violation (SIG-11). This was only encountered in the GT.M development environment, and was never reported by a user. (GTM-6085)
- V6.3-001 Code using literals on Linux on x86_64 is faster, and the generated object file is smaller than previously. Compilation is also faster, which should, in turn, speed up operations using indirection and XECUTE. As with any performance enhancement, actual benefit will vary, depending on the extent to which application code uses constructs that benefit from this change. [x86_64 Linux] (GTM-6332)
- V6.3-001 The gtmsecshr wrapper on AIX uses the standard system timezone taken from /etc/environment for the timestamps of any syslog entries it generates. It also passes this timezone on to gtmsecshr to use for its entries as well. Previously, syslog timestamps from the AIX gtmsecshr wrapper could be either in the timezone of the process that started gtmsecshr or UTC, depending on whether environment variables had been cleared at the time of the error or not. [AIX] (GTM-7778)
- V6.3-001 Critical section acquitions are slightly more efficient. [x86_64 Linux] (GTM-8430)
- V6.3-001 GT.M manages time-related tasks in a more lightweight fashion. On heavily loaded systems with large numbers of processes, this should reduce the number of interrupts and context switches that the operating system needs to process. In addition, GT.M processes time-related tasks in a timely fashion. Previously, in rare conditions, timed operations (e.g., HANG) could be delayed up to eight seconds. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8544)
- V6.3-001 A GT.M process functions correctly when its M data heap exceeds 2GiB. Previously, such a process could experience damage to internal data structures, leading to incorrect process behavior including process termination with segmentation violations (SIG-11). This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8561)
- V6.3-001 The GT.M configure installation script always creates the plug-in routine source and object file directories. Previously, when installing over an existing directory the installation script did not create the plug-in routine source and object file directories. (GTM-8562)
- V6.3-001 When directed to install GT.M with UTF-8 support, the configure script defaults to the C.UTF-8 locale when systems have none defined. Previously, the configure script terminated

prematurely on such a condition. This was only seen by users building GT.M from source in a restricted build environment. (GTM-8567)

- V6.3-001 When starting up, the replication Receiver Server only issues a NOMORESEMCNT message to the syslog if it is the first process to determine that more than 32Ki GT.M processes have run in that instance. Previously, in very rare cases, it could inappropriately log a duplicate message. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8569)

- V6.3-001 The GT.M compiler reduces the size of object modules by not placing literals in an object module once the compilation eliminates a need for them. Previously, it sometimes left obsolete literals in the object files; in V6.3-000[A], GTM-7762 made this a more significant issue. (GTM-8570)

- V6.3-001 ^%GI appropriately handles empty string data values in GO format input. In V6.3-000 and V6.3-000A, ^%GI ignored such input, leading to incorrect loads. The workaround was to use ZWR format or MUPIP LOAD. (GTM-8571)

- V6.3-001 ^%PEEKBYNAME() handles cases where the type of data it is to access is an array of known types by returning a comma delimited list of the array elements; for example, $$^%PEEKBYNAME("sgmnt_data.tp_cdb_sc_blkmod",base) returns a string in the form "0,0,0,0,0,0,0,0" because the arguments identify an array of 8 integers (type int). Previously, this kind of invocation produced a BADZPEEKFMT error. (GTM-8578)

- V6.3-001 GT.M briefly defers certain operations while processing character input; previously, the non-reentrant system memory management services used in character-based input could hang a process if their timed operations also invoked memory management services. (GTM-8598)

- V6.3-001 MUPIP JOURNAL accepts multiple EOF records in the same journal file. Multiple EOF records only occur when the last process to halt out of a journaled database terminates abnormally just before cleanly shutting down the journal file. Note that FIS strongly recommends against using kill -9 on any GT.M process performing database updates. Previously, MUPIP JOURNAL produced a GTM-E-JNLUNXPCTERR error when it encountered multiple EOF records in a journal file. (GTM-8602)

- V6.3-001 GT.M prevents certain timed operations during external calls. Previously, in an environment using encrypted databases,an external call could cause the process to hang due to an invocation of non-reentrant memory management system services also invoked by the encryption plug-in. (GTM-8615)

- The gtm_coredump_filter environment variable specifies the mappings of the process address space for a GT.M process, with the bits having the same meaning as those specified for /proc/<pid>/coredump_filter in "man 5 core". If unspecified, GT.M uses a value of 0x73; a value of -1 prevents GT.M from modifying the coredump_filter value. A running process can change its coredump_filter by writing to the file /proc/<pid /coredump_filter, and can query the current value by reading that file. [Linux x86_64] (GTM-8632)

- V6.3-001 Processes handle a large volume of TP updates appropriately; previously, a long-running process, such as an Update Process, adding new global nodes could eventually inappropriately terminate with a GTMASSERT2. (GTM-8637)

- V6.3-001 GT.M appropriately handles the case of a process receiving a MUPIP STOP while another process has recently been performing a MUPIP REORG -ENCRYPT to change the encryption keys. In V6.3-000[A] this combination could rarely cause the stopped process to terminate with a GTMASSERT2 error. This caused no problems for the database. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8642)

- V6.3-001 GT.M more accurately maintains the count of the number of processes attached to a database file $$^%PEEKBYNAME("node_local.ref_cnt",<region>) or "Reference count" in DSE DUMP -FILE output. Note that GT.M does not rely on this field, which exists to supply operational information to the user, but which may be inaccurate if processes are subject to kill - 9 - something that FIS recommends against and that GT.M does not guard against. Previously, in case the database has the quick-rundown mode enabled (MUPIP SET -QDBRUNDOWN), this counter could become inaccurate even without interference from kill -9. (GTM-8645)

- V6.3-001 When GT.M exits with a fatal error, such as GTM-F-MEMORY, it cleans up any empty file it was in the process of creating at the time of the failure. This applies to GT.M OPEN, DSE OPEN and a number of MUPIP functions which create files. Previously, this unusual situation left an empty file and in some cases the process terminated with a segmentation fault (SIG-11). (GTM-8659)

- V6.3-001 GT.M correctly marshals parameters when calling out to external Java libraries and programs. Previously GT.M over allocated the parameter space and in rare situations an overrun could occur. This problem was never reported by a customer and was only seen in development. (GTM-8669)

- V6.3-001 DSE FIND -KEY reports the correct path leading to the input key. For versions V6.0-000 to V6.3-000A, due to a regression introduced by GTM-6341, it sometimes reported an incorrect Global tree path if the input key is not found in the database file. (GTM-8676)

- V6.3-001 ^%TI works as documented; previously, it rejected some documented forms. (GTM-8686)

This page is intentionally left blank.

More Information

Additional information for GTM-6838 - Asynchronous database IO

Released as field test grade functionality in a production release, asynchronous IO is an option for database segments using the BG access method; previously GT.M performed only synchronous I/O through the file system cache.

- **$$^%PEEKBYNAME("sgmnt_data.asyncio",**<region-name>**)** - returns TRUE (1) if the region has asynchronous I/O enabled and FALSE (0) if it does not.
- **$$^%PEEKBYNAME("node_local.wcs_wip_lvl",**<region-name>**)** - returns the number of blocks for which GT.M has issued writes that it has not yet recognized as complete.

DSE DUMP -FILEHEADER reports the above information for a region as follows:

- **Async IO** - whether AsyncIO is ON or OFF for the database
- **WIP queue cache blocks** - the number of blocks for which GT.M has issued writes that it has not yet recognized as complete

GDE ADD, CHANGE and TEMPLATE commands accept the following qualifier for SEGMENT objects with an access method of BG:

- -[NO]ASYNCIO specifies whether to use asynchronous I/O for a database file; the default is NOASYNCIO.

MUPIP SET recognizes the -[NO]ASYNCIO qualifier. As there are two MUPIP SET qualifiers beginning with "A", MUPIP no longer recognizes "-A" to specify "-ACCESS_METHOD" - please use the recommended minimum of four characters (-ACCE). Also, when a MUPIP SET command changes the database access method, it reports the change; previously it made the change silently.

Database files have an empty database block logically after the last usable database block; previously this location had an empty 512 byte block. Because of this change, downgrading a database using V6.3-001 or later for use by a version prior to V6.3-001 requires a MUPIP DOWNGRADE -VERSION=V63000A. This syntax uses V63000A to specify DOWNGRADE change the terminating block from having the current size of a database block to using a 512-byte size used by all releases up to V6.3-000A, and should be used regardless of which release older than V6.3-001 you plan to use after the DOWNGRADE.

For Linux x86_64, the gtm_aio_nr_events environment variable controls the number of structures a process has per global directory to manage asynchronous writes, and therefore determines the number of concurrent writes a process can manage across all regions within a global directory. If not specified, the value controlled by gtm_aio_nr_events defaults to 128. If a process encounters a situation where it needs to perform an asynchronous write, but has no available slots with which to manage an additional one, it either falls back to synchronous writing if the write is blocking other actions, and otherwise defers the write until a slot becomes available as other writes complete. Linux allocates the structures on a system-wide basis with the setting of /proc/sys/fs/aio-max-nr. Therefore you should configure

this parameter to account for the needs (as determined by gtm_aio_nr_events or the default) of all processes using asynchronous I/O. When processes use multiple global directories with asynchronous I/O, their need for the system resources increases accordingly. For example, if an environment runs 10,000 processes each of which open two global directories and /proc/sys/fs/aio-max-nr is set to a value of 200,000 then gtm_aio_nr_events needs to be set to a value <= 200,000 / (10,000 * 2) = 10. Conversely if gtm_aio_nr_events is set to a value of 20, then aio-max-nr needs to be bumped up to (10,000 * 2 * 20) = 400,000. GT.M captures the number of errors encountered when attempting to write database blocks for a region, and, barring problems with the storage subsystem, hitting an asynchronous write limit would constitute primary (probably only) contribution to that value, which you can access with the following:

- **$$^%PEEKBYNAME("sgmnt_data.wcs_wterror_invoked_cntr",<region>)**

The performance characteristics of asynchronous IO are likely to be quite different from the traditional sequential IO. Although asynchronous IO in theory should be more efficient than synchronous IO by eliminating the need for the UNIX file buffer cache and eliminating certain filesystem locks, in practice asynchronous IO is likely to emerge from the starting gate under-performing synchronous IO because of the years that synchronous IO has been the common IO model operating systems and filesystems have had used by applications. So, you should anticipate extensive benchmarking and tuning for your application to achieve the best performance it can with asynchronous IO. Some notes and observations that we have to share:

- As asynchronous IO dispenses with the UNIX file buffer cache, GT.M global buffers are the sole caching mechanism. To make asynchronous IO perform well, you will likely need to increase the number of global buffers considerably. With GT.M's limit of 2GiB per shared memory segment, a database segment with 4KiB blocks has a limit of almost two million global buffers.

- A large number of global buffers potentially implies a large number of dirty global buffers to be flushed at an epoch. You should investigate the impact on application response time of GT.M epoch tapering vs. turning off epoch tapering and using a separate stand-alone process that executes a line of code such as: **for set x="" for set x=$view("gvnext",x) quit:""=x view "dbflush":x,"dbsync":x,"epoch":x hang** □ where □ is a number that causes each region to be flushed at an appropriate interval. If you choose this option, remember to turn off epoch tapering, and to set the epoch interval in the file header to be large enough to prevent application processes from performing epochs, and consider scripted timely switching of journal files by other than application processes (switching journal files involves an epoch).

- On AIX, consider mounting file systems with the CIO mount option. The CIO mount option drops support for the file buffer cache (unused by asynchronous IO), and also eliminates a lock that is a potential bottleneck to GT.M performance on the AIX jfs2 filesystem.

- Limited experience with solid-state storage (SSDs) on Linux in the GT.M development environment suggests a considerable difference in asynchronous IO performance on the same underlying hardware, with f2fs performing better than xfs, which in turn performed better than ext4.

- On Linux, which does not have a mount option such as AIX's CIO, even when GT.M uses asynchronous IO, a command like cp (used under the covers by MUPIP BACKUP) can still use synchronous IO. Owing to what we expect is a race condition between the two IO models, MUPIP BACKUP -DATABASE on Linux has in our testing on rare occasions created backups with database

errors in them. The errors were observed most often with an ext4 filesystem, less frequently on xfs, and never on f2fs; however, we must treat it as if all filesystems are potentially vulnerable. FIS recommends using MUPIP BACKUP -BYTESTREAM if your application uses asynchronous IO.

In GT.M development, we have not benchmarked asynchronous IO on the types of storage commonly used for enterprise scale applications (as workloads vary widely, we do not routinely benchmark workloads in development). Please consider the above observations in this light.

Additional information for GTM-6699 - Monitoring of shared database statistics

GT.M provides a fast and efficient mechanism for processes to share their database access statistics for other processes to monitor. Processes opt in or out with the VIEW "[NO]STATSHARE" command, defaulting to VIEW "NOSTATSHARE". At process startup, a value of 1, or any case-independent string or leading substrings of "TRUE" or "YES" in the environment variable gtm_statshare provides an initial setting of VIEW "STATSHARE". When a process changes whether it is opting in or out, there is no change to the output of a ZSHOW "G" within that process. GT.M does not permit this keyword of the VIEW command within a TP transaction. Monitoring the statistics of other processes does not require a process to opt-in to sharing its own statistics.

Processes that have opted-in share their statistics as binary data in database files located in the directory specified by the gtm_statsdir environment variable. If you do not explicitly define this environment variable for a process, GT.M defines this to the evaluation of $gtm_tmp, which defaults to /tmp. All processes that share statistics MUST use the same value for $gtm_statsdir. FIS suggests that you point gtm_statsdir at a tmpfs or ramfs on Linux, and a filesystem on a ram disk on AIX. These database files have a name derived from the user defined database file name and a .gst extension. They are not usable as normal database files by application code, except to read statistics. GT.M automatically creates and deletes them as needed. Under normal operation, applications do not need to manage them explicitly. The mapping of ^%YGS to statistics database files is managed by GT.M within global directories, transparently to applications. As described below, the ^%YGBLSTAT utility program gathers and reports statistics from nodes of ^%YGS(region,pid).

Labels in the ^%YGBLSTAT utility program gather and report statistics, presenting both a high level API and a low level API. While we intend to preserve backward compatibility of the high level API in future GT.M releases, we may change the low level API if and when we change the underlying implementation. A call to a label in ^%YGBLSTAT does not in any way slow the execution of other processes. Because the gathering of statistics is not instantaneous, and processes concurrently open database files as well as close them on exit, and may turn their participation in statistics monitoring on and off, statistics typically do not show a single moment in time, as statistics change during the short time interval over which they are gathered.

In the following, an omitted response or argument is equivalent to "*".

The high level API implemented by $$STAT^%YGBLSTAT(expr1[,expr2[,expr3[,expr4]]]) reports global variable statistics and has arguments as follows:

- expr1 (treated as an intexpr - coercing an expr to an integer is equivalent to +(expr)) specifies the PID of a process on which to report; if such a process does not exist, has not opted in, or no database

file mapped by expr3 and expr4 includes statistics for such a process, the function returns an empty string. Specifying "*" as the value of expr1 returns the aggregate statistic(s) specified by expr2 for all processes whose statistics are included in the database file(s) of the region(s) specified by expr4 within the global directory specified by expr3, or the empty string if there are no statistics to report for any process.

- expr2 specifies the statistic(s) to report as follows:
  - If expr2 is a single statistic, e.g., "LKF", the function returns the requested value as an integer
  - If expr2 is a series of comma-separated names of statistics, e.g., "DTA,GET", the function returns a string with each requested statistic in ZSHOW "G" order, e.g., "GET:3289,DTA:598...", rather than in the order in which they appear within the specifying argument.
  - If expr2 is omitted, or consists of the string "*", the return value reports all statistics formatted like the ZSHOW "G" statistics for a single region, e.g., "SET:563,KIL:39,GET:3289,DTA:598...
- expr3 specifies a global directory file name (producing a ZGBLDIRACC error if such a global directory is not accessible); if unspecified, the utility defaults this value to $ZGBLDIR of the invoking process.
- expr4 specifies the name of a region (producing a NOREGION error if no such region exists in the global directory expr3); if expr4 is unspecified, or the string "*", the function returns statistics for the process or processes summed across all regions of the global directory explicitly or implicitly specified by expr3.

When invoked as an interactive utility program using DO, ^%YGBLSTAT, prompts for:

- the process id (respond * for all processes)
- a comma separated list of the statistics desired (respond * for all statistics)
- the global directory to use
- region (respond * to report statistics summed across all regions).

When invoked from a shell, the command line is: mumps -run %YGBLSTAT [--help] [--pid pidlist] [--reg reglist] [--stat statlist] where:

- pidlist is a single pid, or "*" (quoted to protect it from expansion by the shell) for all processes currently sharing statistics.
- reglist is a single region name in the global directory specified by $gtmgbldir, or "*" to report statistics summed across all regions
- statlist is one or more comma separated statistics, or "*"
- When statlist specifies a list of statistics, %YGBLSTAT reports them in the same order in which ZSHOW "G" reports those statistics, rather than in the order in which they appear within the specifying argument.

$$ORDERPID^%YGBLSTAT(expr1[,expr2[,expr3]]) reports PIDs of processes that have opted in and recorded statistics. Its arguments are as follows:

- expr1 coerced to an intexpr specifies a PID such that the function returns the next PID after expr1 of a process that has opted in to be monitored and which has recorded statistics in any region(s) specified by expr3 from the global directory specified by expr2, or the empty string if expr1 is the last PID. A value of the empty string ("") for expr1 returns the first monitored PID meeting the specifications in expr2 and expr3.
- expr2 specifies a global directory file name (producing a ZGBLDIRACC error if such a global directory is not accessible); if unspecified or the empty string, the utility defaults this value to the $ZGBLDIR of the invoking process.
- expr3 evaluates to the name of a region (producing a NOREGION error if no such region exists in the global directory specified by expr2); if expr3 is unspecified, or the string "*", the function returns the PID for the next process after expr1 for any region of the global directory specified by expr2.
- Applications should not rely on GT.M returning the PIDs in a sorted or other predictable order: the order in which PIDs are returned is at the discretion of the implementation, and may change from release to release.

The low level API implemented by $$SHOW^%YGBLSTAT(glvn[,strexp]) reports raw statistics of a process and has arguments as follows:

- glvn specifies a node containing raw statistics for a process
- the raw data is stored in uniquely managed database files as nodes of ^%YGS(expr1,expr2) where:
  - expr1 evaluates to the name of a region in the current global directory (or the global directory of an extended reference), producing an UNDEF error, or, in NOUNDEF mode, an empty string, if no such region exists
  - expr2 coerced to an intexpr is a PID.
  - The data in the node is a series of binary bytes which are the raw statistics shared by a process
- strexp specifies statistics to report with the same interpretation as the expr2 parameter of $$STAT^%YGBLSTAT.
- $$SHOW^%YGBLSTAT() reports a zero value for any statistic whose name is unrecognized. This facilitates application code written for a version of GT.M that includes a statistic, but which also needs to run on an earlier version without that statistic.
- Because a process sharing statistics can exit or turn off sharing, deleting its node, between the time a monitoring process decides to access its statistics, e.g., finding it using $$ORDERPID^%YGBLSTAT() or $ORDER(^%YGS()), and the time the monitoring process performs the database access, any direct access to ^%YGBLSTAT should be wrapped in $GET().
- As raw statistics are binary data, processes in UTF-8 mode that gather and monitor statistics should use code with appropriate BADCHAR handling. Note that processes sharing statistics and processes

gathering statistics for monitoring and reporting need not run in the same UTF-8/M mode. As statistics sharing by processes is identical in M and UTF-8 modes, FIS suggests that processes gathering statistics run in M mode

Except as documented here for sharing and gathering statistics, FIS strongly recommends that applications not access statistics database files unless otherwise directed by your GT.M support channel.

As they do for unshared statistics, shared statistics reflect all database actions for a TP transaction, including those during RESTARTs. Because the sharing of statistics is not a database operation that modifies the relative time stamps GT.M uses to maintain serialized operation preserving the Consistency and Isolation aspects of ACID operation, statistics generated by a sharing process inside a transaction (TSTART/TCOMMIT) do not cause transaction restarts as a consequence of updates to shared statistics by other processes.

DSE DUMP -FILEHEADER reports the following information for a region:

- DB is auto-created TRUE or FALSE to indicate whether the database file is automatically created
- DB shares gvstats TRUE or FALSE to indicate whether the database supports sharing of statistics

By default, DSE does not map the regions used to store shared GVSTAT information, however the -STATS qualifier on a FIND -REGION=<region-name> command directs DSE to any existing statistics database associated with the named region. Note that these special purpose regions have a lower-case name that corresponds to their actual upper-case region name, and their databases tend not to exist unless they are in current use.

GDE, ADD. CHANGE and TEMPLATE commands accept the following qualifiers for REGION objects:

- -[NO]AUTODB specifies whether GT.M should automatically create the associated database file if it does not exist when a process first attempts to access it; it defaults to NOAUTODB. This feature is not intended to replace MUPIP CREATE, but rather permit somewhat easier management of temporary databases or process private databases.
- -[NO]STATS specifies whether GT.M should permit statistics sharing for this region; it defaults to STATS. This characteristic permits operational exclusion of statistics sharing for a region.

Any attempt to map globals with names starting with %Y produces a NOPERCENTY error, as this namespace is reserved by the standard for implementation use.

MUPIP commands with the exception of INTEG, SET and RUNDOWN do not apply to statistic databases; that is they are not available for operations such as BACKUP, EXTRACT or LOAD.

- INTEG recognizes the -STATS qualifier which directs it to integrity check any active statistics database associated with the region(s) specified for the command.
- RUNDOWN -FILE can be directed to a statistics database file and works even if the corresponding actual database file does not exist. MUPIP RUNDOWN with no argument removes any statistics database file resources associated with actual database file resources it can remove.

- SET recognizes the -[NO]STATS qualifier. Please refer to the GDE description above for a description of these database characteristics.

This page is intentionally left blank.

Error and Other Messages

CHANGELOGINTERVAL

CHANGELOGINTERVAL, ssss Server now logging to ffff with a IIII second interval

MUPIP Information: This message confirms a change to a replication server (ssss) by showing the current log file (ffff) and log interval (IIII)

Action: None Required

CRYPTNOMM

CRYPTNOMM, ffff is an encrypted database. Cannot support MM access method.

MUPIP Error: This error is triggered by an attempt to mark an MM database as encrypted with GDE or to switch an encrypted database from BG to MM with MUPIP SET. The MM access method is not supported for encrypted databases.

Action: Use the BG access method for encrypted files.

DBDUPNULCOL

DBDUPNULCOL, Discarding kkkk=vvvv key due to duplicate null collation record

MUPIP Error: This idicates that MUPIP LOAD discarded a key-value pair from a binary EXTRACT because it contained conflicting empty string subscripts. This can only happen is someone changes the "Null" subscript representation used by a database while it contains such subscripts. FIS recommends against such a change.

Action: Determine whether the described data has value and restore it, typically with a SET command, appropriately.

DBMISALIGN

DBMISALIGN, Database file xxxx has yyyy blocks which does not match alignment rules. Reconstruct the database from a backup or extend it by at least zzzz blocks.

MUPIP Error: This is an auxiliary message, and is preceded by a primary message.

Action: Follow the primary message description and action as specified in this manual.

DBNULCOL

DBNULCOL, NULL collation representation for record rrrr in block bbbb is RRRR which differs from the database file header settings of hhhh

DSE/MUPIP/Run Time Error: This indicates the database contains a record rrrr with an empty subscript ("Null" subscript) representation RRRR in block bbbb that is incompatible with the current setting hhhh for such representation. 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 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 many change due to ongoing updates, so this operation requires great care and familiarity with DSE.

DBTOTBLK Δ

DBTOTBLK, File header indicates total blocks is tttt but file size indicates total blocks would be eeee

MUPIP Information: This is an auxiliary message, and is preceded by a primary message.

Action: Follow the primary message description and action as specified in this manual.

GDECRYPTNOMM Δ

GDECRYPTNOMM, ssss segment has encryption turned on. Cannot support MM access method

GDE Error: This error is triggered by an attempt to mark an MM database segment ssss as encrypted with GDE. The MM access method is not supported for encrypted databases.

Action: Use the BG access method for encrypted files.

GDINVALID Δ

GDINVALID, Unrecognized Global Directory file format: ffff, expected label: eeee, found: bbbb

Run Time Error: This indicates that a version of the global directory file ffff does not match with the version expected by GT.M. The file might have been created by an incompatible GT.M version. If the text of eeee or bbbb contain non-graphic characters, GT.M replaces each of them with a period (.).

Action: Compare the labels eeee and bbbb. If the global directory was created by an earlier GT.M version, upgrade the file by loading and then saving the file using the GDE of the new GT.M version.

INVADDRSPEC Δ

INVADDRSPEC, Invalid IP address specification

Run Time Error: This indicates the IP address and/or port specified is not in a valid format.

Action: Verify and correct the IP address and port.

INVLINKTMPDIR Δ

INVLINKTMPDIR, Value for $gtm_linktmpdir is either not found or not a directory: dddd

Run Time Error: Indicates the process cannot access directory dddd, which it needs in order to do auto-relink as specified by its $ZROUTINES; the directory may not exist as a directory or the process lacks authorization to the directory.

Action: The directory specification comes from $gtm_linktmpdir if it is defined, otherwise from $gtm_tmp if that is defined; otherwise it defaults to the system temporary directory, typically /tmp. Either correct the environment variable definition or ensure directory dddd is appropriately set up. Note that all users of auto-relink for a directory normally need to use the same temporary directory for their relink control files.

INVMEMRESRV Δ

INVMEMRESRV, Could not allocate GT.M memory reserve (xxxx)

Images Warning: GT.M could not allocate xxxx KiB of reserve memory for handling and reporting out-of-memory conditions. Examine the subsequent messages for more information on why the memory reserve allocation failed.

Action: If $gtm_memory_reserve is too high, specify a lower value and retry. If the value is reasonable, determine what else is preventing the allocation (process or system limits or usage by other system components). Note that GT.M uses this reserve only when a process runs out of memory so it mostly requires address space and almost never requires actual memory.

IOEOF Δ

IOEOF, Attempt to read past an end-of-file

Run Time/MUPIP Error: This indicates that a READ command for a run-time system or a MUPIP command attempted to move past an end-of-file.

Action: Verify that the $ZEOF special variable is tested by the function betwee READs or that an EXCEPTION code string is assigned to handle EOFs. Alternatively, have your $ETRAP (or $ZTRAP) error handling deal with this error. The USE command has a REWIND deviceparameter that allows you to read from the beginning of the file without having to CLOSE and OPEN again, which may facilitate recovery from this error. Attempting to READ from a non-existent file not opened READONLY also causes this error. In the event of a MUPIP error, make sure the file being read is not corrupted.

JOBLVN2LONG Δ

JOBLVN2LONG, The zwrite representation of a local variable transferred to a JOB'd process is too long. The zwrite representation cannot exceed MMMM. Encountered size: LLLL

Run Time Error: This error indicates that the total length LLLL (in bytes) of the ZWRITE representation of the variable name, subscripts, and value exceeds the maximum MMMM supported by the

PASSCURLVN facility. Note that the ZWRITE representation contains the appropriate punctuation for any subscripts, the equal-sign and replaces any non-graphic characters with their $[Z]CHAR() representations.

Action: Consider whether the JOB'd process needs the variable(s) that exceed the maximum for PASSCURLVN - if not, they can be taken out of scope before the JOB command. Alternatively, pass them using global variables or a local SOCKET device.

JOBLVNDETAIL Δ

Last used version: V6.2-003

JOBLVNDETAIL, The zwrite representation of a local variable transferred to a JOB'd process is too long. The zwrite representation cannot exceed XXXX. Encountered size: YYYY

Run Time Error: The length of the zwrite representation of a local, (including the quotes, the '=', concatenate operator "_", and "$[Z]C()") has the length of YYYY which exceeds the maximum limit of XXXX.

Action: Please check the sizes of locals that needs to be sent and make sure their lengths are less than XXXX. For those big locals, consider using another mechanism such as sockets.

MUPJNLINTERRUPT Δ

MUPJNLINTERRUPT, Database file xxxx indicates interrupted MUPIP JOURNAL command. Restore from backup for forward recover/rollback.

MUPIP Error: This indicates that a MUPIP JOURNAL -ROLLBACK -FORWARD or a MUPIP JOURNAL -RECOVER -FORWARD did not proceed because a previous MUPIP JOURNAL command attempted on the database was terminated abnormally.

Action: Restore the database and journal files from a backup to proceed with the MUPIP JOURNAL -ROLLBACK -FORWARD or MUPIP JOURNAL -RECOVER -FORWARD.

NOPRINCIO Δ

NOPRINCIO, NOPRINCIO Unable to write to principal device

Run Time Fatal: This indicates that GT.M attempted to, but could not, READ from, or WRITE to, the principal device and therefore attempted to issue an appropriate error, for example, an IOEOF. However if the error handling does not prevent any and all subsequent READs and WRITEs to the no longer available principal device, the next subsequent I/O error shuts down the process immediately with a NOPRINCIO to prevent mysteriously lost output, or, worse, an indefinite loop.

Action: The NOPRINCIO error message is FATAL which does not drive device or trap handlers and terminates the process. This termination does not allow any application level orderly shutdown and, depending on the application may lead to out-of-design application state. Therefore FIS recommends

appropriate application level error handling that recognizes the preceding error and performs an orderly shutdown without issuing any additional READ or WRITE to the principal device. The most common causes for the principal device to cease to exist involve terminal sessions or socket connections (including those from processes started by inetd/xinetd). When the remote client terminates the connection, the underlying principal device becomes inaccessible making any subsequent attempt to READ from, or WRITE to, it hopeless. In the case of terminals, a user closing the window of a session without cleanly exiting from the GT.M process sets up the case that can drive this error.

NOTALLJNLEN Δ

NOTALLJNLEN, Journaling disabled/off for dddd regions

MUPIP Warning: This indicates that some or all regions do not have journal state ON.

Action: Ensure you have journaling enabled for all regions that require it; use MUPIP SET to enable journaling.

NOTALLREPLON Δ

NOTALLREPLON, Replication off for dddd regions

MUPIP Warning: This indicates that some or all regions have replication state OFF.

Action: Ensure you have replication on for all regions that require it; use MUPIP SET to enable replication.

OFRZACTIVE

OFRZACTIVE, Region aaaa has an Online Freeze

MUPIP Warning: A MUPIP operation has been requested while an Online Freeze is in place, but the operation can not be performed with an Online Freeze.

Action: The operation was not performed. Remove the freeze with MUPIP FREEZE -OFF and retry the operation.

OFRZAUTOREL

OFRZAUTOREL, Online Freeze automatically released for region aaaa

Operator log Warning: A process needed to modify the database file for region aaaa, which had an Online Freeze, but with AutoRelease selected. The process continued normally, modifying the file.

Action: Discard any database copy or snapshot made after the Online Freeze, as its contents are suspect. Perform a MUPIP FREEZE -OFF to clean up the prior Online Freeze. If the AutoRelease behavior is

not desired, try again with MUPIP FREEZE -ON -ONLINE -NOAUTORELEASE. If the cause of the AutoRelease is unclear, report this and the accompanying ERRCALL message to your GT.M support channel.

OFRZCRITREL

OFRZCRITREL, Proceeding with a write to region aaaa after Online Freeze while holding crit

Operator log Warning: A process previously encountered a OFRZCRITSTUCK condition, which has since been resolved.

Action: None.

OFRZCRITSTUCK

OFRZCRITSTUCK, Unable to proceed with a write to region !AD with Online Freeze while holding crit. Region stuck until freeze is removed.

Operator log Warning: A process needed to do a database write while holding a critical resource, but an Online Freeze was in place without AutoRelease enabled. No other process will be able to acquire the critical resource until the Online Freeze is removed.

Action: MUPIP FREEZE -OFF will remove the freeze and allow the process to continue, at which time it will send a OFRZCRITREL message to the operator log. This situation can be avoided by specifying MUPIP FREEZE -ON -ONLINE without the -NOAUTORELEASE option, or by including the -AUTORELEASE option.

OFRZNOTHELD

OFRZNOTHELD, Online Freeze had been automatically released for at least one region

MUPIP Warning: A MUPIP FREEZE -OFF command encountered at least one region which previously had an Online Freeze, but a process had AutoReleased it.

Action: The command cleaned up the region with the AutoReleased Online Freeze, and database operations are back to normal. However, any database file snapshots or copies made after the Online Freeze should be discarded, as processes likely will have written to the file since the AutoRelease. An OFRZAUTOREL message in the operator log will report which process performed the AutoRelease.

RECLOAD

RECLOAD, Error loading record number: nnnn

MUPIP Error: This message identifies a record nnnn that MUPIP could not LOAD and follows a message about the cause. If this message is Fatal, which it can be for BIN format, it produces a core file for diagnostic analysis.

Action: Address the cause or, for GO and ZWR format input files, examine the record with a text editor for possible correction or alternate action and for BIN format if fixing the cause does not resolve the error switch to ZWR format EXTRACT.

REPLLOGOPN Δ

REPLLOGOPN, Replication subsystem could not open log file LLLL : eeee. Logging done to OOOO

MUPIP Error: This indicates that MUPIP could not find, or did not have access permission to open, the log file LLLL, because of the error eeee. If there is another log file available (a previously opened file), MUPIP writes to the other log file OOOO. If there is no other log file available, MUPIP sends any remaining messages to /dev/null and terminates the replication server process.

Action: Check the log file permissions, and if permissions are correct, move the log file and specify that MUPIP should log to a log file which has appropriate access permissions.

REPLSTATEOFF Δ

REPLSTATEOFF, MUPIP JOURNAL -ROLLBACK -BACKWARD cannot proceed as database xxxx does not have replication ON

MUPIP Error: This indicates that a MUPIP JOURNAL -ROLLBACK -BACKWARD command cannot proceed because the specified database xxxx does not have replication state ON. In most situations, this error occurs when the journal file storage runs out of disk space.

Action: Ensure replication is turned ON for a database, before executing the MUPIP JOURNAL -ROLLBACK -BACKWARD command. If the database is in the WAS_ON state, refer to the "Recovering from the WAS_ON state" section in the Database Replication chapter of the Administration and Operations Guide. Alternatively, if replication was not in use on the database, use MUPIP JOURNAL -RECOVER.

REQROLLBACK Δ

REQROLLBACK, Error accessing database dddd. Run MUPIP JOURNAL -ROLLBACK -NOONLINE on cluster node cccc.

Run Time Error: This indicates that GT.M could not open a previously replicated database file dddd due to a prior improper shutdown on cluster node cccc. A GT.M process on cluster node ccccc may have failed to attach a database memory segment or it was terminated by a method other than MUPIP STOP.

Action: Perform MUPIP JOURNAL -ROLLBACK -NOONLINE to cleanup the instance file, database, and journal files before starting a source server on this instance.

RESRCINTRLCKBYPAS Δ

RESRCINTRLCKBYPAS, tttt with PID qqqq bypassing the ssss semaphore for region rrrr (ffff) currently held by PID pppp.

All GT.M Components Information: GT.M issues the RESRCINTRLCKBYPAS message to the system log as an indication it may not detect when the last process detaches from the shared resource and therefore may not rundown the database shared resources as it normally would. GT.M protects the actions of setting up and tearing down the shared resources associated with a database with a pair of semaphores. Because DSE, and LKE are tools for diagnosing issues, when they start and find they cannot acquire the semaphores after a reasonable number of tries, they proceed to open the database anyway because it is highly probable the database is already set up. When DSE and LKE bypass the semaphore acquisition, they leave the count of attached processes incorrect. When many processes terminate at the same time, typically because of a system shutdown, there can be significant contention for the semaphores that can cause their terminations to take an unusually long time. When this happens, and the count of remaining attached processes is significant, a process may skip the semaphore acquisition, again leaving the count of attached process incorrect. If either of these events occurs, GT.M issues the RESRCINTRLCKBYPAS message where tttt identifies the process type: "LKE", "DSE" or "GT.M"; qqqq is the bypassing process's PID; ssss identifies the semaphore type: "FTOK" or "access control"; rrrr is the region bypassed; ffff is the file corresponding to region rrrr; pppp is the PID of the process holding the semaphore.

Action: These messages when shutting down GT.M activity may indicate a need to complete the process by invoking a MUPIP JOURNAL -ROLLBACK -BACKWARD for replicated databases, a MUPIP JOURNAL -RECOVER -BACKWARD for unreplicated journaled databases and a MUPIP RUNDOWN for journal-free databases to get the database to a safe state; doing so as part of every shutdown is good practice.

RESRCWAIT Δ

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

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

Action: None required.

TPRESTART Δ

TPRESTART, Database mmmm; code: xxxx; blk: yyyy in glbl: zzzz; pvtmods: aaaa, blkmods: bbbb, blklvl: cccc, type: dddd, readset: eeee, writeset: ffff, local_tn: gggg, zpos: hhhh

Run Time Information: The UNIX environment variables GTM_TPRESTART_LOG_FIRST and GTM_TPRESTART_LOG_DELTA control the logging of TPRESTART messages. GTM_TPRESTART_LOG_FIRST indicates the number of TP restarts to log from a GT.M invocation. Once that many have been logged, every GTM_TPRESTART_LOG_DELTA TP restarts, GT.M logs a restart message. If GTM_TPRESTART_LOG_DELTA is undefined, GT.M performs no operator logging. The default value for GTM_TPRESTART_LOG_FIRST is 0 (zero), which leaves the control

completely with GTM_TPRESTART_LOG_DELTA. The facility that produces this message can serve as a diagnostic tool in developmental environments for investigating contention due to global updates. A zzzz of "*BITMAP" indicates contention in block allocation which might involve multiple globals. hhhh is the $ZPOSITION of the line of M code that caused the restart of the transaction; utilities leave this field blank.

Action: Disable, or adjust the frequency of, these messages with the mechanism described above. To reduce the number of restarts, consider changes to the global structure, or varying the time when work is scheduled. Consider whether the business and program logic permits the use of NOISOLATION.

TRIGINVCHSET Δ

TRIGINVCHSET, Trigger tttt for global gggg was created with CHSET=cccc which is different from the current $ZCHSET of this process

Trigger/Run Time Error: TRIGINVCHSET occurs when a process invokes a trigger on a global using a $ZCHSET that is different from the $ZCHSET used at the time of loading the first trigger on that global. GT.M implicitly uses the $ZCHSET of the first trigger on a global to invoke all triggers on that global. Note that tttt is the name of the first trigger on the global gggg-not necessarily the name of the trigger being invoked. cccc is the $ZCHSET of the process at the time of loading tttt on global gggg.

Action: Ensure that the process invoking a trigger on a global uses the same $ZCHSET that was used to load the first trigger on that global. If your application requires triggers in both M and UTF-8 modes, use different globals to load M mode and UTF-8 mode triggers.

ZATRANSERR

ZATRANSERR, The input string is too long to convert

Run Time Error: The first (expression) argument to a $ZATRANSFORM() produces a result that exceeds the maximum key length.

Action: Analyze the logic to determine if the argument is correct. If you need to produce translations that exceed the maximum key length, you must use $ZCOLLATE() or break them into chunks to avoid this error, Note that some transforms may use context such that selecting the chunks requires an understanding of the transform.

This page is intentionally left blank.