GT.M

V6.0-002 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: gtmsupport@fisglobal.com
Automated attendant for 24 hour support: +1 (610) 578-4226
Switchboard: +1 (610) 296-8877
Website: http://fis-gtm.com

Legal Notice

Revision History		
Revision 1.1	28 January 2014	Added GTM-6899.
Revision 1.0	03 May 2013	V6.0-002 - First published version.

Table of Contents

V6.0-002 1
- Overview 1
- Conventions 1
- Platforms 3
  - Platform support lifecycle 5
- Migrating to 64-bit platforms 5
  - Call-ins and External Calls 5
  - Internationalization (Collation) 6
  - Environment Translation 6
- Recompile 6
- Rebuild Shared Libraries or Images 7
- Additional Installation Instructions 7
  - UNIX 7
  - OpenVMS 8
- Upgrading to GT.M V6.0-002 8
  - Stage 1: Global Directory Upgrade 8
  - Stage 2: Database Files Upgrade 9
  - Stage 3: Replication Instance File Upgrade 11
  - Stage 4: Journal Files Upgrade 12
  - Stage 5: Trigger Definitions Upgrade 12
  - Downgrading to V5 or V4 13
- Managing M mode and UTF-8 mode 14
  - Compiling ICU 15
- Setting the environment variable TERM 15
- Installing Compression Libraries 16

Change History 19
- V6.0-002 19

M-Database Access 25

M-Other Than Database Access 29

Utilities-MUPIP 35

Utilities-Other Than MUPIP 39

Error and Other Messages 41
- BADZPEEKARG 41
- BADZPEEKFMT 41
- BADZPEEKRANGE 41
- DBBADFREEBLKCTR ● 42
- DBMBMINCFREFIXED 42
- FALLINTOFLST 42
- FREEBLKSLOW 42
- FTOKKEY 43
- GTMSECSHRDMNSTARTED 43
- GTMSECSHRGETSEMFAIL 43
- GTMSECSHRREMFILE 43
- GTMSECSHRREMSEM 43
- GTMSECSHRREMSEMFAIL 43

GTMSECSHRREMSHM 44
GTMSECSHRSEMGET 44
GTMSECSHRSHMCONCPROC 44
GTMSECSHRUPDDBHDR 44
HOSTCONFLICT 45
INDRCOMPFAIL ● 45
JNI 45
JNLQIOSALVAGE 45
MALLOCMAXUNIX ● 45
MALLOCMAXVMS ● 46
MMFILETOOLARGE ● 46
MMREGNOACCESS 46
MUUSERECOV 46
MUUSERLBK 47
NULLENTRYREF 47
SECSHRCHDIRFAILED 47
SECSHRCLEARENVFAILED 47
SECSHREXECLFAILED 47
SECSHRGTMDBGLVL2LONG 47
SECSHRGTMDIST2LONG 48
SECSHRGTMTMP2LONG 48
SECSHRNOGTMDIST 48
SECSHRNOTOWNEDBYROOT 48
SECSHRNOTSETUID 48
SECSHRPERMINCRCT 49
SECSHRSETGTMDISTFAILED 49
SECSHRSETGTMTMPFAILED 49
SECSHRSETUIDFAILED 49
SECSHRSTATFAILED 49
SECSHRWRITABLE 49
SEMID 50
ZPEEKNORPLINFO 50

V6.0-002

Overview

V6.0-002 focuses on smaller enhancements, improvements to ease of use and robustness, and bug fixes but brings several useful enhancements to GT.M. For example:

- Support for call-out to, and call-in from Java in the same process. The Java-side support and examples are available as a separate package.
- A compiler option to reduce the process memory footprint of literals.
- A FOLLOW deviceparameter for sequential disk devices enables monitoring of files that are concurrently being updated by other processes.
- A MUPIP SET configuration control to expand GT.M's ability to support larger numbers of processes concurrently accessing databases.

Conventions

This document uses the following conventions:

	UNIX	**OpenVMS**
Syntax	lower case	Both lower case and upper case
Flag/Qualifiers	-	/
Program Names or Functions	upper case. For example, MUPIP BACKUP	
Examples	lower case. For example: mupip backup -database ACN,HIST /backup	
Reference Number	A reference number is used to track software enhancements and support requests. It is enclosed between parentheses ().	
Platform Identifier	[UNIX]	[OpenVMS]
	The platform identifier does not appear for those new features or enhancements that apply to both the platforms.	

Note

The term UNIX refers to the general sense of all platforms on which GT.M uses a POSIX API. As of this date, this includes: AIX, HP-UX on IA64, GNU/Linux on x86 and x86_64 and Solaris on SPARC.

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

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

Effective V6.0-000, GT.M documentation 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.

✓ denotes a new feature that requires updating the manuals.

✓ denotes a new feature or an enhancement that is not upward compatible and might affect your application code.

● denotes deprecated error messages.

● denotes revised error messages.

Platforms

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

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)	-
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. A GT.M database file on this platform can grow to a maximum size of 1TiB-512 bytes. 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. 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

Platform	Supported Versions	Notes
		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. While GT.M supports both UTF-8 mode and M mode on this platform, there are problems with the AIX ICU utilities that prevent FIS from testing 4-byte UTF-8 characters as comprehensively on this platform as we do on others. Running GT.M on AIX 7.1 requires APAR IZ87564, a fix for the POW() function, to be applied. To verify that this fix has been installed, execute **instfix -ik IZ87564.**
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 12.04 LTS; SuSE Linux Enterprise Server 11	To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware. GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.5-24 or later) and ncurses (version 5.5 or later). To install GT.M with Unicode (UTF-8) support on RHEL 6, in response to the installation question **Should an ICU version other than the default be used? (y or n)** please respond **y** and then specify the ICU version (for example, respond 3.6) to the subsequent prompt **Enter ICU version (ICU version 3.6 or later required. Enter as major-ver.minor-ver):** GT.M requires the libtinfo library. If it is not already installed on your system, and is available using the package manager, install it using the package manager. If a libtinfo package is not available (for example on SuSE 11): • Find the directory where libncurses.so is installed on your system. • Change to that directory and make a symbolic link to libncurses.so.<ver> from libtinfo.so.<ver>. Note that some of the libncurses.so entries may themselves be symbolic links, for example, libncurses.so.5 may itself be a symbolic link to libncurses.so.5.9.
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.

Platform	Supported Versions	Notes
		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 V6.0-002 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.0-002_arch (for example, /usr/lib/fis-gtm/V6.0-002_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.0-002_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 V6.0-002

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

GT.M upgrade procedure for V6.0-002 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 V6.0-002 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 V6.0-002.

- 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 V6.0-002.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

Stage 2: Database Files Upgrade

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

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

A 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 V6.0-002 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 or above (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 -FILEHEADER -FULLY_UPGRADED=1 command to stop the warnings.

Caution

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

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

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

For additional upgrade considerations, refer to Database Compatibility Notes.

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

You need to upgrade your database files only when there is a block format upgrade from V4 to V5. However, some versions, for example, database files which have been initially been created with V4 (and subsequently upgraded to a V5 format) may additionally need a MUPIP REORG -UPGRADE operation to upgrade previously used but free blocks that may have been missed by earlier upgrade tools.

- Upgrade your database files using in-place or traditional database upgrade procedure depending on your situation. For more information on in-place/traditional database upgrade, see Database Migration Technical Bulletin.

- Run the MUPIP REORG -UPGRADE command. This command upgrades all V4 blocks to V5 format.

Note

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

Database Compatibility Notes

- Changes to the database file header may occur in any release. GT.M automatically upgrades database file headers as needed. Any changes to database file headers are upward and downward compatible within a major database release number, that is, although processes from only one GT.M release can access a database file at any given time, processes running different GT.M releases with the same major release number can access a database file at different times.
- Databases created with V5.3-004 through V5.5-000 can grow to a maximum size of 224Mi (234,881,024) blocks. This means, for example, that with an 8KiB block size, the maximum database file size is 1,792GiB; this is effectively the size of a single global variable that has a region to itself; a database consists of any number of global variables. A database created with GT.M versions V5.0-000 through V5.3-003 can be upgraded with MUPIP UPGRADE to increase the limit on database file size from 128Mi to 224Mi blocks.
- Databases created with V5.0-000 through V5.3-003 have a maximum size of 128Mi (134, 217,728) blocks. GT.M versions V5.0-000 through V5.3-003 can access databases created with V5.3-004 and later as long as they remain within a 128Mi block limit.
- Database created with V6.0-000 or above have a maximum size of 1,040,187,392(992Mi) blocks.

Stage 3: Replication Instance File Upgrade

V6.0-002 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.0-002 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 V6.0-002, 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 V6.0-002. This is necessary because multi-line XECUTEs for triggers require a different internal storage format for triggers which makes triggers created in V5.4-000/V5.4-000A/V5.4-001 incompatible with V5.4-002/V5.4-002A/V5.4-002B/V5.5-000/V6.0-000/V6.0-001/V6.0-002.

To extract and reapply the trigger definitions on V6.0-002 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 V6.0-002 to reload your trigger definitions.

To extract and reload trigger definitions on a V6.0-002 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 V6.0-002.
6. Run **mumps -run %XCMD 'if $ztrigger("file","trigger_defs.trg")'** to reapply your trigger definitions.
7. Turn replication on.
8. Connect to the originating instance.

Note

Reloading triggers renumbers automatically generated trigger names.

Downgrading to V5 or V4

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

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

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

-VERSION specifies the desired version for the database header.

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

1. The database was created with a major version no greater than the target version.
2. The database does not contain any records that exceed the block size (spanning nodes).
3. The sizes of all the keys in database are less than 256 bytes.
4. There are no keys present in database with size greater than the Maximum-Key-Size specification 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 V6.0-002 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_V6.0-002_i686, then gtmprofile and gtmcshrc set $gtm_dist to /usr/lib/fis-gtm/gtm_V6.0-002_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"
```

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) or LD_LIBRARY_PATH (other UNIX platforms) includes the directory containing the library. The Source and Receiver Server link the shared library at runtime. If this fails for any reason (such as file not found, or insufficient authorization), the replication logic logs a DLLNOOPEN error and continues with no compression.

Change History

V6.0-002

Fixes and enhancements specific to V6.0-002 are:

Id	Prior Id	Category	Summary
GTM-1937	C9902-000876	MUMPS	Calling into gtm_start_timer() exposed function with a non-positive expiration value schedules a timer for 10 ms
GTM-3220	C9A07-001564	MUMPS	Java interface and updates to the C interface for call-in/out ✓
GTM-4324	C9C05-001992	MUMPS	Bad syntax such as SET a="@",@a="bad" reported rather than ignored
GTM-5284	S9E02-002422	MUMPS	Syntax errors for extra indirection characters
GTM-6181	C9I05-002988	Other Utilities	LKE accepts $[Z]CHAR() in resource names
GTM-6330	C9J01-003083	MUMPS	-DYNAMIC_LITERALS compiler qualifier can reduce process memory footprint ✓
GTM-6548	C9K02-003236	MUMPS	Create .mje and .mjo files only at appropriate location
GTM-6717	S9K11-002797	MUMPS	GT.M correctly reports $ZREALSTOR, $ZALLOCSTOR, and $ZUSEDSTOR ISVs
GTM-6802	C9L04-003404	MUMPS	GT.M processes no longer hang or terminate abnormally if interrupted by a signal within a specific time window
GTM-6892	-	DB	Prevent a GTMASSERT that occurs when journal file synchronization from memory to disk encounters an error
GTM-6899	-	DB	Invoke triggers for globals having null subscripts.
GTM-7054	-	DB	Secondary journal file size is now more comparable to the primary's journal file size
GTM-7093	-	DB	Proper handling of attempts to access an inaccessible trigger

Id	Prior Id	Category	Summary
GTM-7268	-	DB	Close journaling when detecting out of disk space during journal buffer flushing
GTM-7381	-	DB	Journal flush retries unlimited
GTM-7412	-	DB	Operator log warnings for files with EXTENSION disabled getting full ✓
GTM-7420	-	MUPIP	MUPIP RUNDOWN may issue MUUSERLBK or MUUSERECOV error in case of a crashed replication/journaling-enabled database ✓
GTM-7447	-	MUPIP	Rundown failure messages are now of error (E) type.
GTM-7490	-	DB	Replaced CLSTCONFLICT with HOSTCONFLICT
GTM-7503	-	DB	Release database connect / disconnect shared resource on an error
GTM-7518	-	MUPIP	Additional MUPIP Error Output
GTM-7540	-	MUMPS	New [NO]FOLLOW deviceparameter for OPEN and USE of sequential devices ✓
GTM-7545	-	Other Utilities	DSE does not attach to a region on VERMISMATCH error
GTM-7548	-	DB	Instance Freeze uses active region for freeze criteria
GTM-7568	-	MUPIP	MUPIP RUNDOWN does not issue an error if another process concurrently removes an identifier RUNDOWN has decided to remove
GTM-7586	-	DB	Idle EPOCH records get correctly written for the MM access method.
GTM-7588	-	MUMPS	M-tracing correctly reports elapsed time
GTM-7592	-	MUPIP	MUPIP REORG and MUPIP SIZE SCAN handle MM databases
GTM-7595	-	DB	Reduce the amount of memory used by each process for journal records it generates
GTM-7596	-	Other Utilities	%DSEWRAP improvements

Id	Prior Id	Category	Summary
GTM-7599	-	DB	A read-only process can be the first one to open an MM database
GTM-7609	-	MUMPS	Fix for inappropriate INDEXTRACHARS errors
GTM-7611	-	MUMPS	GT.M appropriately deals with timer resources while exiting.
GTM-7612	-	MUMPS	Better ZALLOC timeout handling
GTM-7616	-	MUPIP	MUPIP RUNDOWN protects against possible inappropriate results from the UNIX IPCS utility
GTM-7617	-	Other Utilities	gtmsecshr wrapper supresses control characters before sending messages to the system log.
GTM-7622	-	DB	No JNLFLUSH error if process holding the crit no longer exists
GTM-7630	-	DB	TRANS2BIG error when a TP transaction tries to update more than 32Ki blocks in a region with MM access method
GTM-7633	-	DB	With NOBEFORE journaling, periodically harden journal records to disk
GTM-7634	-	MUMPS	Improve handling of deeply nested expressions
GTM-7635	-	MUMPS	Processes use an improved strategy to acquire locks when multiple processes compete for the same resource
GTM-7636	-	DB	GT.M handles errors in memory mapped I/O during database file extensions
GTM-7637	-	DB	Shared memory size for MM databases is now a multiple of huge page size (if applicable)
GTM-7639	-	MUPIP	Replication receiver shutdown command on a frozen instance no longer hangs
GTM-7642	-	DB	A TRESTART in an error handler when no transaction is active ($TLEVEL=0) produces a TLVLZERO
GTM-7646	-	MUMPS	Ensure error handling preserves the original error severity

Id	**Prior Id**	**Category**	**Summary**
GTM-7650	-	MUPIP	MUPIP JOURNAL -EXTRACT appropriately handles signals, such as SIGTERM, within an identified time window
GTM-7651	-	DB	Database file extension logic eliminates a window where inappropriate competing extensions might have snuck in
GTM-7652	-	MUPIP	Read-only processes can now perform ONLINE INTEG in the presence of concurrent updates
GTM-7653	-	MUPIP	INTEG -REGION on MM databases works with read-only processes
GTM-7655	-	MUMPS	While opening the syslog, GT.M defers the handling of signals that could lead to a process hang
GTM-7659	-	DB	Attempting to open an MM database file greater than 4GiB in size now issues MMFILETOOLARGE error
GTM-7666	C9B11-001818	MUPIP	MUPIP exit status returned to the shell shows incomplete or failed operation
GTM-7674	-	MUMPS	GT.M appropriately processes an empty call-in table file
GTM-7679	-	Other Utilities	The gtm script now implements -help and -? command line options ✓
GTM-7681	-	Other Utilities	GDE correctly maps global names requesting a range
GTM-7682	-	MUPIP	Handle writes to snapshot file more robustly
GTM-7683	-	MUPIP	Configurable mutex slots ✓
GTM-7685	-	MUMPS	$ZPEEK() intrinsic function
GTM-7689	-	Other Utilities	Avoid an error when sourcing gtmprofile caused by referencing an undefined environement variable
GTM-7692	-	DB	GT.M now issues DBMBMINCFREFIXED when fixing master map integrity error
GTM-7698	-	MUPIP	MUPIP ROLLBACK cleans up Receive Pool shared memory

Id	Prior Id	Category	Summary
GTM-7700	-	Other Utilities	GDE SHOW COMMANDS correction for handling of special DEFAULT cases
GTM-7701	-	DB	Failure to dynamically load encryption plugin at GT.M startup results in CRYPTDLNOOPEN2 error
GTM-7702	-	MUMPS	[Z]GOTO at the M stack level established by an INTRPT preserves the INTRPT context
GTM-7703	-	MUMPS	GT.M issues a FALLINTOFLST error if M code reaches a label with a formallist as a result of a "fall-through" from the previous label
GTM-7706	-	MUPIP	In Instance Freeze environments, argumentless RUNDOWN removes orphaned IPCs
GTM-7707	-	Other Utilities	GTMSECSHR messages to the syslog start with "%GTM"
GTM-7708	-	MUPIP	Replication servers close standard input
GTM-7712	-	DB	Maintain a database consistent state even when there in a failure to extend an MM database
GTM-7713	-	MUMPS	Use %GTM to introduce messages in the system log
GTM-7715	-	MUPIP	Update process reader helper enhancement
GTM-7718	-	MUMPS	HANG decimal rounding correction
GTM-7721	-	MUPIP	Source Server robustly reads journal records from journal files
GTM-7723	-	DB	GT.M issues a REQROLLBACK or REQRECOV error when opening a crashed replication/journaling-enabled database

M-Database Access

- GT.M releases the shared resource required for writing the journal file and committing it to permanent media when the operation encounters an error. Previously, in the very unusual case of such an error, GT.M did not release the shared resource, which caused subsequent journal activity to encounter fatal GTMASSERT errors. [UNIX] (GTM-6892)
- GT.M invokes triggers for globals having null-subscripts. Null string used in range is considered either -infinity or +infinity. Previously, invoking such triggers could produce a segmentation violation (SIG-11). [UNIX] (GTM-6899)
- By improving heuristics used to manage database and journal writes by the Update Process, journal files on replicating secondary instances are now potentially smaller owing to fewer EPOCH and PBLK records. Previously, overly aggressive heuristics resulted in workloads that consisted of bursts of updates interspersed with periods of idleness on a primary instance causing larger than warranted journal files in instances receiving replication streams. [UNIX] (GTM-7054)
- ZPRINT or ZBREAK targeting a trigger definition that falls outside of the namespace of the current global directory issue a TRIGNAMENF error; in that situation $TEXT() returns the empty string. Previously, these actions caused a TPFAIL.[UNIX](GTM-7093)
- GT.M properly closes journaling when it encounters file system out-of-space conditions while flushing journal buffers. Previously, GT.M processes could issue a fatal GTMASSERT error under these conditions. [UNIX] (GTM-7268)
- When GT.M encounters a high number of retries to acquire the resource controlling flushing of journal records from memory to disk it issues a JNLFLUSHNOPROG warning to the syslog facility and continues to retry. Previously it issued JNLFLUSHNOPROG or FSYNCTIMOUT errors and terminated the process with a GTMASSERT. (GTM-7381)
- When a database file with extension disabled (zero extension size) starts to get full, GT.M now logs a warning in the system log. So as not to compromise performance, GT.M checks whenever the master bit map must be updated to show that a local bit map is full, and issues the warning if there are fewer than 512 free blocks, or if the number of free blocks is less than one thirty-second of the number of total blocks. Note: this means that for databases whose size is 512 blocks or less the warning comes at the last successful update before the database is out of room. (GTM-7412) ✓
- An attempt to attach to a database belonging to one node on another node produces a HOSTCONFLICT error which includes the owner node and the node attempting the attach. Note that a crash can sometimes create abandoned ownership. Previously this condition produced a CLSTCONFLICT error, which had less context and less clear terminology, [UNIX] (GTM-7490)
- When a GT.M run-time process or utility encounters an error attaching to a database, it releases control of the shared resources that control initial database access. Previously, if a process remained active after such an error, it could prevent other processes from attaching to the database in question. [UNIX] (GTM-7503)

- For errors in the sample custom errors file, GT.M and its utilities use the correct region to determine whether INST_FREEZE_ON_ERROR applies to a particular error message. Previously some errors which could occur during RECOVER or ROLLBACK set the instance freeze (or decided not to do so) based on the INST_FREEZE_ON_ERROR setting of a different region. If you are concerned about the behavior for other errors, please contact your FIS support representative. [UNIX] (GTM-7548)
- GT.M appropriately writes idle EPOCH records (corresponding to the GVSTATS statistic JRI) for MM access method database files. In V6.0-001, in some cases, it was possible they were not written which meant that in case of a crash shutdown some journal records at the end of the journal file could get permanently lost after a journal recovery/rollback. Note that BG access method was not affected by this. (GTM-7586)
- GT.M allocates the memory used by each process for journal records it generates based on the current active configuration. Previously, starting in V6.0-000 this memory was allocated based on a worst case calculation to allow large global variable nodes; on UNIX, the reduction is approximately 2MiB for each process that accesses a journaled database. (GTM-7595)
- In GT.M databases with the MM mode access method, the first process to open the database does not need write permission for the database file. Previously, a read-only process attempting the first open of an MM database received a signal 11 (SIGSEGV). The workaround was for the first process opening the database to have write access to the database file. [UNIX] (GTM-7599)
- When GT.M finds the process holding the journal flushing resource no long exists, it sends a JNLQIOSALVAGE informational message to the operator log. Previously under this circumstance, it sent a JNLFLUSH message, which was inappropriate because JNLFLUSH, indicating a real problem with journal write, might appear in the custom errors file, thereby inappropriately causing an Instance Freeze on UNIX platforms. (GTM-7622)
- A TP transaction attempting to update more than 32Ki blocks in a region with MM access method issues a TRANS2BIG error. Previously, such an attempt did not detect this boundary, potentially causing database damage. (GTM-7630)
- GT.M periodically explicitly hardens NOBEFORE journal records to disk. Because of an imperfect optimization in V6.0-000 (GTM-7383), GT.M did not perform this explicit hardening under all appropriate circumstances. Note that Durability of neither BEFORE image journal records nor any journal records for regular TP transactions (i.e., those not specifying "BATCH" on TSTART) was affected. [UNIX] (GTM-7633)
- When a GT.M process is unable to expand the memory needed to access an extension of an MM database, it produces DBFILERR/GBLOFLOW errors, and if the process attempts any further access to that database, it produces MMREGNOACCESS errors. Previously, subsequent attempts to access the inaccessible database produced a fatal SEG-V (SIG-11) error. [UNIX] (GTM-7636)
- When a region has the MM access method and hugepages are configured, GT.M creates shared memory that is a multiple of hugepage size (typically 2MB or 4MB). Previously, in hugepages environment, shared memory for MM databases were not a multiple of hugepage size and caused standalone operations to error out. [LINUX](GTM-7637)
- A TRESTART in an error handler when no transaction is active ($TLEVEL=0) produces a TLVLZERO. In V6.0-001, this situation produced an inappropriate TPRESTNESTERR. (GTM-7642)

- GT.M processes not operating in an Instance Freeze environment handle ENOSPC conditions during file extensions appropriately. In addition, during file extensions, GT.M processes no longer issue WAITDSKSPACE messages to stderr. Previously when not operating in an Instance Freeze environment, GT.M when handling ENOSPC conditions during file extensions, inappropriately allowed multiple processes to concurrently extend the same region, which, in very rare cases, could cause database damage. [UNIX] (GTM-7651)
- On 32-bit platforms, GT.M issues an MMFILETOOLARGE error when an attempt to open a database file with the MM access method fails because of addressability limits. Previously, GT.M memory mapped only a part of the file. Subsequent global variable read and write attempts issued GVxxxFAIL errors, resulting in indefinite file extensions and database damage. [UNIX] (GTM-7659)
- GT.M issues DBMBMINCFREFIXED warning to the operator log whenever it notices a master bitmap incorrectly marking a local bitmap as incorrectly free and auto corrects this by marking the local bitmap as full in the master bitmap. Previously, GT.M issued no such messages thereby leaving no record of such an unusual event. [UNIX] (GTM-7692)
- When GT.M fails to dynamically load the encryption plugin and gtm_passwd is set to an empty string, it issues a CRYPTDLNOOPEN2 error. In V6.0-001, for this combination of circumstances, GT.M terminated with either a SEG-V (SIG-11) or SIGILL (SIG-4). [UNIX] (GTM-7701)
- When GT.M encounters a failure to extend an MM database, it leaves the database in a consistent state, although inaccessible to the process. Previously, such a failure left the database with a DBFGTBC integrity error. Also, during file extension in BG and MM, GT.M utilities check and, if appropriate correct, the accuracy of the free blocks counter. Previously, the check was done only by GT.M [UNIX] (GTM-7712)
- When opening a database that was not cleanly shut down, GT.M 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 you 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. [OpenVMS] (GTM-7723)

M-Other Than Database Access

- When an external routine specifies a non-positive time to the gtm_start_timer() function, GT.M schedules the timer for 10 ms. Previously, GT.M issued a GTMASSERT whenever the expiration time specified in gtm_start_timer() call was non-positive. [UNIX] (GTM-1937)

- GT.M provides an interface intended for use with Java---please refer to the Java Interface Technical Bulletin. The work to add this feature also addressed a number of issues with the C-oriented call-in/out interface. GT.M preserves precision of a double call-in or call-out argument at 15 decimal digits and a float argument at six decimal digits overall when converting to a GT.M internal representation, matching the persistent decimal precision of doubles and floats, respectively, in the IEEE 754 format. Previously, GT.M always reserved six digits after the decimal point. Additionally, GT.M correctly interprets long, long *, unsigned long, and unsigned long * arguments in their full 8-byte capacity when making calls out from M to C and calling-in from C to M. Previously, on 64-bit platforms, internal representations whose values exceeded 4-bytes did not correctly translate into 8-byte long integers; and, in both 32- and 64-bit editions, casting of signed and unsigned long arguments passed by value was reversed in case of call-outs. Also, returning a non-zero value from a C routine with gtm_status_t return type raises an error status. Previously, GT.M sent the error message text to $PRINCIPAL. [UNIX] (GTM-3220) ✓

- Indirection gives a syntax error when passed the string "@". Previously, a command such as SET @"@"=1 would neither have any effect nor give a syntax error. (GTM-4324)

- Indirection appropriately issues a syntax error when a string contains either control characters or extra characters. Previously, indirection would ignore the tail of such a string. (GTM-5284)

- For applications whose source code includes literals, GT.M provides options to help optimize performance and virtual memory usage.

  - When a routine is compiled with the -DYNAMIC_LITERALS qualifier, GT.M computes certain data structures associated with literals used in that routine, and places these structures in the object code. For object code in shared libraries, these data structures are dynamically loaded and unloaded, reducing the amount of private memory required by each process using that routine, which in turn can increase application scalability by allowing more processes to execute with the same memory, or increase application performance by making more memory available for file system buffers. Without this option, these data structures continue to be generated when a routine is linked, and stay in the private memory of each process. As use of the option increases object code size, and as the dynamic loading and unloading only saves memory when the object code is in shared libraries, FIS recommends restricting use of the qualifier only when compiling object code to be loaded into shared libraries.

  - When a routine is compiled with the -NOINLINE_LITERALS qualifier, GT.M generates code to invoke routines that load literals, instead of generating in-line code, reducing routine sizes. Use of -NOINLINE_LITERALS may help counteract growth in object size due to -DYNAMIC_LITERALS.

As the scalability and performance from reduced per-process memory usage may or may not compensate for the incremental cost of dynamically loading and unloading the data structures, and as the performance of routines vs. inline code can be affected by the availability of routines in cache, FIS suggests benchmarking to determine the combination of qualifiers best suited to each workload. Note that applications can freely mix routines compiled with different combinations of qualifiers. In addition, MUMPS compile invocation mode and the ZCOMPILE command inherit default qualifiers from the environment variable gtmcompile. Previously, gtmcompile was only effective within the GT.M run-time environment, for explicit ZLINKs and auto-ZLINKs. [UNIX] (GTM-6330) ✓

- The JOB command now creates .mje and .mjo files only in the working directory of the JOB'd process. Previously, when the DEFAULT process parameter of JOB command specified a directory which was different than the current working directory of the process executing JOB command and VIEW "JOBPID":1 was in effect, GT.M inappropriately also created .mje and .mjo files in the working directory of the process executing the command. [UNIX] (GTM-6548)

- GT.M reports $ZREALSTOR, $ZALLOCSTOR, and $ZUSEDSTOR ISVs with 18 digits of precision. Previously, GT.M returned inappropriate results for the aforementioned ISVs if the values exceeded 4,294,967,295. (GTM-6717)

- GT.M processes no longer hang or terminate abnormally if interrupted by a signal within a specific time window. Previously, a GT.M process could hang or terminate abnormally if it encountered a signal interrupt within one of the certain unprotected, very short time windows. [UNIX] (GTM-6802)

- A sequential disk device accepts the [NO]FOLLOW deviceparameters for OPEN and USE. In FOLLOW mode, a READ returns only when it has a complete record or reaches any specified timeout; it waits for more input rather than terminating for an EOF (end-of-file) condition. Each argument to a READ command is considered a separate READ in the following discussion.

  For STREAM or VARIABLE record format files, a READ returns when it encounters an EOL, or has read #length characters if specified, or WIDTH characters if #length is not specified, whichever occurs first.

  For non Unicode FIXED record format files, a READ returns when it has read #length characters, or WIDTH characters if #length is not specified. For Unicode FIXED record format files, each READ obtains characters from a single RECORDSIZE bytes long record, and an untimed READ that needs a new record does not return until RECORDSIZE bytes are available.

  The USE command can switch a device from NOFOLLOW to FOLLOW or from FOLLOW to NOFOLLOW. This provides a READ mode of operation similar to a tail -f in UNIX. [UNIX] (GTM-7540) ✓

- M-tracing correctly reports elapsed time per each executed line and label of M code. Previously, M-tracing reports could overestimate the elapsed time for certain lines (and labels, as a result) by an unusually large amount. (GTM-7588)

- GT.M correctly parses parentheses around @x@() indirection. Previously, due to enhancements made in V6.0-000 for GTM-5896, commands with indirect arguments encased in parentheses such as COMMANDWORD (@x@()) or LOCK +(@x@()) gave INDEXTRACHARS errors at runtime. (GTM-7609)

- When exiting, GT.M appropriately deals with internal timer resources. Previously, under very rare circumstances a terminating GT.M process could signal a GTMASSERT due to an unexpected timer start. [UNIX] (GTM-7611)
- The ZALLOC command processes timeouts by resuming with the remaining time after a MUPIP INTRPT. Previously it resumed using a random time which could cause it to wait significantly less or more than the specified time. (GTM-7612)
- GT.M compiles expressions with pathologically deep (hundreds) levels of nesting. Previously, compiling such expressions resulted in a segmentation fault. (GTM-7634)
- Processes use a lightweight strategy in acquiring a fine granularity resource lock used for flushing blocks from the global buffers to secondary storage. Previously, it was possible for a process to unnecessarily wait for the resource when multiple processes were competing for it. While we are not aware of this having been an issue for customers, we identified this improvement from testing that showed secondary helper processes interfering with the update process. [UNIX] (GTM-7635)
- GT.M properly handles issuing additional messages during error handling. Previously, in rare cases an INFO-level message could be issued while handling an ERROR or FATAL-level message, causing GT.M to proceed as if the original message were INFO-level. [UNIX] (GTM-7646)
- While opening the syslog, GT.M defers the handling of signals (such as SIGTERM) that could lead to a process hang. Previously, there was a very small chance that a signal could interrupt a GT.M process in the middle of opening the syslog, thus causing the process to hang indefinitely. [UNIX] (GTM-7655)
- GT.M appropriately processes an empty call-in table file. Previously, specifying an empty call-in table file could cause a segmentation violation (SIG-11) during an invocation of an M routine from C. [UNIX] (GTM-7674)
- The $ZPEEK() intrinsic function provides a way to examine memory in the current process address space. It is intended as a tool to make it more convenient for FIS to access information in the address space of processes more efficiently than by calling out to external functions. It is documented here for completeness. While FIS normally maintains stability of GT.M functionality from release to release, this function is not designed for non-FIS usage, and FIS may change or eliminate this function at any time.

  The $ZPEEK() function returns the contents of the memory requested as a string depending on the requested (or defaulted) formatting.

  The format of the $ZPEEK() function is:

  ```
  $ZPEEK("mnemonic[:argument",offset,length[,format])
  ```

  - **mnemonic** specifies the memory area $ZPEEK() is to access. Some mnemonics have arguments separated from the mnemonic by a colon (":"). The mnemonics are case independent. Possible mnemonics, their possible abbreviations and their arguments are:

  - *CSA[REG]* - returns a value from the sgmnt_addrs (process private) control block. Takes a case independent region name as an argument.

  - *FH[REG]* - returns a value from the sgmnt_data (shared file header) control block. Takes a case independent region name as an argument.

  - *GDR[REG]* - returns a value from the gd_region (process private) control block. Takes a case independent region name as an argument.

  - *GLF[REPL]* - returns a value from the jnlpool.gtmsrc_lcl_array[n] control block. Takes a numeric index (n) as an argument.

  - *GRL[REPL]* - returns a value from the recvpool.gtmrecv_local control block. No argument allowed. Only available when run on a non-primary instance.

  - *GSL[REPL]* - returns a value from the jnlpool.gtmsource_local_array[n] control block. Takes a numeric index (n) as an argument.

  - *JPC[REPL]* - returns a value from the jnlpool.jnlpool_ctl control block. No argument allowed.

  - *NL[REG]* - returns a value from the node_local (shared) control block. Takes a case independent region name as an argument.

  - *NLREPL* - returns a value from the node_local (shared) control block associated with replication. No argument allowed.

  - *PEEK* - returns a value based on the supplied argument. Argument is the base address of whatever is being fetched in 0xhhhhhhh format where the h's are hex digits.

  - *RIH[REPL]* - returns a value from the jnlpool.repl_inst_filehdr control block. No argument allowed.

  - *RPC[REPL]* - returns a value from the recvpool.recvpool_ctl control block. No argument allowed. Only available when run on a non-primary instance.

  - *UHC[REPL]* - returns a value from the recvpool.upd_helper_ctl control block. No argument allowed. Only available when run on a non-primary instance.

  - *UPL[REPL]* - returns a value from the recvpool.upd_proc_local control block. No argument allowed. Only available when run on a non-primary instance.

- **offset** (first integer expression) is a numeric value that specifies the offset from the address supplied or implied by the the mnemonic and argument. Specifying a negative offset results in a BADZPEEKARG error. Specifying too large an offset such that unavailable memory is specified results in a BADZPEEKRANGE error.

- **length** (second integer expression) is a numeric value that specifies the length of the field to be fetched. Specifying a negative legnth results in a BADZPEEKARG error. Specifying a length that exceeds the maximum string length results in a MAXSTRLEN error. Specifying too large a length such that unavailable memory is specified results in a BADZPEEKRANGE error.

- **format** is an optional single case independent character formatting code for the retrieved data. The formatting codes are:
  - *C* : returns a character representations of the memory locations; this is the DEFAULT if the fourth argument is not specified.
  - *I* : returns a signed integer value - negative values have a preceding minus sign (-); the length can be 1, 2, 4, or 8 bytes.
  - *U* : returns an unsigned integer value - all bits are part of the numeric value; the length can be 1, 2, 4, or 8 bytes.
  - *S* : returns a character representation of the memory locations and the first NULL character found terminates the returned string; that is: the specified length is a maximum.
  - *X* : returns a hexadecimal value as 0xXXXXXX where XXXXXX is twice the specified length in bytes, so requested length 1 returns 0xXX and length 4 returns 0xXXXXXXXX; the length can be 1, 2, 4, or 8 bytes.
  - *Z* : returns a hexadecimal representation of the memory locations as 'X' does, without regard to endianness, and with no length restriction other than max string length.

Notes

- $ZPEEK() has no UTF-8 checking. It is possible for values returned by the 'C' and 'S' codes to have invalid UTF-8 values in them. Take care when processing values obtained by these codes to either use "VIEW NOBADCHAR" when dealing with such values and/or use the $Zxxx() flavors of functions like $ZPIECE(), $ZEXTRACT(),etc which also do not raise BADCHAR errors when encountering invalid UTF-8 encoded strings.
- Note that $ZPEEK() with 8 byte numeric formatting can return numeric string values that exceed GT.M's current limit of 18 digits of precision. If the values are used as strings, the extra digits are preserved, but if used arithmetically, the lower precision digits can be lost.
- When values from replication structures are requested and the structures are not available due to replication not running or, in the case of the gtmrecv.* control block base options, if not running on a non-primary instance where the gtmrecv.* control are available, a ZPEEKNOREPLINFO error is raised.

(GTM-7685)

- ZGOTO level:entryref within INTRPT code where level specifies the base level of the INTRPT stays within the INTRPT context and the resulting M stack frame shows up in ZSHOW "S[TACK]" output.

Previously this odd action created a state that incompletely removed the INTRPT context and created an M stack frame that ZSHOW "STACK" hid. (GTM-7702)

- GT.M issues a FALLINTOFLST error if M code reaches a label with a formallist as a result of a "fall-through" from the previous label. Prior to V5-5.000 GT.M issued an ACTLSTEXP error in such situations, but in the subsequent releases falling through to a label with a formallist could erroneously trigger an ACTLSTTOOLONG error, leading to abnormal process termination in certain situations. (GTM-7703)
- GT.M uses %GTM to introduce messages in the syslog. Starting in V54001 GT.M messages started with two percent signs (%%GTM). [UNIX](GTM-7713)
- For T greater than or equal to .001, "HANG T" suspends execution for no less than the T seconds. As in previous versions, "HANG T" where T is less than .001 is treated as equivalent to "HANG 0". Previously, if the argument to a HANG command contained more than three decimal places, the command would round down to the nearest second. (GTM-7718)

Utilities-MUPIP

- MUPIP RUNDOWN issues a MUUSERLBK error on a previously crashed replication-enabled database with BEFORE IMAGE journaling, and MUUSERECOV error in case of a non-replicated or NOBEFORE-journaled database. To bypass the errors and proceed with the rundown, MUPIP RUNDOWN requires an OVERRIDE qualifier. Previously, MUPIP RUNDOWN would perform a rundown regardless of the replication and journaling state of the database, thus complicating data recovery from a journal file or another replicated instance. [UNIX] (GTM-7420) ✓

- MUFILRNDWNFL, MUJPOOLRNDWNFL, MURPOOLRNDWNFL and MUFILRNDWNFL2 messages have the error (E) type. Previously, these messages had the informational (I) type; there is no change in behavior - only in the displayed message. (GTM-7447)

- When MUPIP encounters problems getting standalone access to a database, in addition to MUSTANDALONE or SYSCALL errors, it may report an FTOKKEY message, a SEMID message, or additional SYSTEM-E-ENO* messages, depending on the particular circumstances. Previously, MUPIP did not provide this error detail. [UNIX](GTM-7518)

- MUPIP RUNDOWN does not issue an error if another process concurrently removes a shared memory identifier RUNDOWN has identified for removal. Previously in such a situation, MUPIP RUNDOWN issued an error and created a state that caused subsequent startups to issue one of the following errors: REQRECOV, REQROLLBACK or REQRUNDOWN. [UNIX] (GTM-7568)

- MUPIP REORG and MUPIP SIZE SCAN work safely with MM databases. Previously, due to a regression introduced in V6.0-000, in the presence of concurrent file extensions, running either utility could have resulted in a SIG-11. In addition, DSE DUMP -FILEHEADER reports a reduced number of Non-TP retries. Previously, unnecessary retries internal to MUPIP SIZE SCAN could have inflated the number displayed by DSE. Also, MUPIP SIZE SCAN gives a more reliable approximation for BG databases. Previously, a rare concurrent event could have distorted the results. [UNIX] (GTM-7592)

- MUPIP RUNDOWN protects against possible inappropriate results from the UNIX ipcs utility. In its argumentless form, MUPIP RUNDOWN captures output from the UNIX ipcs utility in order to locate GT.M IPC resources. Previously, a buffer not protected against possible overflow from a subverted ipcs could potentially cause the MUPIP RUNDOWN process to misbehave. Note that any misbehavior would be limited by the access permissions of files and shared memory segments accessible to the userid and groups of the MUPIP process. Also, there was no misbehavior from any normal ipcs output. [UNIX] (GTM-7616)

- MUPIP REPLIC -FREEZE=OFF, on a frozen instance, works when a Receiver Shutdown command is already in progress. Previously, attempting to release an instance freeze when a Receiver Shutdown command was already in progress resulted in a deadlock. The work around was to identify the shutdown command (with the string "-receiver -shutdown" in the ps process listing) and issue a MUPIP STOP on that process. Reissuing the shutdown command after this took care of completing the previously interrupted shutdown command. [UNIX] (GTM-7639)

- MUPIP JOURNAL -EXTRACT appropriately handles signals, such as SIGTERM, within an identified time window. Previously, MUPIP journal extract could terminate abnormally if it encountered a signal interrupt within a certain, very short time window. [UNIX] (GTM-7650)
- An ONLINE INTEG started by a process having read-only access to database files creates snapshot files with ownership and permissions identical to those it would use for shared memory if it were the first process to open that database file. Previously, such an ONLINE INTEG process created snapshot files without providing write permissions to other processes updating the database; this eventually caused the INTEG to fail with REGSSFAIL error. [UNIX] (GTM-7652)
- MUPIP INTEG -REGION on MM databases works even if the process only has read permissions on the database. Previously, such a MUPIP INTEG -REG issued DBRDONLY error and continued to the next region. (GTM-7653)
- MUPIP now properly sets exit status on error. Previously, exit status could be zero (0) after errors encountered during database rundown. [UNIX] (GTM-7666)
- GT.M better maintains the snapshot file used for MUPIP INTEG -ONLINE. Previously, an odd condition could result in an EFAULT error and prevent the INTEG from completing. [UNIX] (GTM-7682)
- MUPIP SET -FILE -MUTEX_SLOTS=integer maintains the size of a structure GT.M uses to manage contention for the principal critical section for a database. The minimum value is 1024, which is also the default; the maximum value is 32768. This MUPIP SET option requires standalone access. DSE DUMP -FILEHEADER output shows the current value for this field with the label: "Mutex Queue Slots". When there are many processes contending for database access, if this structure cannot accommodate all the waiting processes, GT.M performance can become erratic and poor. Therefore, FIS recommends setting this value to a minimum of slightly more than the maximum number of concurrent processes you expect to access the database. Previously, GT.M fixed the size of this structure at 1024 slots, which was adequate for most environments; DSE DUMP -FILEHEADER formerly displayed a field "Mutex Spin Sleep Time" which is now obsolete and replaced by "Mutex Queue Slots". (GTM-7683) ✓
- MUPIP JOURNAL -ROLLBACK appropriately invalidates the shared memory identifier (shmid) of the Receive Pool stored in the replication instance file. Previously, in environments configured for Instance Freeze, MUPIP JOURNAL -ROLLBACK did not appropriately invalidate this stored identifier, causing subsequent Receiver Server restarts to terminate with a GTMASSERT. The work around was to issue a MUPIP RUNDOWN -REG "*" before restarting the Receiver Server. [UNIX](GTM-7698)
- In environments configured for Instance Freeze, argumentless MUPIP RUNDOWN now removes IPC resources associated with Journal Pool on the Originating and/or the Replicating instance. Previously, in Instance Freeze environments, such IPC resources were not removed by argumentless MUPIP RUNDOWN. [UNIX] (GTM-7706)
- In order to not unnecessarily tie up a device which they do not use, the replication Source and Receiver Servers ensure their standard input device is assigned to /dev/null. Previously, they retained whatever device with which they were started. [UNIX](GTM-7708)
- An update process reader helper moves on to the next block when it finds the data block it is attempting to pre-read is already being read by some other process. In limited benchmarking this

seems to make the update reader helpers more effective when a replicating instance is processing a high level of updates. Previously, an update reader helpers waited like other processes for any block it was trying to access to complete a concurrent read. (GTM-7715)

- The Source Server robustly reads journal records from the journal files. Previously, in certain rare cases involving either a ROLLBACK (on a replicating instance) or a replication setup consisting of one or more tertiary instances, the Source Server terminated with GTMASSERT. [UNIX] (GTM-7721)

Utilities-Other Than MUPIP

- LKE SHOW/CLEAR -LOCK qualifier now accepts concatenations of $CHAR()/$ZCHAR() and graphic characters for resource argument subscripts, rejects incorrect subscript termination and empty subscript parenthesis, works properly when a comma or double-quote appears within a subscript string. Previously, any use of the concatenation operator would issue an error, $CHAR()/ $ZCHAR() representation required a case sensitive $C()/$ZCH(), subscripts could be empty parenthesis or list of subscripts could be terminated with a comma, and using a double-quote or comma inside a subscript string gave unintended results.(GTM-6181)

- DSE does not attach to a region after a VERMISMATCH error. Previously, VERMISMATCH was an informational message for DSE. This exception allowed DSE to attach the shared memory associated with a region having a different GTM version. While this behavior was consistent with the DSE design goal of being able to adjust anything, it ultimately seemed too dangerous for any benefit it might provide. (GTM-7545)

- When reading DSE output, %DSEWRAP ignores duplicated file header output. It returns only the last duplicated file header information. Additionally, %DSEWRAP ignores GT.CM regions. To provide appropriate input for %DSEWRAP for a GT.CM region, the operator must do a file header dump on the computer where the GT.CM hosted database file resides.(GTM-7596)

- The gtmsecshr wrapper program replaces control characters with asterisks (*) before sending messages to syslog. Previously control characters in $gtm_dist were sent unaltered to syslog, and if the syslog service did not suppress control characters, the log file's appearance could be altered by embedded control characters such as newlines, carriage returns and vertical tabs. [UNIX] (GTM-7617)

- The gtm script now implements -help and -? command line options to assist new users. Previously using either option resulted in a GTM-E-CLIERR error message. [UNIX] (GTM-7679) ✓

- GDE correctly maps global names to corresponding regions. Previously, in some odd cases where name specification ended in "*" to correctly specify a range, GDE VERIFY failed causing GDE not to exit; and in other cases GDE exited, writing a new global directory which gave an INPINTEG error when read; even when they worked properly such cases sometimes showed maps that could be reduced to simpler form. (GTM-7681)

- When the environment variable gtm_repl_instance is set, but the environment variable gtm_dist is not set, sourcing the gtmprofile file does not generate an error and leaves gtm_repl_instance with an appropriate value. Previously, sourcing gtmprofile with this combination of gtm_repl_instance and gtm_dist elicited the complaint "sed: -e expression #1, char 0: no previous regular expression" and cleared the existing value of gtm_repl_instance. [UNIX] (GTM-7689)

- GDE SHOW COMMANDS generates proper output if the * namespace is mapped to a region other than DEFAULT ($DEFAULT in VMS) OR the DEFAULT segment ($DEFAULT in VMS) is renamed. Previously if the GDE SHOW COMMANDS output for the above cases were fed to a fresh GDE session, it would fail verification and issue a OBJNOTFND error. (GTM-7700)

- GTMSECSHR messages to the syslog start with "%GTM". [UNIX] (GTM-7707)

Error and Other Messages

BADZPEEKARG

BADZPEEKARG, Missing, invalid or surplus xxxx parameter for $ZPEEK()

Runtime error: One of the parameters specified to $ZPEEK() is incorrect. Possible values of xxxx:

- **mnemonic type** - The mnemonic in the first argument is unknown.
- **mnemonic argument** - An argument for the given mnemonic (specified after a ":" character) is expected and missing, is present and unexpected, or not in its proper form.
- **mnemonic argument (region name)** - Expected a region name argument which is either missing or not available.
- **mnemonic argument (array index)** - Expected a numeric array index argument which is either missing or out of range.
- **mnemonic argument (peek base address)** - Expected an address in the form 0xHHHHHHHH.. which is either missing or invalid.
- **offset** - Expected non-negative numeric value which is either missing or invalid.
- **length** - Expected non-negative numeric value which is either missing or invalid.
- **format** - Expected a single character format code which is either missing or invalid.

Action: Review the invocation and correct the defective parameter.

BADZPEEKFMT

BADZPEEKFMT, $ZPEEK() value length inappropriate for selected format

Runtime error: The format code specified is not valid for the length specified. For example, format code 'I' works with 1, 2, 4, and 8 byte fields. A field length of 3 would raise this error.

Action: Review the invocation and correct the defective length and/or format character.

BADZPEEKRANGE

BADZPEEKRANGE, Access exception raised in memory range given to $ZPEEK()

Runtime error: Some combination of base address, offset, length and/or alignment caused GT.M to raise a memory access exception when fetching the requested value.

Action: Review the invocation and correct the defective parameter.

DBBADFREEBLKCTR ●

DBBADFREEBLKCTR, Database xxxx free blocks counter in file header: oooo appears incorrect; should be nnnn. Auto-corrected.

Run Time Warning: This indicates that during a file extension, because they differed, GT.M adjusted the free blocks counter (oooo) in the file header to agree with free blocks indicated by the master map (nnnn). Because this may indicate a master bitmap integrity error (DBMBPINCFL), check the next MUPIP INTEG carefully.

Action: Run MUPIP INTEG; if it reports a DBMBPINCFL integrity error, use DSE to correct it, and to increase the file header free blocks counter by the amount GT.M reduced it in the DBBADFREEBLKCTR message. Run an additional INTEG to confirm the corrections.

DBMBMINCFREFIXED

DBMBMINCFREFIXED, Master bitmap incorrectly marks local bitmap 0xAAAA as free. Auto-corrected

Runtime warning: The above error is issued when the runtime engine detects an integrity error with the master map that indicates that the local bitmap 0xAAAA is free when in actually it is not. The error is also auto-corrected by the runtime engine by marking the local bitmap as full in the master bitmap.

Action: This error is entirely benign, but because it should not occur, be sure to check your next MUPIP INTEG output thoroughly and also check your operator logs prior to this warning for other unusual events.

FALLINTOFLST

FALLINTOFLST, Fall-through to a label with formallist is not allowed

Runtime error: This error indicates that M code reached a label with a formallist by falling through from the previous label.

Action: Revisit your code to ensure that all invocations of labels with a formallist occur using a DO command or extrinsic function ($$).

FREEBLKSLOW

FREEBLKSLOW, Only bbbb free blocks left out of tttt total blocks for ffff>

Run Time warning message: This message warns that database file ffff with automatic file extensions disabled has only bbbb blocks left out of a total allocation of tttt.

Action: Use MUPIP EXTEND to extend the file, or MUPIP SET to enable automatic extensions, or reduce the amount of data in the file.

FTOKKEY

FTOKKEY, FTOK key 0xnnnnn

MUPIP informational message: This message reports additional information for an associated error which had trouble with the FTOK key 0xnnnnn.

Action: Check "ipcs -s" for the given key, and see the associated error.

GTMSECSHRDMNSTARTED

GTMSECSHRDMNSTARTED, gtmsecshr daemon started (key: 0xhhhh) for version vvvv from dddd

gtmsecshr informational message: This message indicates that GTMSECSHR process has been started with an IPC key of 0xhhhh (in hexadecimal notation) for GT.M version vvvv from directory dddd.

GTMSECSHRGETSEMFAIL

GTMSECSHRGETSEMFAIL, error getting semaphore errno = xxxx

gtmsecshr error: This error indicates that GTMSECSHR failed to obtain a semaphore set identifier for a specific IPC key during process termination, and that the error code returned by semget() is xxxx.

Action: The IPC resources that GTMSECSHR uses should be unique to GT.M, and this message indicates an unexpected condition. Investigate whether other software is using IPC resources in a way that conflicts with GT.M. If you can't find an explanation, report the entire incident context to your GT.M support channel.

GTMSECSHRREMFILE

GTMSECSHRREMFILE, [client pid pppp] File (ffff) removed

gtmsecshr informational message: This message indicates that GTMSECSHR removed file ffff on behalf of process pppp.

GTMSECSHRREMSEM

GTMSECSHRREMSEM, [client pid pppp] Semaphore (ssss) removed

gtmsecshr error: This message indicates that GTMSECSHR removed a semaphore identified by the key ssss on behalf of process pppp.

Action: This is benign. No action necessary.

GTMSECSHRREMSEMFAIL

GTMSECSHRREMSEMFAIL, error removing semaphore errno = xxxx gtmsecshr error: This error indicates that GTMSECSHR failed to remove a semaphore set identified by a specific IPC key during process termination, and that the error code returned by semctl() is xxxx.

Action: The IPC resources that GTMSECSHR uses should be unique to GT.M, and this message indicates an unexpected condition. Investigate whether other software is using IPC resources in a way that conflicts with GT.M. If you can't find an explanation, report the entire incident context to your GT.M support channel.

GTMSECSHRREMSHM

GTMSECSHRREMSHM, [client pid pppp] Shared memory segment (ssss) removed, nattch = nnnn

gtmsecshr informational message: This message indicates that GTMSECSHR removed a shared memory segment identified by the key ssss on behalf of process pppp, and that there were nnnn processes attached to that segment.

GTMSECSHRSEMGET

GTMSECSHRSEMGET, semget error errno = xxxx

gtmsecshr error: This error indicates that GTMSECSHR process failed to obtain a semaphore set identifier for a specific IPC key, and that the error code returned by semget() is xxxx.

Action: Consult your system administrator to ensure semaphores are appropriately configured.

GTMSECSHRSHMCONCPROC

GTMSECSHRSHMCONCPROC, More than one process attached to Shared memory segment (ssss) not removed (nnnn)

gtmsecshr error: This error indicates that the shared memory segment identified by the key ssss has not been removed because nnnn processes are currently attached to it.

Action: The IPC resources that GTMSECSHR uses should be unique to GT.M, and this message indicates an unexpected condition. Investigate whether other software is using IPC resources in a way that conflicts with GT.M. If you can't find an explanation, report the entire incident context to your GT.M support channel.

GTMSECSHRUPDDBHDR

GTMSECSHRUPDDBHDR, [client pid pppp] database fileheader (dddd) updated iiii

gtmsecshr informational message: This message indicates that GTMSECSHR updated database fileheader dddd on behalf of process pppp for the purpose of iiii.

HOSTCONFLICT

HOSTCONFLICT, Host hhhh could not open database file dddd because it is marked as already open on node nnnn

Runtime error: The database file (dddd) has already been opened by a host (nnnn) other than the local host (hhhh).

Action: Ensure that host nnnn has closed dddd. Make sure both host names are correct. Changing host names in the middle of a database access can cause this error.

INDRCOMPFAIL ●

INDRCOMPFAIL, Compilation of indirection failed

Run Time Error: This indicates that an indirection or XECUTE command failed due to syntax errors.

Action: Review the the code and make sure the indirection or XECUTE string has valid syntax and contains no non-graphic characters. Consider using $ZWRITE to identify any such characters.

JNI

JNI, xxxx

Run Time Error: GT.M uses this message with appropriate accompanying text, xxxx, to indicate an error condition with a Java call-out invocation.

Action: Examine the text and address the described error condition.

JNLQIOSALVAGE

JNLQIOSALVAGE, Journal IO lock salvaged

Run time informational message: An active process salvaged a critical resource marked as belonging to a dead process during a journal flush.

Action: The system automatically returns the critical resource to normal operation and continues execution. If this message continues to occur, please investigate why the process holding the crit abnormally exited.

MALLOCMAXUNIX ●

Exceeded maximum allocation defined by $gtm_max_storalloc.

Runtime error: This error accompanies a MEMORY error as a secondary error to indicate that the limit the process hit was not an OS limit but one artificially defined by the $gtm_max_storalloc environment variable.

Action: Increase the value of, or unset, $gtm_max_storalloc, or identify the source of the memory consumption (for example, creating and keeping lots of local variables) and reduce it.

MALLOCMAXVMS ●

Exceeded maximum allocation defined by GTM_MAX_STORALLOC.

Runtime error: This error accompanies a MEMORY error as a secondary error to indicate that the limit the process hit was not an OS limit but one artificially defined by the GTM_MAX_STORALLOC logical.

Action: Increase the value of or unset the GTM_MAX_STORALLOC logical or identify the source of the memory consumption (for example, creating and keeping lots of local variables) and reduce it.

MMFILETOOLARGE ●

Size of rrrr region (ffff) is larger than maximum size supported for memory mapped I/O on this platform.

Runtime error: GT.M and its Utility programs issue this to indicate an attempt to open the database ffff corresponding to region rrrr when the size of the database file is greater than the maximum size supported for memory mapped I/O.

Action: Consider as appropriate: migrating to a platform not having this limitation, using more, but smaller, regions, or using the BG access method.

MMREGNOACCESS

MMREGNOACCESS, Region rrrr (ffff) is no longer accessible. See prior error messages in the operator and application error logs

Runtime error: Issued when a process attempts to access region rrrr corresponding to database file ffff (opened with the MM access method) which previously became inaccessible to this process due to failure during file extension.

Action: Review the operator log for DBFILERR messages and application error logs for GBLOFLOW status to diagnose the circumstances for the earlier failure of memory mapped I/O operations and take corrective action.

MUUSERECOV

MUUSERECOV, Abnormal shutdown of journaled database dddd detected

Runtime error: This error is issued when attempting a MUPIP RUNDOWN on a previously crashed journaling-enabled database dddd.

Action: Use MUPIP RECOVER to restore the normal state of the database.

MUUSERLBK

MUUSERLBK, Abnormal shutdown of replication-enabled database dddd detected

Runtime error: This error is issued when attempting a MUPIP RUNDOWN on a previously crashed replication-enabled (with BEFORE IMAGE journaling) database dddd.

Action: Use MUPIP ROLLBACK to restore the normal state of the database.

NULLENTRYREF

NULLENTRYREF, JOB command did not specify entryref

Runtime error: This error is issued when mandatory entryref is not specified with JOB command.

Action: Specify the entryref for JOB command.

SECSHRCHDIRFAILED

SECSHRCHDIRFAILED, gtmsecshr unable to chdir to its temporary directory (dddd)

Runtime error: This error indicates that GTMSECSHR process failed to change the current working directory to dddd.

Action: Verify that the environment provides the desired dddd, that dddd exists, and that it is a directory.

SECSHRCLEARENVFAILED

SECSHRCLEARENVFAILED, clearenv failed. gtmsecshr will not be started

Runtime error: This error indicates that GTMSECSHR process failed to clear its environment of all name-value pairs.

Action: Verify that no other processes have corrupted the environment via non-standard means.

SECSHREXECLFAILED

SECSHREXECLFAILED, execl of ffff failed

Runtime error: This error indicates that GTMSECSHR process failed to update its process image with file ffff.

Action: Ensure that GTMSECSHR executables and their parent directories have correct permissions.

SECSHRGTMDBGLVL2LONG

SECSHRGTMDBGLVL2LONG, gtmdbglvl env var too long. gtmsecshr will not be started

Runtime error: This error indicates that the value of the gtmdbglvl environment variable is too long.

Action: Verify that gtmdbglvl contains a proper integer in the decimal or hexadecimal format.

SECSHRGTMDIST2LONG

SECSHRGTMDIST2LONG, gtm_dist env var too long. gtmsecshr will not be started

Runtime error: This error indicates that the value of the gtm_dist environment variable is too long.

Action: Verify that gtm_dist contains a proper path to GT.M installation directory.

SECSHRGTMTMP2LONG

SECSHRGTMTMP2LONG, gtm_tmp env var too long. gtmsecshr will not be started

Runtime error: This error indicates that the value of the gtm_tmp environment variable is too long.

Action: Verify that gtm_tmp contains a proper path to the directory for GT.M/GTMSECSHR socket communication.

SECSHRNOGTMDIST

SECSHRNOGTMDIST, gtm_dist env var does not exist. gtmsecshr will not be started

Runtime error: This error indicates that the gtm_dist environment variable is not set.

Action: Ensure that gtm_dist is set and points to the GT.M installation directory.

SECSHRNOTOWNEDBYROOT

SECSHRNOTOWNEDBYROOT, dddd not owned by root. gtmsecshr will not be started

Runtime error: This error indicates that dddd is not owned by root.

Action: Ensure that GTMSECSHR executables and their parent directories have correct ownership and permissions.

SECSHRNOTSETUID

SECSHRNOTSETUID, ffff not set-uid. gtmsecshr will not be started

Runtime error: This error indicates that ffff does not have the set-uid bit set.

Action: Ensure that GTMSECSHR executables and their parent directories have correct permissions.

SECSHRPERMINCRCT

SECSHRPERMINCRCT, dddd permissions incorrect (pppp). gtmsecshr will not be started

Runtime error: This error indicates that pppp, the current permissions of dddd, are incorrect.

Action: Ensure that GTMSECSHR executables and their parent directories have correct permissions.

SECSHRSETGTMDISTFAILED

SECSHRSETGTMDISTFAILED, setenv for gtm_dist failed. gtmsecshr will not be started

Runtime error: This error indicates that the GTMSECSHR process failed to overwrite gtm_dist environment variable.

Action: Ensure that the root user is configured with adequate space for environment variables.

SECSHRSETGTMTMPFAILED

SECSHRSETGTMTMPFAILED, setenv for gtm_tmp failed. gtmsecshr will not be started

Runtime error: This error indicates that the GTMSECSHR process failed to overwrite gtm_tmp environment variable.

Action: Ensure that the root user is configured with adequate space for environment variables.

SECSHRSETUIDFAILED

SECSHRSETUIDFAILED, setuid failed. gtmsecshr will not be started

Runtime error: This error indicates that the GTMSECSHR process failed to set its effective user ID to root.

Action: Ensure that GTMSECSHR executables and their parent directories have correct permissions.

SECSHRSTATFAILED

SECSHRSTATFAILED, stat failed on dddd, errno xxxx. gtmsecshr will not be started

Runtime error: This error indicates that the GTMSECSHR process failed to obtain file information on dddd, and that the error code is xxxx.

Action: Ensure that GTMSECSHR executables and their parent directories have correct permissions.

SECSHRWRITABLE

SECSHRWRITABLE, ffff writable. gtmsecshr will not be started

Runtime error: This error indicates that ffff has 'write' permission.

Action: Ensure that GTMSECSHR executables and their parent directories have correct permissions.

SEMID

SEMID, Semaphore id nnnn

MUPIP informational message: This message reports additional information for an associated error which had trouble with the semaphore with id nnnn.

Action: Check "ipcs -s" for the given id, and see the associated error.

ZPEEKNORPLINFO

ZPEEKNORPLINFO, $ZPEEK() unable to access requested replication structure

Runtime error: The replication structure specified in the mnemonic is not available to this process because either replication is not running or, in the case of receive pool type requests, is not running on a non-primary where such control blocks are available.

Action: Only fetch replication values when replication is active and if accessing gtmrecv.* fields, do not run on a primary.