V6.0-000

# GT.M

## Release Notes

## Contact Information

GT.M Group
Fidelity Information Services, Inc.
2 West Liberty Boulevard, Suite 300
Malvern, PA 19355
United States of America

GT.M Support for customers: +1 (610) 578-4226
gtmsupport@fisglobal.com
Switchboard: +1 (610) 296-8877
Website: http://fis-gtm.com

## Legal Notice

Copyright © 201, 2014 Fidelity Information Services, Inc. All Rights Reserved

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

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

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

| Revision History | | |
|---|---|---|
| Revision 1.1 | 27 October 2014 | Corrected the description of GTM-7327. |
| Revision 1.0 | 19 September 2012 | First published version. |

# Table of Contents

# V6.0-000

## Overview

V6.0-000 extends several limits of GT.M database files

- The maximum supported key size increases from 255 to 1019 bytes.
- The maximum supported global variable node size is 1MiB regardless of database block size; previously, the key, value, and delimiters had to fit within one database block. This means that any local variable value can be assigned to a global variable without regard to the database file block size.
- The maximum number of blocks in a database file increases from 228 to 992Mi blocks.

By moving special case blocks, MUPIP REORG TRUNCATE now reclaims empty space that precedes them in a database file.

The CLOSE command supports a [NO]DESTROY device parameter to control retention of device state for Sequential Disks and SOCKET devices.

V6.0-000 supports MUPIP ROLLBACK -ONLINE as a production grade option, although it will be more useful to future enhancements than it is today.

This release also adds preliminary (field test grade) support for an Instance Freeze mechanism that allows an application to specify a set of conditions under which GT.M suspends database updates. Although this functionality has been extensively tested by FIS, we are designating this specific feature as field test grade since it is new functionality that we may change in future releases in ways that may not be backward compatible.

As usual, the release includes many smaller improvements and corrections. For a comprehensive list, refer to Change History.

## Conventions

Command Syntax: This document uses UNIX syntax (that is, lowercase text and "-" for flags/qualifiers). OpenVMS accepts both lowercase and uppercase text; DCL on OpenVMS uses "/" rather than a "-" to delimit command qualifiers.

Program Names: References to a GT.M program or function use upper case, for example, MUPIP BACKUP. Examples use lower case UNIX, for example, mupip backup -database ACN,HIST /backup

Reference Number: Reference numbers used to track software enhancements and support requests appear in parentheses ().

Platform Identifier: If a new feature or software enhancement does not apply to all platforms, the relevant platform or platforms appear in brackets [].

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

| Pre V5.5-000 terminology | Pre V5.5-000 qualifier | Current terminology | Current qualifiers |
|---|---|---|---|
| originating instance or primary instance | -rootprimary | originating instance or originating primary instance. | -updok (recommended)<br><br>-rootprimary (still accepted) |

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

Effective V6.0-000, GT.M documentation is adopting IEC standard Prefixes for binary multiples. This document therefore uses prefixes Ki, Mi and Ti (e.g., 1MiB for 1,048,576 bytes). All GT.M documentation will over time be updated to this standard.

## 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.

GT.M runs on a wide variety of UNIX/Linux implementations as well as OpenVMS. Consult FIS for currently supported versions. 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 will 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 support team 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 following hardware and operating system versions. Contact FIS for a current list of supported platforms.

| Platform | Supported Versions | Notes |
|---|---|---|
| Hewlett-Packard Integrity IA64 HP-UX | 11V3 (11.31) | - |
| IA64 GNU/Linux - Red Hat Enterprise Linux | | Please contact your FIS account manager if you need GT.M on this platform. |
| Hewlett-Packard PA-RISC HP-UX | | Please contact your FIS account manager if you need GT.M on this platform. |
| Hewlett-Packard Alpha/AXP Tru64 UNIX | | Please contact your FIS account manager if you need GT.M on this platform. |
| Hewlett-Packard Alpha/AXP OpenVMS | 7.3-2 / 8.2 / 8.3 | GT.M supports M mode but not UTF-8 mode on this platform. GT.M does not support several recent enhancements on this platform, including but not limited to database encryption, on-line backup, multi-site replication, PIPE devices and triggers.<br><br>If you need to work with external calls written in C with Version 6.x of the C compiler on Alpha OpenVMS, then you must carefully review all the provided kits for that product and apply them appropriately.<br><br>Although this platform remains at present fully supported with respect to bug fixes, owing to its looming sunset by HP, new functionality is supported on this platform only for FIS' convenience. Future GT.M releases may not be supported on this platform. Regardless of ongoing plans for support of the OpenVMS platform itself, the next GT.M release will likely no longer support OpenVMS 7.x. Please contact your FIS account manager if you need ongoing support for GT.M on this platform. |
| IBM System p AIX | 6.1, 7.1 | Since GT.M processes are 64-bit, FIS expects 64-bit AIX configurations to be preferable.<br><br>While GT.M supports both UTF-8 mode and M mode on this platform, there are problems with the AIX ICU utilities that prevent FIS from testing 4-byte UTF-8 characters as comprehensively on this platform as we do on others.<br><br>Running GT.M on AIX 7.1 requires APAR IZ87564, a fix for the POW() function, to be applied. To verify that this fix has been installed, execute **instfix -ik IZ87564.** |
| GNU/Linux on IBM System z | | Please contact your FIS account manager if you need GT.M on this platform. |
| IBM System z z/OS | | Please contact your FIS account manager if you need GT.M on this platform. |
| Sun SPARC Solaris | 10 (Update 6 and above) | The deprecated DAL calls operate in M mode but not in UTF-8 mode. Please refer to the Integrating External Routines chapter in the Programmer's Guide for appropriate alternatives. |
| x86_64 GNU/Linux | Red Hat Enterprise Linux 6; Ubuntu 8.04 LTS through 12.04; SuSE Linux Enterprise Server 11 | To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware.<br><br>GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.5-24 or later) and ncurses (version 5.5 or later). |

| Platform | Supported Versions | Notes |
|---|---|---|
| | | 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 3.6) to the subsequent prompt **Enter ICU version (ICU version 3.6 or later required. Enter as major-ver.minor-ver):**<br><br>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 (for example on SuSE 11):<br><br>• Find the directory where libncurses.so is installed on your system.<br><br>• 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. |
| 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.<br><br>GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.5-49 or later) and ncurses (version 5.5-24 or later). The minimum CPU must have the instruction set of a 686 (Pentium Pro) or equivalent. |

## 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 is diminished after the two year window.

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

## Migrating to 64-bit platforms

The same application code runs on both 32-bit and 64-bit platforms. Please note that:

- You must compile the application code separately for each platform. Even though the M source code is exactly the same, the generated object modules are different even on the same hardware architecture - 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, except on Tru64 UNIX, where GT.M remains a 32-bit application. |
| 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 (UNIX) or Shareable Executable Images (OpenVMS) after recompiling all M and C source files..

## 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, restart replication, once that replicating instance is current, switch it over to originating instance and upgrade the prior originating instance to become a replicating instance, at least until it's current.

### UNIX

- 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 V6.0-000 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.0-000_arch (for example, /usr/lib/fis-gtm/V6.0-000_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.0-000_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 MUPIP RUNDOWN command of the old GT.M version to ensure all database files are cleanly closed.

- In UNIX editions, 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***.

**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).

#### Additional Information for JFS1 on AIX

If you expect a database file or journal file to exceed 2GB with older versions of the JFS file system, then you must configure its file system to permit files larger than 2GB. Furthermore, should you choose to place journal files on file systems with a 2GB limit, since GT.M journal files can grow to a maximum size of 4GB, you must then set the journal auto switch limit to less than 2 GB.

### OpenVMS

To upgrade from a GT.M version prior to V4.3-001, you must update any customized copy of **GTM$DEFAULTS** to include a definition for **GTM$ZDATE_FORM**.

You can ignore the following section if you choose the standard GT.M configuration or answer yes to the following question:

```
Do you want to define GT.M commands to the system
```

If you define GT.M commands locally with **SET COMMAND GTM$DIST:GTMCOMMANDS.CLD** in **GTMLOGIN.COM** or other command file for each process which uses GT.M, you must execute the same command after installing the new version of

GT.M and before using it. If you define the GT.M commands to the system other than during the installation of GT.M, you must update the system DCLTABLES with the new GTMCOMMANDS.CLD provided with this version of GT.M. See the OpenVMS "Command Definition, Librarian, and Message Utilities Manual" section on "Adding a system command." In both cases, all GT.M processes must match the proper **GTMCOMMANDS.CLD** with the version of GT.M they run..

## Upgrading to GT.M V6.0-000

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

GT.M upgrade procedure for V6.0-000 consists of 5 stages:

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

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V6.0-000 depends on your GT.M upgrade history and your current version.

## Stage 1: Global Directory Upgrade

FIS strongly recommends you back up your Global Directory before upgrading. There is no single-step method for downgrading a Global Directory 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 V6.0-000.
- 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 file-name.
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 V6.0-000.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

## Stage 2: Database Files Upgrade

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

A V6.0-000 database file is a superset of a V5.0-000 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 it. A V6 database supports global variable nodes up to 1 MiB and is not backward compatible. 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 V6.0-000 operations. However, that database can only grow to the maximum size of the version in which it was originally created. If the database is V5.0-000 through V5.3-003, the maximum size is 128Mi blocks. If the database is V5.4-000 through V5.5-000, the maximum size is 224Mi blocks. Only a database created with V6.0-000 (with a V6 MUPIP CREATE) can have a maximum database size of 992Mi blocks.

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 operator log 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 -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; 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 have a maximum size of 1,040,187,392(992Mi) blocks.

## Stage 3: Replication Instance File Upgrade

V6.0-000 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.0-000 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 Supplementary Instance Replication Technical Bulletin. 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 in the Multi-Site Replication technical bulletin 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 V6.0-000, you do not need to extract and reload triggers.

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 V6.0-000. 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.

To extract and reapply the trigger definitions on V6.0-000 using MUPIP TRIGGER:

1. Execute a command like **mupip trigger -select="*" trigger_defs.trg** using the old version. 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. Run **mupip trigger -triggerfile=trigger_defs.trg** using V6.0-000 to reload your trigger definitions.

To extract and reload trigger definitions on a V6.0-000 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 V6.0-000.
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 V6.0-000 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 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.

If 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 V.6-000 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

On selected platforms, 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 other platforms, or 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 Unicode in the utf8 subdirectory, and one compiled without support for Unicode 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.
- 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_V6.0-000_i686, then gtmprofile and gtmcshrc set $gtm_dist to /usr/lib/fis-gtm/gtm_V6.0-000_i686/utf8).
    - On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

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

## Compiling ICU

GT.M versions prior to V5.3-004 require exactly ICU 3.6, however, V5.3-004 (or later) accept ICU 3.6 or later. For sample instructions to download ICU, configure it not to use multi-threading, and compile it for various platforms, refer to Appendix C: Compiling ICU on GT.M supported platforms of the UNIX Administration and Operations Guide.

Although GT.M uses ICU, ICU is not FIS software and FIS does not support ICU. The sample instructions for installing and configuring ICU are merely provided as a convenience to you.

Also, note that download sites, versions of compilers, and milli and micro releases of ICU may have changed since the dates when these instructions were tested rendering them out-of-date. Therefore, these instructions must be considered examples, not a cookbook.

## 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 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 provided by zlib. Simple instructions for compiling zlib on a number of platforms follow. Although GT.M can use zlib, zlib is not FIS software and FIS does not support zlib. These instructions are merely provided as a convenience to you.

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

**Solaris/cc** compiler from Sun Studio:

```
./configure --sharedmake CFLAGS="-KPIC -m64"
```

HP-UX(IA64)/HP C compiler:

```
./configure --sharedmake CFLAGS="+DD64"
```

AIX/XL compiler:

```
./configure --sharedAdd -q64 to the LDFLAGS line of the Makefilemake CFLAGS="-q64"
```

Linux/gcc:

```
./configure --sharedmake CFLAGS="-m64"
```

z/OS:

Refer to the  steps FIS used to install zlib on z/OS in the GT.M for z/OS technical bulletin .

By default, GT.M searches for the libz.so shared library (libz.sl on HPUX PA-RISC) 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 and z/OS) or LD_LIBRARY_PATH (other UNIX platforms) includes the directory containung 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..

# Change History

## V6.0-000

Fixes and enhancements specific to V6.0-000 are:

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-358 | S9607-000255 | DB | UNIX accepts database keys up to 1019 bytes |
| GTM-4053 | C9B12-001861 | Other Utilities | DSE and LKE less likely to hang while waiting for an unavailable shared resource |
| GTM-4525 | S9C07-002167 | DB | Instance Freeze on Error |
| GTM-4820 | C9D01-002237 | Other Utilities | The GT.M OpenVMS distribution kit includes a README.txt file |
| GTM-5698 | C9F05-002722 | DB | GT.M checksums the entire journal record to eliminate the possibility of journal file damage going undetected |
| GTM-6057 | C9H08-002891 | MUPIP | Print timestamp and backlog information as part of the "REPL INFO - Seqno" message in replication logs |
| GTM-6341 | C9J02-003092 | DB | Increase the maximum allowable size of a global variable node to 1 MB |
| GTM-6502 | D9I07-002691 | DB | Improve checking for overlength keys |
| GTM-6571 | S9K04-002763 | DB | Improvements to MUTEXLCKALERT notification mechanism |
| GTM-6686 | C9K09-003328 | MUMPS | Prevent occasional garbling of terminal and operator log messages |
| GTM-6692 | S9K10-002788 | DB | Faster shut down time even when there are very large number of processes |
| GTM-6779 | C9L03-003387 | MUMPS | No more than 511 LOCK + operations on the same lock |
| GTM-6907 | - | MUMPS | Addressed problem with SOCKET READ |
| GTM-7010 | - | MUPIP | Argumentless MUPIP RUNDOWN no longer removes shared memories in use |
| GTM-7071 | - | Other Utilities | Open source makefile place encryption plug-in scripts properly |
| GTM-7075 | - | MUMPS | M-profiling handles larger time values; $HOROLOG is no longer subject to hardware/operating system rounding errors |
| GTM-7167 | C9H08-002891 | MUPIP | Reduced frequency of logging of repeated Source Server connection attempts |

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-7184 | - | Other Utilities | In GDE, -NULL no longer interferes with other qualifiers |
| GTM-7194 | - | MUPIP | Reduce IO impact and improve performance of MUPIP INTEG -FAST |
| GTM-7200 | - | DB | Prevent possible hang of the last process after a kill -9 |
| GTM-7205 | - | MUMPS | Improved handling of LOCKSPACEFULL condition |
| GTM-7208 | - | MUMPS | Issue a LOCKSUB2LONG error when a LOCK subscript resource exceeds 255 bytes |
| GTM-7218 | - | MUPIP | Better handling of problems with the replication log file |
| GTM-7219 | - | Other Utilities | Additional information in operator log messages |
| GTM-7221 | - | MUPIP | Improvements in replication filter handling |
| GTM-7227 | - | MUMPS | Protection against SET $EXTRACT() and SET $PIECE() exceeding the maximum string length |
| GTM-7231 | - | MUPIP | MUPIP JOURNAL -EXTRACT more resilient |
| GTM-7239 | - | DB | Correction for 32-bit platform handling of buffer pool validation |
| GTM-7242 | - | MUPIP | MUPIP REORG -TRUNCATE moves root blocks |
| GTM-7243 | - | MUPIP | Change the "Process That Last Wrote to the Journal File" label to "Process That First Opened the Journal File" in the output of MUPIP JOURNAL -SHOW=HEADER |
| GTM-7253 | - | MUPIP | MUPIP ROLLBACK -ONLINE improvements. |
| GTM-7255 | - | MUMPS | Appropriately handle invalid commands within a FOR body |
| GTM-7257 | - | MUPIP | Prevention of inappropriate STRMSEQMISMTCH errors |
| GTM-7258 | - | MUPIP | New -INITIALIZE qualifier for -UPDATERESYNC and fix issues with -UPDNOTOK and the A->B->P replication configuration |
| GTM-7265 | - | MUPIP | Correct "Start Seqno" in Source Server log |
| GTM-7277 | - | MUMPS | Full Boolean mode corrections |
| GTM-7283 | - | MUMPS | Compile again catches multiple decimal points in a literal |
| GTM-7286 | - | MUPIP | After a process crash, GT.M issues a REQROLLBACK error in cases with replication and BEFORE IMAGE journaling, and REQRECOV error with NOBEFORE journaling; MUPIP SET -JOURNAL cuts the link to the old journal file on system crash |

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-7294 | - | MUPIP | Improve rollback operations in the WAS_ON replication state |
| GTM-7296 | - | MUPIP | MUPIP REPLICATE -{RECEIVER \|SOURCE} -STATSLOG={OFF \| ON} send detailed status to the server log and -LOG no longer accepted |
| GTM-7298 | - | MUPIP | MUPIP REORG concurrency issue |
| GTM-7306 | - | MUMPS | READ from stderr of a PIPE device in UTF-8 works appropriately |
| GTM-7308 | - | DB | Index keys honor max key size |
| GTM-7312 | - | Other Utilities | Additional gtmsecshr permission checks |
| GTM-7317 | - | MUPIP | The Source Server no longer hangs and corrupts its control structure while concurrently doing buffer expansion and adding history records to the instance file |
| GTM-7319 | - | MUMPS | DESTROY deviceparmeter for SOCKET and sequential disk devices |
| GTM-7327 | - | DB | TPNOTACID checking extended |
| GTM-7332 | - | MUMPS | Exponentiation to a fractional power producing a result between (not including) 0-10 reliably recognizes numeric equality |
| GTM-7336 | - | DB | Security enhancements for gtmsecshr |
| GTM-7338 | - | Other Utilities | More accurate reporting of transient control structure problems from DSE CACHE -VERIFY |
| GTM-7339 | - | MUPIP | Duplicate of GTM-7286 |
| GTM-7344 | - | MUPIP | Optimize the generation of REPL_WARN messages from the Source Server and improve the JNLFILOPN message context |
| GTM-7348 | - | Other Utilities | GDE SHOW -COMMAND corrections |
| GTM-7351 | - | MUMPS | WRITE /EOF to a PIPE device flushes output the same way as a CLOSE |
| GTM-7353 | - | DB | TP validation correction |
| GTM-7358 | - | MUMPS | Eliminated occasional segmentation violations, mostly on Linux |
| GTM-7363 | - | MUPIP | Source Server more robust during large catchup |
| GTM-7365 | - | DB | Improved management of abandonned KILLs by MUPIP operations |
| GTM-7371 | - | MUPIP | Fix MUPIP JOURNAL -ROLLBACK to correctly initialize Stream Sequence Number in the instance file |

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-7378 | - | MUPIP | Receiver Server no longer does redundant ONLINE ROLLBACKs |
| GTM-7383 | - | MUPIP | With NOBEFORE_IMAGE journaling, eliminate unnecessary flushing and fsync during an EPOCH. |
| GTM-7388 | - | MUMPS | Correctly handle TP transactions where trigger definitions and/or trigger deletions are the only updates |
| GTM-7389 | - | MUMPS | New process statistics for ZSHOW "G" |
| GTM-7390 | - | MUPIP | MUPIP ENDIANCVT correction |
| GTM-7393 | - | MUPIP | Improve the efficiency of recovery and rollback operations in events like a system crash |
| GTM-7396 | - | MUPIP | MUPIP SET -PARTIAL_RECOV_BYPASS resets file_corrupt flag in database file header |
| GTM-7397 | - | MUMPS | The ]] operator sorts canonical numerals ahead of strings in alternate local collations |
| GTM-7398 | - | MUPIP | Correct rollback replay of NULL journal records |
| GTM-7400 | - | MUMPS | ZHELP works on UNIX without extra steps |
| GTM-7401 | - | MUPIP | No IPC left around in case of errors during Source Server startup |
| GTM-7402 | - | MUPIP | Correctly maintain KILLs in Progress and Abandoned KILLs counters when a KILL command encounters an error |
| GTM-7403 | - | Other Utilities | The first line of the GT.CM scripts invokes the Bourne shell |
| GTM-7405 | - | Other Utilities | gtminstall.sh creates a new configure.sh each time it runs |
| GTM-7410 | - | MUMPS | Call-in interface fix |
| GTM-7411 | - | MUMPS | UNIX editions support some special purpose devices |
| GTM-7413 | - | MUMPS | Improve the handling of a failed OPEN of a new SOCKET device |
| GTM-7426 | - | DB | Trigger concurrency fix |
| GTM-7431 | - | DB | UNIX editions report DBFSYNCERR to the operator log |
| GTM-7433 | - | MUMPS | $ZPREVIOUS() fix |
| GTM-7439 | - | DB | Prevent possible problem switching journal files, typically at mass process exit |
| GTM-7444 | - | MUPIP | Truncate concurrency fix |

**Change History**

| Id | Prior Id | Category | Summary |
|---|---|---|---|
| GTM-7452 | - | MUMPS | Fix a memory segmentation violation (SIG-11) that occurs when $ZTRIGGER() processes trigger definitions stored as constants in a routine within a shared library |

## M-Database Access

- GT.M allows keys up to 1019 bytes in length. To accommodate the fact that a key must still fit within one database block, the maximum key size is also limited to 40 bytes less than the block size, e.g., a 1024 byte block can support a maximum key size of 984 bytes and a 1536 byte block size is the smallest that supports a key with the maximum size of 1019 bytes. Both GDE and MUPIP also enforce these limits. Previously keys were limited to 255 bytes. Since the smallest database block size is 512 bytes, neither GDE nor MUPIP previously needed to validate the maximum key size against the block size. [UNIX] (GTM-358) ✓

- GT.M now has a field test grade implementation of an Instance Freeze facility. This provides users with a mechanism to freeze regions associated with an instance on a set of conditions they control. It provides an alternative to the GT.M design of keeping the system up in the face of operational challenges, even when they represent rising levels of risk. That original design continues to be what FIS considers wholesome practice for maximum business continuity and recommends that you use it in conjunction with logical multi-site configurations layered on replication. Instance freeze is provided for those users who prefer for operational reasons to freeze an operating instance instead of switching over.

  GT.M recognizes the gtm_custom_errors environment variable as a flag that enables a facility to freeze updates to an instance on the occurrence of a configurable set of errors. We refer to this mechanism as "Instance Freeze" to differentiate it from the region freeze associated with MUPIP and DSE operator actions. $gtm_custom_errors must resolve to a file, including its path, which contains a text list of error mnemonics from the set documented in the GT.M Message and Recovery Guide. The mnemonics must be capitalized, one per line. To be affected by this mechanism, a region must be a part of the instance and the error must have a database region context. To activate a freeze, a region must be configured to set freezes. In other words, a region can be affected by freezes, but also be excluded from setting freezes. An operator can set or clear Instance Freeze with a MUPIP command (see below). When Instance Freeze is active, GT.M modifies the NOSPACEEXT message from error (-E-) to warning (-W-) to indicate it is performing the extension even though the available space is less than the specified extension amount. When GT.M detects a hard out-of-space condition in any database, journal or replication instance file storage media, it sends a DSKNOSPCAVAIL message to the operator log. DSKNOSPCAVAIL is a condition which GT.M retries indefinitely and, if and when the condition clears due to operator intervention, GT.M automatically lifts the freeze (sending a DSKSPCAVAILABLE). When an instance is frozen, attempts to flush buffers to the database and journal files hang, but flushes to ancillary files like the backup temporary file, journal extract file, etc. may continue.

  If the freeze scheme is turned ON ($gtm_custom_errors is defined) and $gtm_repl_instance is defined, the following MUPIP operations and any GT.M process that opens a database (even for the purpose of reading a global) attempt to open the replication journal pool. Any error while opening the journal pool, except for the case when it is unavailable (due to the instance being cleanly shutdown), results in a runtime error.

  - MUPIP INTEG -REG [-NOONLINE]
  - MUPIP BACKUP
  - MUPIP EXTEND
  - MUPIP FREEZE
  - MUPIP REORG [-UPGRADE][-DOWNGRADE]
  - MUPIP EXTRACT
  - MUPIP RUNDOWN -REG

- MUPIP JOURNAL -EXTRACT
- MUPIP JOURNAL -ROLLBACK [-ONLINE]
- MUPIP SET -FILE
- MUPIP SET -FILE
  - -EXTENSION_COUNT
  - -VERSION
  - -INST_FREEZE
  - -QDBRUNDOWN

The following lists the syntax for GDE and MUPIP commands that manage the facility:

```
 GDE  === $char(10) add -region B -dynamic_segment=B -noinst[_freeze_on_error] $char(10) add -region C -
dynamic_segment=C -inst[_freeze_on_error] $char(10) template -region -inst[_freeze_on_error] $char(10) template
 -region -noinst[_freeze_on_error] $char(10) change -region A -inst[_freeze_on_error] $char(10) change -region C
 -noinst[_freeze_on_error]$char(10) show -region # includes inst freeze on error flag $char(10) show -template
 # includes inst freeze on error flag $char(10) show -command # includes inst freeze on error flag $char(10)
 $char(10) MUPIP $char(10) ===== $char(10) mupip set -region B -inst[_freeze_on_error] $char(10) mupip set -
region D -noinst[_freeze_on_error] $char(10) mupip replicate -source -freeze # display freeze state $char(10)
 mupip replicate -source -freeze=on -[no]comment[='"for example comment"'] # freeze instance with optional
 comment $char(10) mupip replicate -source -freeze=off # unfreeze instance $char(10) mupip replicate -source -
checkhealth # checks freeze state, displays if frozen
```

Note that the argument to the -comment qualifier must be enclosed in both single and double quotes and if you don't wish to supply a comment, you must explicitly specify -nocomment.

There are certain error messages which can occur before attaching to, or after detaching from, a journal pool - in such cases, those error message do not have access to the journal pool and cannot activate a freeze. For example, GT.M detects SCNDDBNOUPD after detaching from the journal pool and hence can't cause an Instance Freeze. Because processes typically can't update instance database files without access to the journal pool, this should not be a material restriction.

If an I/O error occurs while flushing global buffers, GT.M reports a DBIOERR error to the operator log; if such an error occurs while writing an EPOCH, GT.M raises the DBIOERR as a run-time error. Previously, GT.M did not report such errors and repeatedly invoked cache-recovery logic.

GT.M distribution kits include a custom_errors_sample.txt file which can be used as a target for the gtm_ custom_errors environment variable. The configure script copies this file into the installation directory. [UNIX] (GTM-4525) ✓

- The checksum used by GT.M to validate journal records includes the entire record. Previously, GT.M computed the checksum only on parts of the journal record deemed sufficient to detect damaged journal records. A system failure that resulted in damage only to the unprotected part of a journal record could previously result in database damage, or memory segmentation violations (SIG-11 on UNIX; ACCVIO in OpenVMS) if that journal record was used in recovery or rollback. We are not aware of such a failure ever occurring at an actual user site. Recent testing by FIS alerted us to possibility of failure scenarios that could damage the unprotected parts but not also damage the protected parts, while in system failure scenarios previously analyzed, the prior approach provided sufficient protection. (GTM-5698)

- GT.M handles global variable nodes up to 1MiB - the same maximum as that of a local variable node - by spanning large global variable nodes across multiple database blocks. The meaning of the maximum record size for a database region henceforth refers only to the size of the value. In the event a global variable node spans multiple blocks, and the process is not already within a TP transaction, the GT.M run-time system automatically and transparently performs the entire operation within an implicit TP transaction (just as is the case with triggers). GDE and MUPIP restrict the maximum record size to 1,048,576 bytes without regard to the database block size. Previously the maximum record size was limited by the block size and referred to the maximum size of the record within a database block including the key and delimiters.

The following journaling-related limits and defaults have been updated to support the increased record and key sizes:

| Journaling limits and defaults | Pre V6.0-000 | V6.0-000 |
|---|---|---|
| Minimum Allocation (blocks) | 200 | 2,048 |
| Minimum Align Size (blocks) | 256 | 4,096 |
| Default Align Size (blocks) | 2,048 | 4,096 |
| Minimum Autoswitch Limit (blocks) | 4,096 | 16,384 |
| Minimum Journal Buffer Size (blocks) | No hard lower limit | 2,307 |
| Default Journal Buffer Size (blocks) | No hard default | 2,308 |

In conjunction with these changes, the maximum database size has been increased to 1,040,187,392(992Mi) blocks. Previously this limit was 234,881,024(224Mi) blocks.

MUPIP INTEG and DSE deal appropriately with global variable nodes that span database blocks. MUPIP INTEG validates record size using the revised definition and limit, and also validates spanning node format and ordering. The INTEG report contains the total number of spanning nodes and the number of blocks holding spanning node fragments. DSE DUMP BLOCK -FORMAT=ZWR output for a block containing a spanning node has the syntax of a SET $ZEXTRACT() argument. MUPIP LOAD -FORMAT=ZWR accepts input including $ZEXTRACT() syntax. Together these utilities enable a block level salvage strategy as described in the Maintaining Database Integrity chapter of the Administration and Operations Guide.

The binary extract header format supports maximum record lengths up to 7 decimal digits. Formerly, maximum record length could not exceed 64KiB. Since the binary extract is at the record level and new subscripts have been added, binary extracts cannot be used to transfer data to previous GT.M versions that do not support spanning nodes. If the key lengths and value length allow all global variable nodes to fit in one database block of previous GT.M versions, use ZWR format for MUPIP EXTRACT and LOAD.

Operationally, this uncoupling of the maximum size of a global variable node from the database block size means that a minuscule percentage of large global variable nodes - perhaps just one large global variable node - need no longer force your application to use a larger block size than the optimum one for application performance. Although there is no performance impact from this enhancement for global variable nodes that do not span blocks, nodes that span blocks do incur some overhead, and optimum application performance is likely to be obtained from a block size that accommodates the majority of nodes within a single block. When and if you tune for smaller block sizes to take advantage of this new feature to enhance performance, you should also adjust the number of global buffers. Start with the number of global buffers that gives the same product of block size and number of global buffers, and adjust from there. A further consideration is that since TP can be involved in an update and a transaction is limited to half the number of global buffers, the number of global buffers should be at least twenty-six plus twice the number of the blocks required by the largest global variable node in your application.

Spanning nodes are not supported via GT.CM. Maximum record size checking was enhanced to only use the size of the value (and create a spanning node if needed). Since spanning nodes are not supported on GT.CM, maximum record size is calculated as done previously (meaning that there must be space for the record header, key, etc). If a GT.CM GNP server finds a database contains spanning nodes and a particular non-spanning node contains the value $c(0), it will issue an error even though the node is not a spanning node. [UNIX]

In addition to the above UNIX-only features, as a part of this project GT.M has also introduced the following cross-platform changes: MUPIP no longer issues JNLBUFFTOOSM and JNLBUFFTOOLG errors when the journal buffer size specified by the user or previously stored in the database file header falls outside the permissible range. Instead, the journal buffer size is automatically adjusted, up or down, to take a legitimate value. Note that the auto-adjustment occurs on top of the pre-existing rounding-up of the journal buffer size to the next multiple of [block size / 512] ratio. MUPIP issues a notification about any adjustment of the journal buffer size via the newly introduced JNLBUFFREGUPD or JNLBUFFDBUPD warning message, for region and database file accordingly. If the value prior to adjustment was invalid, MUPIP reflects that fact in an appendix message following one of the above warnings. A MUMPS process may also increase the journal buffer size value in the database file header when it initializes shared memory, to prevent insufficient resource allocation if, for example, using a database created with older GT.M executables. (GTM-6341) ✓

- GT.M appropriately handles the case where an unsubscripted global variable name exceeds the maximum key size of the database region, a case that requires a maximum key size of less than 32, by issuing a KEY2BIG error. As part of this change GT.M is likely to display more of the over-length key when reporting a GVSUBOFLOW condition. Previously a name that exceeded the maximum key size could cause an incorrect GTMASSERT error and/or memory corruption, which in turn caused a memory segmentation fault (UNIX SIG-11 or OpenVMS ACCVIO); because this required both an unusually small maximum key size and an unusually long global variable name, we expect this was never encountered outside our test scenarios. (GTM-6502)

- The critical section management that produces the MUTEXLCKALERT warnings in the operator log includes the gtm_procstuckexec mechanism for automating the gathering diagnostic information or other operator actions. The MUTEXLCKALERT mechanism is much more careful to only issue alerts if a single process is holding a resource without ever releasing it and does so at intervals in the range of approximately 30-60 seconds. Previously it only took into account how long a single process had waited regardless of whether other work was getting completed and it did not include the gtm_procstuckexec capability. [UNIX](GTM-6571) ✓

- To facilitate usage (for example, benchmarking scenarios) where it is desirable for a large number of processes to shut down almost concurrently, GT.M database files provide a [NO]QDBRUNDOWN setting (the default of NOQDBRUNDOWN preserves current behavior for normal usage). If QDBRUNDOWN is set for a database region, when a terminating process observes that a large number of processes are attached to the database, it bypasses logic to check whether it is the last process attached to that database (so that it can clean up the database fileheader and IPC resources associated with that database file when it is the last process) . Since the check to determine whether a process is the last process must be performed in a critical section, eliminating the check when there are a large number of processes attached to the database allows rapid process shutdown, as well as removing an impediment to process startup. Note that with QDBRUNDOWN there exists the possibility of race conditions that leave the database fileheader and IPC resources in need of cleanup. Although the logic minimizes the probability of such a race condition, it cannot eliminate it. When using QDBRUNDOWN, FIS recommends an explicit MUPIP RUNDOWN of the database file after the last process exits, to ensure cleanup of the database fileheader and IPC resources.

  This setting in the database fileheader is changed by the -[NO]QDBRUNDOWN switch of the MUPIP SET and DSE CHANGE -FILEHEADER commands. DSE DUMP -FILEHEADER -ALL shows the number of processes that have performed a quick exit since the database was last opened). As with other database parameters, the initial value of the setting is taken from the global directory when MUPIP CREATE initially creates a database file. Use the -[NOQDBRUNDOWN] switch of GDE's CHANGE -REGION and TEMPLACE -REGION commands to manage this setting in global directory files. GDE SHOW shows the setting of the switch, labeled QdbRndwn.

  In conjunction with this change, GDE SHOW changes the labeling of certain fields, in order to improve the readability of the output:

- "Standard NullColl" field is now "Std Null Coll"
- "Journaling" field is now "Jnl"
- "Qdb Rndwn" is newly introduced

[UNIX] (GTM-6692) ✓

- If the last owner of a critical section terminated without releasing ownership, the last remaining active process can successfully terminate without first doing a database access. Previously under the rare occasions when those conditions existed, typically because of a kill -9 that last process exiting could hang. Note that FIS strongly recommends against terminating processes with kill -9. [UNIX] (GTM-7200)

- On 32-bit platforms, GT.M appropriately issues warning messages during buffer pool validation. In most recent releases, these warnings could cause segmentation violations (SIG-11). The most common reason for GT.M processes to initiate buffer pool validation is to clean up after processes are terminated without the opportunity to exit cleanly (for example, terminated with a kill -9); other causes are rare internal GT.M errors. Note that FIS strongly recommends against terminating processes with kill -9. [Linux X86, HP-UX PA-RISC, TRU64] (GTM-7239)

- GT.M ensures keys inserted into index blocks honor the maximum key size field in the fileheader. Previously, due to changes introduced in V54002, database updates could create index keys exceeding the maximum key size, resulting in INTEG errors. We are not aware of this having happened outside FIS test environments. (GTM-7308)

- GT.M provides protection from certain coding practices that can cause deadlocks (in the worst case) and long hangs (in the best case) by enhancing and extending to HANG, JOB, LOCK, OPEN, READ, and ZALLOCATE, protection previously provided for ZSYSTEM and BREAK when used within TP transactions. When $TRETRY>2 and the GT.M process holds the critical sections for all participating database regions, if the process executes one of the aforementioned commands with no timeout, or with a timeout exceeding the value of the gtm_tpnotacidtime UNIX environment variable or OpenVMS logical name GTM_TPNOTACIDTIME, GT.M sends a TPNOTACID informational message to the system log and releases the critical sections before continuing. GT.M limits gtm_tpnotacidtime to the range zero (0) through 30 seconds, defaulting it to two (2) seconds if it is not specified or its value is outside the range. In the cases of LOCK and ZALLOC with timeouts that do not exceed the gtm_tpnotacidtime, if the resource name is locked on the final retry, the process may generate TPNOACID messages while it tries to ensure there is no possibility of a deadlock. The text in the TPNOTACID message for the Direct Mode case is now: "Entering DIRECT MODE - TP RESTART will fail". This change does not affect application logic; for most cases, it does limit the time which a long running command can hold control of a database resource. A process with a non-Isolated or long running command may encounter potentially indefinite restarts. Such processes constitute a non-productive and disruptive load. If your log file contains TPNOTACID messages, review the logic to see whether the non-Isolated commands can be moved outside of transaction encapsulation, or use timeouts that ensure they are minimally disruptive. As a consequence of this change, a process running code which gives TPNOTACID messages might fail to get an error indicating database damage should it encounter such an unlikely eventuality - since the process is in an indefinite restart; such a case cannot cause any additional damage. Another consequence of this change is that MUPIP TRIGGER operations have an implicit timeout of zero (0), meaning the read must succeed on the first try or the command will act as if it received no input. Consider using $ZMAXTPTIME to prevent transactions from running unreasonably long times. The new gtm_zmaxtptime UNIX environment variable or OpenVMS logical name GTM_ZMAXTPTIME can specify an initial value for the $ZMAXTPTIME Intrinsic Special Variable, which controls whether and when GT.M issues a TPTIMEOUT error for a TP transaction that runs too long. GT.M limits this initial value to from one (1) to 60 seconds and ignores values outside of the recognized range; this range check does not apply to SET $ZMAXTPTIME. Prior versions limited only ZSYSTEM and Direct Mode and did not provide the initialization capabilities. The workaround was to use short timeouts for timed operations in order to prevent them from stalling database activity. (GTM-7327) ✓

- As part of an ongoing process to tighten GT.M security, V6.0-000 brings changes to the operation of gtmsecshr as described below.

  1. gtmsecshr messages go to syslog rather than a gtmsecshr specific log. These messages can be identified with the GTMSECSHR facility name as part of the message. Previously, all messages were from the GTM facility.

  2. $gtm_dist/gtmsecshr can be invoked directly from the command line ($gtm_dist must be set in the environment and must match the directory from which gtmsecshr is executed). Invocation from the command line previously generated an error.

  Please note that these are part of our continuing improvements to GT.M - these changes neither alter the GT.M security philosophy and model nor are they intended to block any particular exploit. [UNIX] (GTM-7336) ✓

- GT.M correctly validates TP transactions involving splits of index blocks. Previously, a TP transaction running concurrently with another GT.M process, each doing a specific and unlikely sequence of database updates, could result in data loss. While we we able to create this scenario during our internal testing, we are not aware of it ever having been encountered during actual use. In addition, GT.M transactions in the final retry more robustly invoke cache recovery upon detecting a concurrent transaction that has encountered an error in the midst of commit. Previously, these updates issued an inappropriate error. (GTM-7353)

- MUPIP FREEZE on a database with abandoned kills proceeds after producing a warning message. Previously in this situation, the FREEZE operation gave an error and did not complete. (GTM-7365)

- GT.M handles a specific trigger-related concurrency issue correctly. Previously, there was a small possibility of a SIG-11 or access violation during a trigger invocation if trigger definitions changed concurrently. As far as we know this issue has only been encountered in our test environment. [UNIX] (GTM-7426)

- GT.M reports all DBFSYNCERR occurrences to the system log. Previously, journal flushes did not report DBFSYNCERR to the system log. Such errors should be rare, however, please contact FIS if this becomes an issue in your environment.[UNIX] (GTM-7431)

- GT.M correctly switches journal files when the remaining space in the file seems insufficient to accommodate a journal EOF record. In previous versions, GT.M did not ensure that there was adequate space at the end of a journal file to close it cleanly (for example, to write an EOF record). This resulted in abnormal process termination (and possible core files) for all processes accessing the database, as each in turn attempted to close the journal file, leaving an improperly terminated journal file. Scenarios in which this could occur were very rare and more likely to be encountered in testing / benchmark scenarios than in normal usage - although the issue was long-standing, FIS is not aware of it ever having been encountered in other than testing / benchmarking scenarios. (GTM-7439)

## M-Other Than Database Access

- Messages that GT.M sends to the terminal or operator log are printed in their entirety. Previously, when dealing with timer-driven events, GT.M sometimes wrote a second message on top of a message under construction, which resulted in rare instances of garbled messages. [UNIX] (GTM-6686)
- GT.M gives a LOCKINCR2HIGH error when an Incremental LOCK (LOCK +) attempts to raise the level above 511. Previously GT.M had the same limit but did not enforce it, causing an asymmetrically small number of decrements (LOCK -) to release the LOCK; ZSHOW "L" showed the wrapped value. (GTM-6779) ✓
- GT.M processes appropriately handle a MUPIP INTRPT arriving in a certain specific window during a SOCKET READ. Previously processes could fail when a SOCKET READ was interrupted. Even with knowledge of the fix, FIS was never able to reproduce the failure in our development environment and it was only ever reproduced at one site under a specific workload involving intensive SOCKET READs on a heavily loaded system. While the only observed failure was a segmentation violation (signal-11 on UNIX; ACCVIO on OpenVMS), it was theoretically possible for the failure to happen even on a lightly loaded system performing infrequent SOCKET READs, and it was also theoretically possible for the READ to return incorrect data without failing. (GTM-6907)
- M-profiling appropriately handles large values for the user, CPU, and elapsed time values collected for individual lines, labels, and master and child processes. Previously, these values could overflow on long-running processes, yielding incorrect values in the tracing report. In addition, in UNIX editions, $HOROLOG is not subject to hardware or operating system rounding errors. Previously, on certain (at least AIX) platforms $HOROLOG could occasionally report the same value when, in fact, at least one second had elapsed between the measurements. (GTM-7075)
- A LOCK operation that finds no room in LOCK_SPACE to queue a waiting LOCK, does a slow poll waiting for space. Previously a LOCK encountering such a situation generated a LOCKSPACEFULL error (GTM-7205) ✓
- M LOCKS issue a LOCKSUB2LONG error when a subscript in the resource name exceeds 255 bytes. Previously, a LOCK with such a resource name allocated the memory but LKE SHOW and ZSHOW "L" could not report it. (GTM-7208)
- GT.M does more appropriate checking for maximum length output value for SET $PIECE() and SET $EXTRACT(). Previously, a large piece value or byte position respectively could cause MEMORY (UNIX) or VMSMEMORY (VMS) errors, return corrupt or otherwise incorrect data, or result in a memory segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). (GTM-7227)
- GT.M handles an invalid command within a FOR body appropriately. In V5.5-000, the change to permit the run-time to skip over an INVCMD error with a FALSE postconditional created a situation where an invalid command within a FOR with a termination condition generated a memory segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO) at compile time. (GTM-7255)
- In "full Boolean" mode, GT.M handles all known cases where parentheses surround a term other than the last term with multiple components. In V5.5-000, depending on the order and sense of the terms GT.M could evaluate such constructs incorrectly.(GTM-7277)
- The GT.M compiler reports an error if it encounters a numeric literal that contains multiple decimal points. This corrects an unintended change associated with the numeric normalizations (C9H02-002826) in V5.4-002. (GTM-7283)
- READ from stderr of a PIPE device in UTF-8 works appropriately. Previously, such a READ could hang. [UNIX] (GTM-7306)
- The [NO]DESTROY deviceparameter for the CLOSE command for sequential files, FIFO and SOCKET devices determines whether the process retains device characteristics after the CLOSE. Preserving prior GT.M behavior, the default is DESTROY

for sequential disk files and FIFO devices, and NODESTROY for SOCKET devices. While NODESTROY allows the re-OPEN of previously CLOSE'd device with the same characteristics as when it was last CLOSE'd, as described by the M standard, every tracked device uses process memory. A device that has been DESTROYed on CLOSE cannot be re-opened with the previous characteristics, but reclaims memory used by the process for that device. This device parameter is ignored for other devices - terminals and VMS MBX devices act as if the specification were NODESTROY and PIPE devices act as if the specification were DESTROY. Also, the device parameter is ignored for CLOSE of a specific socket rather than the entire socket device. Previously, CLOSE removed all information for a sequential file and FIFO, and kept the information for a SOCKET. (GTM-7319) ✓

- Numeric equality comparisons of certain fractional exponents correctly evaluate to 1. Previously, as a consequence of incorrect internal flags, although such results printed correctly, and produced accurate results when used in subsequent calculations, a comparison for equality could incorrectly evaluate to 0. The result values subject to this incorrect behavior were a subset of the range -999999.999 through 999999.999 (i.e., with up to three digits after the decimal point). For example, 4**.5=2 previously incorrectly evaluated to 0. (GTM-7332)

- Prior to closing the underlying UNIX pipe file descriptor, a WRITE /EOF to a PIPE device now flushes output in the same way as a CLOSE and sets $X to zero (0). If you do not want WRITE /EOF to flush any pending output including padding in FIXED mode or a terminating EOL in NOFIXED mode, SET $X=0 prior to the WRITE /EOF. A READ from a PIPE device in FIXED mode operates correctly after a WRITE /EOF. Previously, WRITE /EOF on a PIPE closed the output file descriptor without flushing any pending output and a READ after a WRITE /EOF in FIXED mode caused a "SYSTEM-E-ENO9, Bad file descriptor" error. [UNIX](GTM-7351) ✓

- GT.M appropriately handles some situations where it could, under some circumstances, stimulate underlying system-related software to behave in an unintended fashion usually (always as far as we know) resulting in a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). This has only been reported on certain combinations of LINUX releases and x86 or x86-64 hardware, but could, at least in theory, occur on other platforms. (GTM-7358)

- GT.M correctly handles TP transactions where the only updates to the database are definitions and/or deletions of triggers. Previously, in some rare cases, such transactions could result in a segmentation violation (SIG-11). [UNIX] (GTM-7388)

- ZSHOW "G" reports the following [additional] process statistics as part of a comma separated list where each statistic has its mnemonic followed by a colon and the count. DSE DUMP -FILEHEADER reports the same statistics for a region. The characters used to arrive at each mnemonic are highlighted in upper-case in the corresponding description.

```
DFL : # of Database FLushes$char(10)DFS : # of Database FSyncs$char(10)JFL : # of Journal FLushes$char(10)JFS :
 # of Journal FSyncs$char(10)JBB : # of Journal Buffer Bytes written$char(10)JFB : # of Journal File Bytes
 written$char(10)JFW : # of Journal File Writes$char(10)JRL : # of Journal Logical Records$char(10)JRP : #
 of Journal Records for Pblks$char(10)JRE : # of Regular Epoch Journal Records $char(10)JRI : # of Idle epoch
 Journal Records$char(10)JRO : # of Journal Records of Other types$char(10)JEX : # of Journal file EXtensions
$char(10)DEX : # of Database file EXtensions
```

The above list is in the order of presentation. The below list describes the mnemonics in more detail.

- DFL - counts the times a process flushes the entire set of dirty database global buffers in shared memory to disk.

- DFS - counts the times a process does an fsync of the database file. For example: a) after writing an epoch journal record, b) as part of database file extension c) during database rundown d) as part of mupip reorg -truncate etc.

- JFL - counts the times a process flushes all dirty journal buffers in shared memory to disk. For example: when switching journal files etc.

- JFS - counts the times a process does an fsync of the journal file. For example: when writing an epoch record, switching a journal file etc.

- JBB - counts the number of bytes written to the journal buffer in shared memory.
- JFB - counts the number of bytes written to the journal file on disk. For performance reasons, GT.M always aligns the beginning of these writes to file system block size boundaries. On Unix, JFB counts all bytes including those needed for alignment in order to reflect the actual IO load on the journal file. Since the bytes required to achieve alignment may have already been counted as part of the previous JFB, processes may write the same bytes more than once, causing the JFB counter to typically be higher than JBB.
- JFW - counts the times a process invokes a write system call for a journal file.
- JRL - counts the number of of logical journal records (e.g. SET, KILL etc.)
- JRP - counts the number of of PBLK and AIMG journal records written to the journal file (these records are seen only in a -detail journal extract)
- JRE - counts the number of of regular EPOCH journal records written to the journal file (only seen in a -detail journal extract); these are written every time an epoch-interval boundary is crossed while processing updates
- JRI - counts the number of of idle EPOCH journal records written to the journal file (only seen in a -detail journal extract); these are written when a burst of updates is followed by an idle period, around 5 seconds of no updates after the database flush timer has flushed all dirty global buffers to the database file on disk
- JRO - counts the number of of all journal records other than logical, PBLK, AIMG and EPOCH records written to the journal file (e.g. PINI, PFIN, etc.)
- JEX - counts the number of of times a process extends the journal file
- DEX - counts the number of of times a process extends the database file

(GTM-7389)

- The ]] ('sorts-after') operator sorts canonical numerics ahead of strings when using an alternate local collation. Previously, the ]] operator gave no precedence to canonical numerics over strings in alternate local collations. For information on how to revert to the prior behavior please contact FIS. (GTM-7397)
- The configure installation script retains the GTMHELP.o object file needed to execute the MUMPS zhelp online documentation command. Previously the installation deleted this file which prevented ZHELP from working as intended. The workaround was to compile GTMHELP.m into the gtm_dist direcory. [UNIX] (GTM-7400)
- The GT.M call-in interface properly passes negative integers and longs between -1 and -999999 to GT.M. In 5.5-000, it passed these numbers as positive. (GTM-7410)
- The OPEN command appropriately opens /dev/zero, /dev/random and /dev/urandom devices. Previously, an OPEN command for the /dev/zero, /dev/random or /dev/urandom devices with FIXED device-parameter could result in a memory segmentation violation, while without that deviceparameter, it gave an UNIMPLOP error. (SIG-11). [UNIX](GTM-7411)
- A failing OPEN of a new SOCKET device releases the memory allocated to set up the device and a subsequent ZSHOW "D" does not show that device. Previously, a failed OPEN of a new SOCKET device leaked some memory and the device subsequently showed up as CLOSED device in ZSHOW "D". (GTM-7413)
- $ORDER() and $ZPREVIOUS() return an empty string when they reach the trigger definitions (stored in ^#t) as it is not a normally accessible global. Since the introduction of triggers in V5.4-000 if there were trigger definitions, these functions could return ^#t. [UNIX] (GTM-7433)

- $ZTRIGGER() appropriately processes trigger definitions stored as constants in routine within a shared library. In addition, MUPIP TRIGGER and $ZTRIGGER() accurately report trigger locations in error messages. Previously $ZTRIGGER() gave a segmentation violation (SIG-11) when trying to define a trigger from a string value embedded in a shared library and, along with MUPIP TRIGGER, could slightly misstate trigger locations in error reports. The workaround for the $ZTRIGGER() issue was to use a flat file to hold the trigger definition. [UNIX] (GTM-7452)

## Utilities-MUPIP

- The receiver server log prints the timestamp as well as the current backlog (on the receiver side) as part of the "REPL INFO - Seqno" messages. Previously, it did not print this information. A sample new message follows.

  ```
  Wed Jul 11 13:39:15 2012 : REPL INFO - Seqno : 23001 [0x59d9] Jnl Total : 2208264 1 Msg Total : 2392384 1 Current
   backlog : 9263 [0x242f]
  ```

  The source server prints hexadecimal values as well as the timestamp as part of the "REPL INFO - Seqno" messages. Previously, it printed only decimal values. A sample new message follows.

  ```
  Wed Jul 11 13:39:15 2012 : REPL INFO - Seqno : 23482 [0x5bba] Jnl Total : 2254272 [0x2265c0] Msg Total : 2442128
   [0x254390] CmpMsg Total : 2442128 [0x254390] Current backlog : 0 [0x0]
  ```

  (GTM-6057)

- Argumentless MUPIP RUNDOWN leaves shared memory associated with databases or replication servers if there are processes attached to the respective shared memories. Previously, argumentless MUPIP RUNDOWN in some cases removed shared memories associated with database or replication servers thereby causing errors during replication server shutdown commands. [UNIX]

  If Argumentless MUPIP RUNDOWN does not remove a database or replication shared memory segment (due to errors with database or replication instance files), it reports MUFILRNDWNFL2 and MUJPOOLRNDWNFL or MURPOOLRNDWNFL (depending on whether the segment is a Journal Pool or Receive Pool) messages along with appropriate details in the terminal and operator log. Previously, argumentless MUPIP RUNDOWN did not issue such messages. [UNIX] (GTM-7010) ✓

- The Source Server logs its unsuccessful connection attempts starting frequently and slowing to approximately every five minutes. This change does not affect the rate of connection attempts, only the frequency of logging unsuccessful attempts. Previously the Source Server logged every connection attempt. (GTM-7167)

- As MUPIP INTEG with the FAST option does not check data (level 0) blocks, an optimization results in application processes never updating the snapshot file for modifications to level 0 blocks. This reduces IO impact and improves performance. Previously, application processes wrote before images of data blocks as well during a MUPIP INTEG FAST. Note that this speeds up MUPIP INTEG -FAST -FULL, which is a quick way to get an approximate size of each global variable in the database.[UNIX] (GTM-7194)

- In the event that the replication log file specified on a MUPIP REPLICATE command line is not writable (for example, the directory does not exist or the process lacks permission to create or write to the file), the MUPIP process immediately terminates with an error message, returns a failure status to the shell and does not report an issue to the system log. Previously during the start-up of servers, the MUPIP process reported the problem in the system log; or during the running of servers, MUPIP ignored the problem so it appeared that the command executed successfully without returning failure status to the shell, but the server running background subsequently terminated with a segmentation fault (SIG-11 on UNIX; ACCVIO on OpenVMS). (GTM-7218)

- GT.M manages external filters in a more robust manner by correcting, if necessary, journal sequence and stream sequence numbers for transactions returned by the filter; it also removes any null records generated by the filter inside a wrapped transaction. Previously a filter that damaged these values or produced inappropriate null records could cause replication to fail or to enter an indefinite loop. [UNIX] (GTM-7221)

- MUPIP JOURNAL -EXTRACT starts appropriately independent of the state of the corresponding database file and does not write new records to journal file. Previously, if the database had a buffer pool in a suspect state, typically from an incomplete or improper shutdown, a JOURNAL -EXTRACT attempt involving the region terminated with a segmentation fault (UNIX SIG-11 or OpenVMS ACCVIO) or inappropriately wrote to the journal file. (GTM-7231)

- By default MUPIP REORG block reordering includes all global data and index blocks except the master root block for the region. This permits the -TRUNCATE option to more consistently free space; it also induces reorganization of blocks occupied by trigger definitions for a given region. Previously REORG did not move Global Directory Tree blocks, the root blocks of individual globals, or blocks occupied by trigger definitions for a given region; this meant that globals created for the first time when the database was in a reasonably full state could substantially reduce the effectiveness of -TRUNCATE. [UNIX] (GTM-7242)

- MUPIP JOURNAL -SHOW=HEADER reports details of the process that first opened the journal file under the section labeled "Process That First Opened the Journal File". Previously it incorrectly labeled the section as "Process That Last Wrote to the Journal File". (GTM-7243) ✓

- MUPIP ROLLBACK -ONLINE is now considered production grade functionality. Since it can take a database backwards in state space, please make sure that you understand what it is that you intend it to do when you invoke it. Please refer to the V5.5-000 release notes for more discussion of its operation. As FIS developed it as a step towards a much larger project, we anticipate that it will not be broadly useful as is.

  The following changes were made as part of GTM-7253:

  - When starting, the Update Server checks for concurrent ONLINE ROLLBACK and informs the Receiver Server to disconnect and re- establish connection. Previously, in some rare cases if a concurrent ONLINE ROLLBACK was in progress while the Update Server was starting, it might hang indefinitely.

  - The Receiver Server on a Supplementary Instance operates normally when connecting with a non-Supplementary Originating Instance all of whose updates are rolled back on the Supplementary Instance by MUPIP JOURNAL -ROLLBACK. In this case, the connection information for the instances remains intact. Previously, a Receiver Server resuming after such a ROLLBACK issued an inappropriate INSUNKNOWN error. The workaround was to restart the Receiver Server with -UPDATERESYNC qualifier.

  - GT.M issues an IGNBMPMRKFREE message to the operator log when a multi-node KILL bit map cleanup operation detects a concurrent ONLINE ROLLBACK. Previously, in certain rare cases, GT.M failed to record this event in the log.

  - GT.M resumes normally after a concurrent ONLINE ROLLBACK. Previously, in some rare cases after an ONLINE ROLLBACK, GT.M issued an inappropriate REPLREQROLLBACK error with a secondary message indicating that the instance file was corrupt. In this field test release there are still a few known issues.

  - GT.M appropriately releases the database critical section in case of concurrent ONLINE ROLLBACK or MUPIP REORG. In V5.5-000, in certain rare cases (involving ONLINE ROLLBACK or MUPIP REORG), GT.M could continue to hold a critical section for an extended period of time.

  - The Update Process handles concurrent ONLINE ROLLBACK while processing triggers. Previously, the Update Process did not detect ONLINE ROLLBACK when processing triggers and continued with a transaction restart which could cause a GTMASSERT.

  - The Receiver Server, if started with -autorollback, when the journal files are setup with NOBEFORE images, shuts down if signaled to rollback. In V5.5-000, if the journal files are setup with NOBEFORE images and the Receiver Server, started with -autorollback, receives a REPL_ROLLBACK_FIRST message, it could perform indefinite ONLINE FETCHRESYNC ROLLBACKs. The workaround was to shutdown the Receiver Server explicitly and refresh the Receiver Side with a BACKUP of the Source Side and restart replication.

  [UNIX] (GTM-7253)

- Replication from a Propagating Instance, including a Supplementary Instance to a Receiving Instance works more robustly. Previously in some rare cases, the Receiving Instance could have incorrect history records written in its instance file causing inappropriate STRMSEQMISMTCH errors. [UNIX] (GTM-7257)

- MUPIP REPLICATE -RECEIVE -START -UPDATERESYNC accepts the -INITIALIZE qualifier. For SI replication, -INITIALIZE specifies that this is the first connection between the instances, while -RESUME (mutually exclusive with -INITIALIZE) specifies that replication should continue from where the previous replication connection left off. Use of UPDATERESYNC when starting an SI replication connection requires an accompanying -INITIALIZE or -RESUME. MUPIP ignores these qualifiers when starting BC replication (that is, no updates permitted on the instance with the Receiver Server). In previous versions of GT.M, starting an SI replication connection with UPDATERESYNC without an explicit RESUME caused replication to operate as if it was an initial connection, which could result in replication starting from a different point from where the previous connection left off, potentially causing gaps in the data. You need to revise existing shell scripts to ensure explicit use of the INITIALIZE qualifier where the initial connection was implicit. FIS has determined that the additional protection against inadvertent error warrants making this change which is not strictly upward compatible.

  The -UPDNOTOK qualifier works as documented. In V5.5-000, any source server command involving -UPDNOTOK incorrectly terminated with a CLIERR error.

  MUPIP rejects conflicting -PROPAGATEPRIMARY and -UPDOK qualifiers on the same command and issues a CLIERR. In V5.5-000, a Source Server command with conflicting -UPDOK and -PROPAGATEPRIMARY qualifiers incorrectly proceeded as if only - UPDOK had been specified.

  A replication configuration of A->B->P where A->B replication is BC replication and B->P replication is SI replication works as documented. Previously, P's Receiver Server terminated with a PRIMARYNOTROOT error when connecting with B's Source Server. [UNIX] (GTM-7258)

- The replication source server log displays a correct value of "Start Seqno" in the message that describes the content of the history record sent across (this messages reads "New History Content ..."). In GT.M V5.5-000, in some cases, it used to display a "Start Seqno" that was less than what it actually sent across, which was the correct value. The corresponding message in the receiver log for the same time showed the correct value. This did not affect the correctness of replication but confused anyone who looked at both logs and found the discrepancy. [UNIX] (GTM-7265)

- MUPIP SET -JOURNAL now cuts the link to the previous-generation journal file if it was not properly closed due to a system crash, and the database has been the subject of a MUPIP RUNDOWN afterwards. Previously, when creating a new journal file upon a system crash and a subsequent database rundown, MUPIP SET -JOURNAL tried to link the created journal file back to the potentially invalid pre-crash journal file. Problems with integrity in the older journal file could result in a number of other journaling-related issues. Note that in the event of a crash FIS strongly recommends performing a MUPIP ROLLBACK on a database with replication, MUPIP JOURNAL RECOVER on a journaled database, and MUPIP RUNDOWN only if using neither journaling nor replication. To promote this decision-making model, in addition to REQRUNDOWN message in a process crash scenario with neither journaling nor replication enabled, on UNIX GT.M now issues a REQROLLBACK error in cases with replication and BEFORE IMAGE journaling, and REQRECOV error with NOBEFORE journaling. Together, REQRUNDOWN, REQROLLBACK and REQRECOV messages provide more context-specific instructions, thus better guiding the user in protecting and recovering data after a process crash. Previously, GT.M issued REQRUNDOWN error on a process crash regardless of journal and replication states. (GTM-7339)(GTM-7286)

- When journaling is off and replication is in the "WAS_ON" state for any target database, MUPIP JOURNAL -ROLLBACK -BACKWARD issues a REPLSTATEOFF and terminates immediately with a non-zero exit status. When the journal files are inaccessible, UNIX editions of MUPIP JOURNAL -ROLLBACK -BACKWARD or -RECOVER -BACKWARD issues a JNLACCESS error and terminates with a non-zero exit status. Running the journal file storage out of space most commonly causes both of these conditions. In prior versions for these cases, MUPIP terminated without reporting the issue and gave a success exit return even though it was not able to restore the database(s) to a wholesome state. (GTM-7294)

- Specifying -LOG=<file-name> with MUPIP REPLICATE -{RECEIVER |SOURCE} -STATSLOG={OFF | ON} generates a GTM-E-CLIERR, and with STATSLOG=ON the detailed status information appears in the Receiver Server or Source Server log. Previously specifying -LOG= with -STATSLOG caused the server to create the file specified with -LOG with a single entry, and could subsequently cause the Server to terminate with a memory segmentation fault (SIG-11). Although the revised behavior is not technically upward compatible, the previous behavior made the now disallowed combination at best useless and at worse disruptive. [UNIX] (GTM-7296) ✓

- MUPIP REORG correctly handles a concurrency issue discovered in the course of our test development. Previously, this very rare case could result in database damage. (GTM-7298)

- When reading multiple journal records from the journal file, the Source Server handles buffer expansion correctly. Previously, if buffer expansion occurred concurrently with an action that added history records to the instance file, the Source Server could corrupt its control structures and hang indefinitely. [UNIX] (GTM-7317)

- Duplicate of GTM-7286. (GTM-7339)

- The Source Server exits with one or more JNLFILOPN messages after six failed attempts to open journal files. The JNLFILOPN messages contain information on the journal files and regions that were not opened. The first five open failures generate REPL_WARN messages. Previously, the Source Server generated REPL_WARN messages every 10 seconds indefinitely and provided no information on which files were associated with the failure. [UNIX] (GTM-7344)

- The Source Server properly handles sending very large amounts of data after reestablishing a dropped connection to the Receiver. Previously this could lead to the Source Server sending invalid messages to the Receiver, possibly leading to data corruption, errors, or crashes.(GTM-7363)

- MUPIP JOURNAL -ROLLBACK on a supplementary instance works correctly even if there were no locally generated updates since the instance file was created or if the rollback undoes all the updates done to this instance since the instance file was created. Previously, the rollback command incorrectly initialized the Stream Sequence Number fields in the instance file header resulting in REPLINSTDBSTRM errors from the next source server startup command. The workaround for this issue was to repeat the rollback after which the source server startup worked fine. [UNIX] (GTM-7371)

- The Receiver Server avoids redundant FETCHRESYNC ONLINE ROLLBACKs. In V5.5-000, the Receiver Server, in some rare cases, did not wait for the Update Server to acknowledge an ONLINE ROLLBACK and could therefore perform multiple ONLINE ROLLBACKs, all but one of which were redundant. Note that although this could cause a momentary hiccup in replication throughput, the extraneous rollbacks could not cause any damage. [UNIX] (GTM-7378)

- With NOBEFORE_IMAGE journaling, at an EPOCH, GT.M does not flush global or journal buffers from memory to secondary storage or invoke file system hardening with an fsync system call. Previously, GT.M used flushing and fsync at an EPOCH for all types of journaling, even though they did not add value for NOBEFORE_IMAGE journaling, and unnecessarily limited performance for that mode of journaling. [UNIX] (GTM-7383)

- MUPIP ENDIANCVT now correctly recognizes directory tree leaf blocks. Previously, if a such a block lay at an offset approximately 4GB past the beginning of the database file and contained only one record, ENDIANVCT would incorrectly identify it as global variable tree block. ENDIANCVT would therefore leave the block pointer value unconverted, causing a subsequent MUPIP INTEG to fail. [UNIX] (GTM-7390)

- MUPIP JOURNAL RECOVER or ROLLBACK handles abnormally terminated journal files (for example, system crash) much more efficiently. Previously, in these scenarios, it could expend more effort than needed in the backward processing phase trying to figure out the last valid record in the journal file. There was no correctness issue. It just look longer to run than it needed to. [OpenVMS] (GTM-7393)

- MUPIP SET -PARTIAL_RECOV_BYPASS resets the file_corrupt flag in the database file header. In V5.5-000, this command, in some rare cases, did not reset the file_corrupt flag thereby leading to operational inconvenience. [UNIX] (GTM-7396)

- MUPIP JOURNAL ROLLBACK plays forward NULL journal records (usually created by replication filters) with the correct journal sequence number (and stream sequence number if any). Previously the replay wrote NULL journal replacement records with a different journal sequence number than the original records causing various issues including for example the source server issuing incorrect REPLBRKNTRANS errors while replicating. (GTM-7398)

- If any errors occur during Source Server startup (like REPLINSTFMT), the error handling logic appropriately cleans up IPC resources. Previously. errors during Source Server startup left IPC resources around unnecessarily complicating subsequent operation. [UNIX] (GTM-7401)

- GT.M correctly maintains the KILLs in Progress and Abandoned KILLs counters (displayed by DSE DUMP -FILE) when a KILL command encounters an error. Previously, if a KILL command encountered an error, say a GBLOFLOW, it could increment the counter for one region and decrement the counter in another region, where one or both of the regions might not have held the argument of the KILL, resulting in subsequent inappropriate INTEG errors in one or two regions. (GTM-7402)

- MUPIP REORG -TRUNCATE appropriately handles a rare concurrency conflict. Previously, due to rare concurrent circumstances, it could fail with a GTMASSERT2 error.

  **Caution**

  In V5.5-000 and V6-0.000, GT.M has a known limitation that it does not detect concurrent MUPIP BACKUP and MUPIP REORG -TRUNCATE and so doing can produce a defective backup that has integrity errors.

  [UNIX] (GTM-7444)

## Utilities-Other Than MUPIP

- DSE and LKE start even when another process holds a resource needed to connect to existing shared memory associated with the database. They do this by bypassing the logic used to gain control of the shared resource after a wait of 3 seconds. This does not work if they cannot determine that the shared memory has been properly initialized which they know if there are at least two attached processes. When DSE and LKE bypass the resource control, they also skip resource cleanup when they exit. This means you need to do a MUPIP RUNDOWN to clean up the resources if the bypassing process becomes the last process using the database shared memory to detach as they exit. Previously DSE and LKE could hang waiting for an unavailable shared resource. Note that if DSE and LKE do not start with a bypass, they may still hang on exit if they find the shared resource unavailable, but because they can be presumed to have completed their task(s), this should not pose a significant problem. [UNIX] (GTM-4053)

- The OpenVMS distribution kits include a README.txt file which clarifies license related text in the header of each source file and its relationship to the COPYING file. Previously, the kit did not provide this information. [OpenVMS] (GTM-4820)

- The Open Source release of GT.M places the GT.M encryption plug-in shell scripts in the plug-in directory. Previously the build Makefile did not copy the encryption plug-in scripts and routines into the documented locations. (GTM-7071)

- For the GDE operations involving null subscripts (including add, change and template), when the -null qualifier is set as a distinguishable prefix of one of the following five strings: 'always','true','never','false' or 'existing', (e.g. -null=exist), the GDE processes the null subscripts in the same way as setting the -null qualifier as the complete string (e.g. -null=existing). Previously, if the -null qualifier was set as a distinguishable prefix of one of the above five strings, the GDE always set the null subscripts as "NEVER". (GTM-7184)

- In operator log messages from GT.M, the facility starts with "GTM-" followed by a component designation, and, if there is an Instance File or Replication Journal Pool present, another dash and the instance name, as shown in the following table:

| **Component** | **Instance File $char(10)o Replicatio $char(10)J Pool** | **Receiver Pool** | **Designation** |
|---|---|---|---|
| Source Server | Y | N/A | SRCSRVR |
| | N | N/A | MUPIP |
| Receiver Server | Y | N/A | RCVSRVR |
| | N | N/A | MUPIP |
| Update Process | Y | N/A | UPD |
| | N | N/A | MUPIP |
| Reader Helper | N/A | Y | UPDREAD |
| | N/A | N | UPDHELP |
| Writer Helper | N/A | Y | UPDWRITE |
| | N/A | N | UPDHELP |

In prior releases, the facility was just "GTM". [UNIX](GTM-7219) ✓

- The gtmsecshr wrapper validates permissions on the gtmsecshrdir directory and gtmsecshr executable. Previously, the gtmsecshr wrapper did not validate these permissions, leading to a local root shell vulnerability. FIS previously supplied a revised gtmsecshr with this change for use with prior versions. [UNIX] (GTM-7312)
- DSE CACHE -VERIFY avoids inappropriate cache verification messages. Previously, in certain rare cases, DSE CACHE -VERIFY incorrectly reported transient control structure problems. [UNIX] (GTM-7338) ✓
- GDE SHOW -COMMAND consistently displays the correct value for the NULL_SUBSCRIPTS setting JOURNAL BUFFER_SIZE. Previously, it displayed the corresponding values for the template rather than the region. (GTM-7348) ✓
- The gtmstart, gtmstop and gtcm_run scripts follow the convention for executable scripts and explicitly invoke the Bourne shell on the very first line. Previously, these scripts ran in the calling shell which may not be Bourne shell compatible. [UNIX] (GTM-7403) ✓
- The gtminstall.sh installation script creates a new configure.sh each time it runs. Previously, repeated executions of gtminstall.sh from the same directory incorrectly duplicated the installation commands in configure.sh. [UNIX] (GTM-7405)

## CUSTERRNOTFND

*CUSTERRNOTFND, Error mnemonic eeee specified in custom errors file is not valid for this version of GT.M*

Run Time Error: This error indicates that the GT.M runtime did not recognize the error mnemonic eeee in the file referenced by $gtm_custom_errors.

Action: Modify the file so that it no longer contains the invalid mnemonic or set the gtm_custom_errors environment variable to point to an appropriate file.

## CUSTERRSYNTAX

*CUSTERRSYNTAX, Syntax error in file ffff at line number nnnn*

Run Time Error: This error indicates that the custom errors file ffff contains an inappropriate syntax on line nnnn.

Action: Modify the file ffff so that it contains a single valid error mnemonic on line nnnn or set the gtm_custom_errors environment variable to point to an appropriate file.

## CUSTOMFILOPERR

*CUSTOMFILOPERR, Error while doing oooo operation on file ffff*

Run Time Error: This indicates that the operating system reported an error while performing operation oooo on custom errors file ffff.

Action: Check that ffff is a proper path to the custom errors file. If it is incorrect, set the gtm_custom_errors environment variable to point to the correct file. If the file path is correct, verify that the user has access to the file and correct any permissions issues.

## DBRSIZMN ⚠

*DBRSIZMN, xxxx Physical record too large*

Run Time Warning: This indicates that a DSE or MUPIP INTEG command failed because block xxxx contains a record that does not meet the minimum size requirement.

Action: Report this database structure error to the group responsible for database integrity at your operation.

## DBRSIZMX ⚠

*DBRSIZMX, xxxx Physical record too large*

Run Time Warning: This indicates that a DSE or MUPIP INTEG command failed because block xxxx contains a record that exceeds the maximum record size (1MB) for a GDS database.

Action: Report this database structure error to the group responsible for database integrity at your operation.

## DBSPANGLOINCMP

*DBSPANGLOINCMP, xxxx Spanning node is missing. Block no yyyy of spanning node is missing*

MUPIP Error: This is a MUPIP INTEG error. Refer to the MUPIP INTEG Errors section of the Messages and Recovery Procedures manual.

## DBSPANCHUNKORD

*DBSPANCHUNKORD, xxxx Chunk of yyyy blocks is out of order*

MUPIP Error: This is a MUPIP INTEG error. Refer to the MUPIP INTEG Errors section of the Messages and Recovery Procedures manual.

## DSKNOSPCAVAIL

*DSKNOSPCAVAIL, Attempted write to file FFFF failed due to lack of disk space. Retrying indefinitely.*

Run time error: This error indicates that GT.M could not update file FFFF due to lack of disk space in the file system.

Action: Make disk space available in the file system to allow updates to file FFFF.

## DSKNOSPCBLOCKED

*DSKNOSPCBLOCKED, Retry of write to file FFFF suspended due to new instance freeze. Waiting for instance to be unfrozen.*

Run time error: This error indicates that a process waiting for space to write to file FFFF determined that another process froze the replication instance. The process will not make more attempts to write to the file until the replication instance is unfrozen.

Action: Check the system log for the most recent REPLINSTFROZEN message to determine the cause of the current freeze and resolve it.

## DSKSPCAVAILABLE

*DSKSPCAVAILABLE, Write to file FFFF succeeded after out-of-space condition cleared.*

Informational message: This indicates that the file system of file FFFF has enough space to allow further updates (previously not possible as indicated by a DSKNOSPCAVAIL error message).

Action: None Required.

## DBIOERR

*DBIOERR, Error while doing write operation on region rrrr (ffff)*

Runtime error: This error indicates that the process encountered an I/O error (other than ENOSPC) while trying to write to database file ffff (corresponding to region rrrr).

Action: Examine accompanying messages to identify the cause of the I/O error and take actions to rectify it.

## ENOSPCQIODEFER

*ENOSPCQIODEFER, Write to file FFFF deferred due to lack of disk space*

Informational message: This indicates GT.M chose to defer updating the file FFFF to avoid a possible deadlock. GT.M uses this message only if the environment is configured for Instance Freeze.

Action: None.

## DBDATAMX

*DBDATAMX, xxxx Record too large*

MUPIP Error: This is a MUPIP INTEG error. Refer to the MUPIP INTEG Errors section of the Messages and Recovery Procedures manual.

## DBFHEADERRANY

*DBFHEADERRANY, Database file ffff: control problem: aaaa was xxxx expecting yyyy*

Run time informational message: This indicates that database cache recovery was triggered due to an abnormal event and the recovery routine detected damage to the control structures in the database global buffer cache or the file header.

Action: The system automatically attempts to correct the problem. If this error continues to occur, attempt MUPIP RUNDOWN and if that fails too, restore database from backup and replay journal files. If necessary, report the entire incident context to your GT.M support channel.

## GTMSECSHRBADDIR

*gtmsecshr is not running from $gtm_dist/gtmsecshrdir or $gtm_dist cannot be determined*

Run Time Error: This message indicates an inappropriate gtmsecshr invocation. Either gtmsecshr is improperly installed or an inappropriate access attempt is underway.

Action: Verify that GT.M (and gtmsecshr) are correctly installed following FIS documented procedures and that filesystem mount points have not changed. If GT.M is correctly installed and filesystem mount points have not changed, investigate this as an attempt to break system security.

## GTMSECSHRDEFLOG

*$gtm_log is either undefined or not defined to an absolute path, so gtm_log is set the default xxxx*

Obsolete Error: No longer issued by GT.M.

## GTMSECSHRISNOT

*gtmsecshr is not running as gtmsecshr but xxxxx - must be gtmsecshr*

Run Time Error: gtmsecshr is running with a name other than the one it is allowed to run by design.

Action: Verify that GT.M (and gtmsecshr) are correctly installed following FIS documented procedures and that filesystem mount points have not changed. If GT.M is correctly installed and filesystem mount points have not changed, investigate this as an attempt to break system security.

## GTMSECSHRLOGF

*XXXX - YYYY; Error while creating GTMSECSHR log file*

Obsolete Error: No longer issued by GT.M.

## GTMSECSHRLOGSWH

*GTMSECSHRLOGSWH, Error switching GTMSECSHR log file*

Obsolete Error: No longer issued by GT.M.

## GTMSECSHRNOARG0

*gtmsecshr cannot identify its origin - argv[0] is null*

Run Time Error: This message occurs when gtmsecshr is called in an inappropriate manner by facilities other those allowed by design (like the gtmsecshr wrapper).

Action: Investigate this as an attempt to break system security.

## GTMSECSHRSRVF

*GTMSECSHRSRVF, xxxx - yyyy; Attempt to service request failed (retry = yyyy)*

Run Time Error: This indicates that a GT.M process with PID yyyy was unable to communicate with gtmsecshr.

Action: This message is displayed when a process needs service from gtmsecshr, cannot communicate with gtmsechsr, or cannot start one. While the most likely cause is a mismatch in the value of the gtm_tmp environment variable between the GT.M process and the gtmsecshr process, examples of other causes include removal of socket files used for communication between GT.M and gtmsecshr processes. Check for a following associated message in syslog or in the stderr of the GT.M process.

## GTMSECSHRSRVFID

Previous: *xxxx: yyyy - Attempt to service request failed. Client ID: zzzz, mesg ID: aaaa, mesg code: bbbb.*

Revised: *xxxx: yyyy - Attempt to service request failed. Client ID: zzzz, mesg ID: aaaa, mesg code: bbbb*

## GTMSECSHRSRVFIL

Previous: *xxxx: yyyy; Attempt to service request failed. Client ID: zzzz, mesg type: aaaa, file: bbbb.*

Revised: *xxxx: yyyy; Attempt to service request failed. Client ID: zzzz, mesg type: aaaa, file: bbbb*

## INVCMD

*INVCMD, Invalid command keyword encountered*

Compile Time Error or Warning: This indicates that the program attempted to use an invalid keyword where a command was expected.

Action: Look for typographical errors or improper command abbreviations.

## JNLBUFFDBUPD

*JNLBUFFDBUPD, Journal file buffer size for database file dddd has been adjusted from xxxx to yyyy*

MUPIP Warning: The journal buffer size specified by the user for the database file dddd or previously stored in the database file header fell outside the permissible range, and was automatically adjusted, up or down, from xxxx to a legitimate value of yyyy.

Action: None Required.

## JNLBUFFREGUPD

*JNLBUFFREGUPD, Journal file buffer size for region rrrr has been adjusted from xxxx to yyyy*

MUPIP Warning: The journal buffer size specified by the user for the region rrrr or previously stored in the corresponding database file header fell outside the permissible range, and was automatically adjusted, up or down, from xxxx to a legitimate value of yyyy.

Action: None Required.

## JNLBUFFTOOLG

*Journal file buff e r xxxx is greater than the maximum allowed size of yyyy. Journal file not created.*

Obsolete Error: No longer issued by GT.M

## JNLBUFFTOOSM

*Journal file buffer xxxx is less than minimum of database block size in 512 byte pages + 1 (yyyy)*

Obsolete Error: No longer issued by GT.M

## KEYFORBLK

*KEYFORBLK, But block size bbbb can only support key size kkkk*

GDE Error: The maximum key for a region must fit in the block size less record overhead and any reserved bytes for that region; kkkk is the maximum key size for block size bbbb.

Action: Reduce the key size or reserved bytes or increase the block size.

## MUFILRNDWNFL2

*MUFILRNDWNFL2, Database section (id = dddd) belonging to database file ffff rundown failed*

MUPIP Error: This error indicates that an argumentless MUPIP RUNDOWN failed for database ffff and could not safely remove the shared memory with ID dddd from the system.

Action: Refer to accompanying for more detail on why the argumentless MUPIP RUNDOWN failed.

## MUINSTFROZEN

*MUINSTFROZEN, tttt : Instance iiii is frozen. Waiting for instance to be unfrozen before proceeding with writes to database file ffff*

MUPIP Runtime Informational message: This indicates the process attempting a write to database file ffff finds the instance iiii frozen (due to either a manual or an anticipatory freeze action). All writes suspend until the instance is unfrozen.

Action: Examine the cause of the Instance Freeze and take necessary actions to unfreeze the instance.

## INITORRESUME

*INITORRESUME, UPDATERESYNC on a Supplementary Instance must additionally specify INITIALIZE or RESUME*

MUPIP Error: Issued by a Receiver Server when started with -UPDATERESYNC on a Supplementary Instance which allows local updates, but started without specifying either -INITIALIZE or -RESUME.

Action: Additionally specify -INITIALIZE if this is the first time this supplementary instance is connecting to the source side OR if the receiver side databases have been refreshed from a backup of the source side. If on the other hand, the receiving instance had already been replicating from the source before and only had its instance file recreated in between, -RESUME might be appropriate with the -UPDATERESYNC. Check -RESUME documentation for more details.

## MUINSTUNFROZEN

*MUINSTUNFROZEN, tttt : Instance iiii is now unfrozen. Continuing with writes to database file ffff*

MUPIP Runtime Informational message: This indicates that the instance iiii (that was previously frozen) is now unfrozen and the MUPIP operation can continue with writes to database file ffff that were previously suspended. Additionally, tttt provides the time stamp at which the MUPIP operation noticed the unfrozen instance.

Action: None needed.

## LOCKINCR2HIGH

*Attempt to increment a LOCK more than LLLL times*

Runtime Error: This message indicates that a LOCK + command attempted to increase a LOCK increment higher than LLLL (the maximum level). The following associated message gives the resource name for the LOCK that failed.

Action: Examine the application for pathological use of incremental LOCKs (LOCK +) and ensure that no process LOCKs a single resource more than 511 increments with no intervening decrements (LOCK -) for that resource or lock releases (LOCK with no + or - in its argument).

## LOCKIS

*Resource name: RRRR*

Runtime Information: This message identifies a lock resource.

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

## LOCKSUB2LONG

*LOCKSUB2LONG, Following subscript is xxxx bytes long which exceeds 255 byte limit.*

Runtime error: This indicates that one of the substrings of a lock is taking more than 255 bytes. Check the following message to see which substring caused this error.

Action: Make sure none of the substrings are larger than 255 bytes. If UTF-8 is enabled, use the encoded byte length rather the character length for the key size.

## MUTEXLCKALERT

*MUTEXLCKALERT, Mutual Exclusion subsystem ALERT - Lock attempt threshold crossed for region rrrr. Process pppp is in crit cycle cccc.*

Run Time Warning: This warning indicates that process attempting to access a critical section lock for rrrr waited longer than the GT.M determined threshold (approximately two minutes) to obtain the critical section lock held by process pppp; the cycle is measure of the frequency of use of the lock.

GT.M produces this warning when:

- There is an IO bottleneck that caused GT.M to slow down: GT.M detects that process pppp is currently using the critical section lock.
- A process owning a critical section dies (most likely because of a kill -9) and the OS gives its PID to another process. To reclaim the inappropriately held critical section, GT.M first checks whether the process is alive and whether it holds hold the critical section. On finding that the process is alive but does not hold the critical section, GT.M concludes that it is not safe to free the critical section and alerts the operator with this message.

Action: Monitor the system to determine whether there is a process with process id pppp and whether that process is a GT.M process.

Implement a script to get a stack trace for process pppp or take other appropriate action and use the $gtm_procstuckexec environment variable to activate it immediately before the block process sends the MUTEXLCKALERT message.

Identify and terminate process pppp to release the permissions for that resource. If the process is a GT.M process, use a MUPIP STOP to terminate it. If a process of another application, use an appropriate mechanism to stop it.

If this message is due to an IO bottleneck, adopt a strategy that reduces IO. Some of the IO reducing strategies are:

- Revisit your database configuration parameters (especially block size, number of global buffers, journal buffers, and so on) to see if you can make improvements.
- Create separate region (database) for temporary globals and do not replicate them.

- Consider whether a different database access method and journaling strategy could improve throughput while satisfying your operational needs.
- For application configurations with large numbers of concurrent processes and/or large process memory footprints, consider placing object code in shared libraries on GT.M editions that support it. This may free system memory which the OS can use for its file system cache, or which you can use to increase the number of global buffers.

**Important**

Do not apply IO reduction strategies all at once. Try them one at a time and always verify/measure the results of each strategy.

## ORLBKINPROG

*ORLBKINPROG, Online ROLLBACK in progress by PID pppp in region rrrr*

Run time informational message: This message in the operator log indicates an online rollback has been in progress by process pppp on region rrrr for more than 30 seconds.

Action: None Required.

## RCVR2MANY

*RCVR2MANY, The instance already has the maximum supportable number of receiver servers nnnn active*

Obsolete Error: No longer issued by GT.M

## REPLINSTFREEZECOMMENT

*REPLINSTFREEZECOMMENT, Freeze Comment: xxxx*

Runtime Information: This message contains details about a freeze on a replication instance. The instance information is included in an associated REPLINSTFROZEN message. In the case of an automatic freeze, xxxx identifies an error that triggered the freeze. In the case of an administrative freeze, xxxx contains the text provided by MUPIP REPLICATE -SOURCE -FREEZE=ON -COMMENT="xxxx".

Action: Refer to REPLINSTFROZEN.

## REPLINSTFROZEN

*REPLINSTFROZEN, Instance xxxx is now Frozen*

Runtime Error: This indicates that the replication instance xxxx is frozen due to a custom error or an out-of-space condition on a region with INST_FREEZE_ON_ERROR set, or by an administrator with the MUPIP REPLICATE -SOURCE -FREEZE=ON command. Updates to database files or shared memory for regions in the instance ar e blocked.

Action: Check the associated REPLINSTFREEZECOMMENT message for details on the cause of the freeze. For out-of-space conditions, make sufficient disk space available to remove the freeze. For custom errors or for administrative freezes, MUPIP REPLICATE -SOURCE -FREEZE=OFF or a system restart removes the freeze.

## REPLINSTUNFROZEN

*REPLINSTUNFROZEN, Instance xxxx is now Unfrozen*

Runtime Information: This indicates that a replication instance which had previously been frozen is no longer frozen and updates to regions in the instance will resume.

Action: None required.

## REQROLLBACK

*Error accessing database dddd. Run MUPIP JOURNAL ROLLBACK on cluster node ccccc.*

Runtime Error: This indicates that GT.M could not open a previously replicated database file dddd due to a prior improper shutdown on cluster node ccccc. 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 to cleanup the instance file, database, and journal files before starting a source server on this instance.

## REQRUNDOWN

*Error accessing database dddd. Must be rundown on cluster node ccccc.*

Runtime Error: This indicates that GT.M could not open database file dddd due to a prior improper shutdown on cluster node ccccc. 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 RUNDOWN from cluster node ccccc. If MUPIP RUNDOWN with no parameters does not work, specify the region name or file-specification with -REGION or -FILE, respectively.

## REQRECOV

*Error accessing database dddd. Must be recovered on cluster node ccccc.*

Runtime Error: This indicates that GT.M could not open a previously journaled database file dddd due to a prior improper shutdown on cluster node ccccc. 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 a MUPIP JOU RNAL RECOVER operation to address this issue.

## RESRCINTRLCKBYPAS

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

Runtime Information: LKE or DSE automatically bypassed FTOK or access control semaphore. xxxx 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: Bypassing processes never flush or take down shared memory. For this reason, make sure to wait until all processes release the database and perform a "MUPIP RUNDOWN" to get the database to a safe state.

## RESCRCWAIT

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

Runtime 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. ssss identifies the semaphore type: "FTOK" or "access control"; rrrr is the region; ffff is the file corresponding to region rrrr; pppp is the PID of the hanging process; ssss is the semaphore ID.

Action: None required.

## RLBKCONFIGBNDRY

*Rollback encountered journal records indicating current source iiii replaced old source oooo; cannot rollback past sequence number ssss*

Obsolete Error: No longer issued by GT.M

## RNDWNSKIPCNT

*A total of nnnn process(es) skipped database rundown due to a concurrent ONLINE ROLLBACK*

Obsolete Error: No longer issued by GT.M