V6.1-000

GT.M

V6.1-000 Release Notes

Contact Information

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

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

Legal Notice

Revision History		
Revision 1.2	22 July 2014	Improved the description of GTM-7355.
Revision 1.1	10 January 2013	Removed all V6.0-000 messages from Error and Other Messages.
Revision 1.0	9 January 2013	V6.0-001 - First published version

Table of Contents

V6.0-001

Overview

V6.0-001 brings a number of useful enhancements to GT.M. For example:

- An environment variable to provide an initial value to $ETRAP at process startup, overriding the default $ZTRAP="B", which in the event of a buggy application error trap can drop the process to direct mode, undesirable behavior in production
- Significantly speed-up a common type of indirection, of the form @x@(), where the subscripts vary with each invocation
- For the symmetric cipher, demonstrating support in the encryption plugin for both libgcrypt and OpenSSL, and with the latter, a choice of AES CFB and Blowfish CFB as the cipher
- On Linux, an option to use huge pages, for those applications that can benefit from a significant reduction in the use of virtual memory page tables
- The new MUPIP SIZE command with a choice of techniques to rapidly estimate global variable size
- For application code that uses, or may use, side-effects in ways where order of evaluation matters, an option to direct the compiler to generate object code with strict left to right evaluation, bypassing certain optimizations that can reorder evaluation
- New functionality in utility programs, for example, the LOOP^%XCMD entryref to facilitate filters in GT.M, where one might otherwise use the UNIX awk utility
- A new $ZWRITE() function to return a string with non-graphic characters represented in the $CHAR() format produced by the ZWRITE command

There are numerous other enhancements, improvements to ease of use and robustness, and bug fixes.

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: $char(10) mupip backup -database ACN,HIST /backup	
Reference Number	A reference number is used to track software $char(10) enhancements and support requests. $char(10) It is enclosed between parentheses ().	
Platform Identifier	[UNIX]	[OpenVMS]

	UNIX	OpenVMS
	The platform identifier does not appear $char(10) for those new features or $char(10) 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.

Δ 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 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.**

Platform	Supported Versions	Notes
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. GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.5-49 or later) and ncurses (version 5.5-24 or later). The minimum CPU must have the instruction set of a 686 (Pentium Pro) or equivalent.

Platform support lifecycle

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

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

Migrating to 64-bit platforms

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

- You must compile the application code separately for each platform. Even though the M source code is exactly the same, the generated object modules are different even on the same hardware architecture - the object code differs between x86 and x86_64.

- Parameter-types that interface GT.M with non-M code using C calling conventions must match the data-types on their target platforms. Mostly, these parameters are for call-ins, external calls, internationalization (collation), and environment translation and are listed in the tables below. Note that most addresses on 64-bit platforms are 8 bytes long and require 8 byte alignment in structures whereas all addresses on 32-bit platforms are 4 bytes long and require 4-byte alignment in structures.

Call-ins and External Calls

Parameter type	32-Bit	64-bit	Remarks
gtm_long_t	4-byte (32-bit)	8-byte (64-bit)	gtm_long_t is much the same as the C language long type, except on Tru64 UNIX, where GT.M remains a 32-bit application.
gtm_ulong_t	4-byte	8-byte	gtm_ulong_t is much the same as the C language unsigned long type.
gtm_int_t	4-byte	4-byte	gtm_int_t has 32-bit length on all platforms.
gtm_uint_t	4-byte	4-byte	gtm_uint_t has 32-bit length on all platforms

Caution

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

Internationalization (Collation)

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

Important

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

Environment Translation

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

Important

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

Recompile

- Recompile all M and C source files.

Rebuild Shared Libraries or Images

- Rebuild all Shared Libraries (UNIX) or Shareable Executable Images (OpenVMS) after recompiling all M and C source files..

Additional Installation Instructions

To install GT.M, see the "Installing GT.M" section in the GT.M Administration and Operations Guide. For minimal down time upgrade a current replicating instance, restart replication, once that replicating instance is current, switch it over to originating instance and upgrade the prior originating instance to become a replicating instance, at least until it's current.

UNIX

- FIS strongly recommends installing each version of GT.M in a separate (new) directory, rather than overwriting a previously installed version. If you have a legitimate need to overwrite an existing GT.M installation with a new version, you must first shut down all processes using the old version. FIS suggests installing GT.M V6.0-000 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.0-000_arch (for example, /usr/lib/fis-gtm/V6.0-000_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.0-000_arch would also be appropriate. Note that the ***arch*** suffix is especially important if you plan to install 32- and 64-bit versions of the same release of GT.M on the same system.
- Use the MUPIP RUNDOWN command of the old GT.M version to ensure all database files are cleanly closed.
- In UNIX editions, make sure gtmsecshr is not running. If gtmsecshr is running, first stop all GT.M processes including the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP *pid_of_gtmsecshr***.

Caution

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

Additional Information for JFS1 on AIX

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

OpenVMS

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

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

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

If you define GT.M commands locally with **SET COMMAND GTM$DIST:GTMCOMMANDS.CLD** in **GTMLOGIN.COM** or other command file for each process which uses GT.M, you must execute the same command after installing the new version of GT.M and before using it. If you define the GT.M commands to the system other than during the installation of GT.M, you must update the system DCLTABLES with the new GTMCOMMANDS.CLD provided with this version of GT.M. See the OpenVMS "Command Definition, Librarian, and Message Utilities Manual" section on "Adding a system command." In both cases, all GT.M processes must match the proper **GTMCOMMANDS.CLD** with the version of GT.M they run..

Upgrading to GT.M V6.0-000

The GT.M database consists of four types of components- database files, journal files, global directories, and 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 V6.0-000 consists of 5 stages:

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

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

Stage 1: Global Directory Upgrade

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

To upgrade from any previous version of GT.M:

- Open your Global Directory with the GDE utility program of GT.M V6.0-000.
- Execute the EXIT command. This command automatically upgrades the Global Directory.

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

1. Open your Global Directory with the GDE utility program on the 32-bit platform.

2. On GT.M versions that support SHOW -COMMAND, execute SHOW -COMMAND -FILE=file-name. This command stores the current Global Directory settings in file-name.

3. On GT.M versions that do not support GDE SHOW -COMMAND, execute the SHOW -ALL command. Use the information from the output to create an appropriate command file or use it as a guide to manually enter commands in GDE.

4. Open GDE on the 64-bit platform. If you have a command file from 2. or 3., execute @file-name and then run the EXIT command. These commands automatically create the Global Directory. Otherwise use the GDE output from the old Global Directory and apply the settings in the new environment.

An analogous procedure applies in the reverse direction.

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

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

- Open the global directory with the GDE utility program of V6.0-000.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

Stage 2: Database Files Upgrade

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

A V6.0-000 database file is a superset of a V5.0-000 database file and has potentially longer keys and records. Therefore, upgrading a database file requires no explicit procedure. After upgrading the Global Directory, opening a V5 database with a V6 process automatically upgrades 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 V6.0-000 operations. However, that database can only grow to the maximum size of the version in which it was originally created. If the database is V5.0-000 through V5.3-003, the maximum size is 128Mi blocks. If the database is V5.4-000 through V5.5-000, the maximum size is 224Mi blocks. Only a database created with V6.0-000 (with a V6 MUPIP CREATE) can have a maximum database size of 992Mi blocks.

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

- Execute the MUPIP REORG -UPGRADE command again, or

- Execute the DSE CHANGE -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 have a maximum size of 1,040,187,392(992Mi) blocks.

Stage 3: Replication Instance File Upgrade

V6.0-000 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.0-000 requires new replication instance files if you are upgrading from any version prior to V5.5-000. Instructions for creating new replication instance files are in the Supplementary Instance Replication Technical Bulletin. Shut down all Receiver Servers on other instances that are to receive updates from this instance, shut down this instance Source Server(s), recreate the instance file, restart the Source Server(s) and then restart any Receiver Server for this instance with the -UPDATERESYNC qualifier.

Note

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

Important

You must always follow the steps in the Multi-Site Replication technical bulletin when migrating from a logical dual site (LDS) configuration to an LMS configuration, even if you are not changing GT.M releases.

Stage 4: Journal Files Upgrade

On every GT.M upgrade:

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

Important

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

Stage 5: Trigger Definitions Upgrade

If you are upgrading from V5.4-002A/V5.4-002B/V5.5-000 to V6.0-000, you do not need to extract and reload triggers.

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

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

1. Execute a command like **mupip trigger -select="*" trigger_defs.trg** using the old version. Now, the output file trigger_defs.trg contains all trigger definitions.
2. Place -* at the beginning of the trigger_defs.trg file to remove the old trigger definitions.
3. Run **mupip trigger -triggerfile=trigger_defs.trg** using V6.0-000 to reload your trigger definitions.

To extract and reload trigger definitions on a V6.0-000 replicating instance using $ZTRIGGER():

1. Shut down the instance using the old version of GT.M.
2. Execute a command like **mumps -run %XCMD 'i $ztrigger("select")' > trigger_defs.trg** . Now, the output file trigger_defs.trg contains all trigger definitions.
3. Turn off replication on all regions.
4. Run **mumps -run %XCMD 'i $ztrigger("item","-*")** to remove the old trigger definitions.
5. Perform the upgrade procedure applicable for V6.0-000.
6. Run **mumps -run %XCMD 'if $ztrigger("file","trigger_defs.trg")'** to reapply your trigger definitions.
7. Turn replication on.
8. Connect to the originating instance.

Note

Reloading triggers renumbers automatically generated trigger names.

Downgrading to V5 or V4

You can downgrade a GT.M V6.0-000 database to V5 or V4 format using MUPIP DOWNGRADE.

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

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

-VERSION specifies the desired version for the database header.

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

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

If your database meets all the above requirements, MUPIP DOWNGRADE -VERSION=V5 resets the database header to V5 elements which makes it compatible with V5 versions.

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

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

1. In a GT.M V.6-000 environment:

    a. Execute MUPIP SET -VERSION=v4 so that GT.M writes updates blocks in V4 format.

    b. Execute MUPIP REORG -DOWNGRADE to convert all blocks from V6 format to V4 format.

2. Bring down all V6 GT.M processes and execute MUPIP RUNDOWN -FILE on each database file to ensure that there are no processes accessing the database files.

3. Execute MUPIP DOWNGRADE -VERSION=V4 to change the database file header from V6 to V4.

4. Restore or recreate all the V4 global directory files.

5. Your database is now successfully downgraded to V4.

Managing M mode and UTF-8 mode

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

On a system that has ICU installed, GT.M optionally installs support for both M mode and UTF-8 mode, including a utf8 subdirectory of the directory where GT.M is installed. From the same source file, depending upon the value of the environment variable gtm_chset, the GT.M compiler generates an object file either for M mode or UTF-8 mode. GT.M generates a new object file when it finds both a source and an object file, and the object predates the source file and was generated with the same setting of $gtm_chset/$ZCHset. A GT.M process generates an error if it encounters an object file generated with a different setting of $gtm_chset/$ZCHset than that processes' current value.

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

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

- Actual files for GT.M executable programs (mumps, mupip, dse, lke, and so on) are in the parent directory, that is, the location specified for installation.

- Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for Unicode in the utf8 subdirectory, and one compiled without support for Unicode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install Unicode support.

- The utf8 subdirectory has files called mumps, mupip, dse, lke, and so on, which are relative symbolic links to the executables in the parent directory (for example, mumps is the symbolic link ../mumps).
- When a shell process sources the file gtmprofile, the behavior is as follows:
  - If $gtm_chset is "m", "M" or undefined, there is no change from the previous GT.M versions to the value of the environment variable $gtmroutines.
  - If $gtm_chset is "UTF-8" (the check is case-insensitive),
    - $gtm_dist is set to the utf8 subdirectory (that is, if GT.M is installed in /usr/lib/fis-gtm/gtm_V6.0-000_i686, then gtmprofile and gtmcshrc set $gtm_dist to /usr/lib/fis-gtm/gtm_V6.0-000_i686/utf8).
    - On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

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

Compiling ICU

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

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

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

Setting the environment variable TERM

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

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

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

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

Installing Compression Libraries

If you plan to use the optional compression facility for replication, you must provide the compression library. The GT.M interface for compression libraries accepts the zlib compression libraries without any need for adaptation. These libraries are included in many UNIX distributions and are downloadable from the zlib home page. If you prefer to use other compression libraries, you need to configure or adapt them to provide the same API provided by zlib. Simple instructions for compiling zlib on a number of platforms follow. Although GT.M can use zlib, zlib is not FIS software and FIS does not support zlib. These instructions are merely provided as a convenience to you.

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

Solaris/cc compiler from Sun Studio:

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

HP-UX(IA64)/HP C compiler:

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

AIX/XL compiler:

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

Linux/gcc:

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

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

V6.0-001

Fixes and enhancements specific to V6.0-001 are:

Id	Prior Id	Category	Summary
GTM-3357	C9A10-001605	MUMPS	ZMESSAGE guards against a set of messages it should not issue
GTM-3907	C9B10-001766	MUMPS	Option for strict left-to-right order of evaluation where side effect expressions can impact the result
GTM-3989	C9B11-001818	MUMPS	Process exit status not success after errors from database close
GTM-4115	C9C01-001890	MUPIP	MUPIP EXTRACT can redirect database extract to standard output
GTM-4661	S9C09-002229	MUMPS	Improve the handling of terminal signals (TSTP, TTIN and TTOU)
GTM-5378	C9E04-002551	MUMPS	VIEW command keyword [NO]LOGT[PRESTART] [=intexpr] dynamically controls the logging of TPRESTART message.
GTM-5870	C9G04-002787	DB	JNLPROCSTUCK and JNLFSYNCLSTCK errors are warning rather than info
GTM-5896	C9G06-002803	MUMPS	Speed-up for name indirection - @x@(...)
GTM-6015	C9H05-002858	MUMPS	CMDLINE parameter for JOB command to allow parent to control child command line
GTM-6395	C9J06-003133	MUMPS	Option for strict left-to-right order of evaluation within indirection in SET command
GTM-6634	C9K06-003285	DB	Encryption calls are now interrupt safe
GTM-7160	-	MUMPS	New %TRIM, %MPIECE, and %DSEWRAP utility programs and a new utility label for %XCMD called LOOP.
GTM-7168	-	MUPIP	Improve recoverability from GT.M replication source-receiver and GT.CM client-server dropped connections
GTM-7192	-	DB	Replace a GTMASSERT after attempts longer than four minutes to write a dirty buffer to disk with indefinite retries

Change History

Id	Prior Id	Category	Summary
GTM-7241	-	MUPIP	MUPIP JOURNAL -RECOVER -FORWARD works correctly for database files with the MM access method
GTM-7254	-	MUMPS	Enhance the LOCK command to ensure fairer access when multiple processes need the same lock resource
GTM-7281	-	MUMPS	Compiler return changed from 255 to 1 to play nice with xargs
GTM-7292	-	MUPIP	MUPIP command for fast global sizing
GTM-7355	-	MUMPS	Handles TPFAIL errors inside a trigger by raising an appropriate error
GTM-7356	-	MUPIP	Report IPC identifiers associated with the Journal Pool and the Receive Pool in the log files
GTM-7374	-	Other Utilities	README, README.txt, and COPYING files in all GT.M distribution kits are read-only.
GTM-7387	-	MUMPS	Release extra DECnet channel
GTM-7395	-	MUMPS	Immediate error trying to OPEN a directory as a file
GTM-7415	-	MUMPS	$ZMESSAGE() protects against unassigned error codes
GTM-7427	-	MUPIP	Issue a MUNOTALLINTEG warning when MUPIP INTEG fails to perform integrity check on one or more requested regions
GTM-7432	-	MUPIP	Issue a BKUPRUNNING error when MUPIP BACKUP detects an ongoing backup
GTM-7443	-	DB	Write idle EPOCH journal records to the journal file only when there is no update for approximately 5 seconds.
GTM-7445	-	DB	See GTM-7451.
GTM-7450	-	DB	See GTM-7451.
GTM-7451	-	DB	Database encryption plugin usability enhancements and bug fixes
GTM-7454	-	MUPIP	INTEG -FAST should cause minimal load on other processes
GTM-7455	-	MUPIP	OpenVMS REORG fix
GTM-7458	-	MUPIP	On receiving MUPIP STOP, Receiver Server resets IPC fields in instance file header

Change History

Id	Prior Id	Category	Summary
GTM-7461	-	MUPIP	MUPIP RUNDOWN and JOURNAL ROLLBACK do Instance Freeze
GTM-7468	-	DB	Eliminated possible error resuming from an Instance Freeze
GTM-7471	-	DB	Instance Freeze After Error Message
GTM-7474	-	DB	Add DBIOERR and JNLCNTRL to custom_errors_sample.txt
GTM-7475	-	MUMPS	New $ZWRITE() formating function
GTM-7476	-	MUPIP	Fix concurrent truncates and online backups
GTM-7477		DB	See GTM-7451.
GTM-7478	-	MUPIP	Allow MUPIP JOURNAL EXTRACT to successfully read current journal file during instance freeze
GTM-7481	-	DB	Prevent Instance Freeze on ENOSPCQIODEFER
GTM-7485	-	MUPIP	Source and Receiver Server shutdown logic now addresses IPC removal issues
GTM-7486	-	MUPIP	Argumentless MUPIP RUNDOWN now issues correct MUJPOOLRNDWNFL and MURPOOLRNDWNFL messages
GTM-7492	-	MUMPS	FOR command fixes
GTM-7493	-	DB	Instance Freeze Fix for Region Selection
GTM-7494	-	MUMPS	New header file in GT.M distribution provides general purpose macros for use by external calls and plugins
GTM-7495	-	DB	Fix issue with concurrent trigger change
GTM-7497	-	MUMPS	Eliminate INDMAXNEST errors
GTM-7501	-	MUMPS	Correction for sign of exponentiation in intermediate results
GTM-7502	-	MUPIP	Instance Freeze Fix for Update Process Shutdown
GTM-7505	-	DB	Shutdown logic bypass during instance freeze
GTM-7510	-	MUMPS	ZSHOW "G" accurately counts LKS value
GTM-7515	-	MUMPS	gtmxc_types.h enhancements
GTM-7516	-	MUMPS	Additional invocation types in $STACK() form triggers and MUPIP INTRPT
GTM-7517	-	DB	Better handling of errors while flushing global buffers

Change History

Id	Prior Id	Category	Summary
GTM-7525	-	DB	MUPIP INTRPT does not cancel untimed lock operation
GTM-7526	-	MUPIP	Fix a GTMASSERT error that occurs when an update process encounters an Instance Freeze.
GTM-7529	-	DB	Support for huge pages on x86[-64] Linux
GTM-7530	-	MUPIP	Replication and ROLLBACK use JNLFILRDOPN in place of JNLFILOPN
GTM-7534	-	DB	Prevent database damage from creation of new globals in very rare cases of high contention.
GTM-7535	-	MUMPS	$ORDER(@x@(y),$$foo) and similar expressions (using $GET() or $NAME()) evaluate left-to-right
GTM-7536	-	MUMPS	Eliminate duplicate evaluation of indirect first argument of SET $[Z]...()
GTM-7538	-	MUPIP	Checkhealth failure with Instance Freeze
GTM-7541	-	MUMPS	Reverse name-level $ORDER() fix
GTM-7544	-	MUMPS	First $ORDER() uses an extended global reference issue fixed
GTM-7546	-	DB	Improved "Incorrect password" message from the reference implementation of the Encryption Plugin
GTM-7550	-	MUMPS	Improved handling of RESTART when in an interrupted error
GTM-7552	-	MUMPS	TP RESTART handles a LOCK in an untouched region correctly
GTM-7553	-	Other Utilities	LKE SHOW -MEM displays LOCKSPACEINFO
GTM-7555	-	DB	No DBIOERR when space is insufficient and instance freeze is enabled
GTM-7556	-	MUMPS	Recognize >= as the syntactic equivalent of '< and <= as the syntactic equivalent of '>
GTM-7557		DB	In Anticipatory Freeze supported environments, Update helpers now open Journal Pool
GTM-7559	-	Other Utilities	Edge case correction for DBCERTIFY
GTM-7564	-	DB	OpenVMS concurrency fix
GTM-7566	-	MUPIP	Source Server gives error when it can't get a needed record
GTM-7571	-	DB	Fix for $REFERENCE maintenance during certain error cases

Change History

Id	Prior Id	Category	Summary
GTM-7575		DB	In instance freeze environments, Journaling does not get turned off.

M-Database Access

- GT.M reports JNLPROCSTUCK and JNLFSYNCLSTCK errors with a -W- (warning) severity level. Previously, it gave these messages a -I- (informational) severity level. (GTM-5870)
- GT.M now invokes all encryption functions, defined in gtmcrypt_interface.h and implemented by an encryption plugin, in an interrupt-safe manner. Previously, only the gtmcrypt_encrypt & gtmcrypt_decrypt functions were invoked in an interrupt-safe manner. In addition to MUPIP INTRPT, GT.M processes use timer interrupts to cooperatively manage databases. As a consequence of not calling the functions in an interrupt safe manner, a process could in theory conclude, for example, that it had an incorrect password for a database, resulting in a run-time error, or possibly process termination. Although theoretically possible, the probability of such a scenario is extremely low and we are not aware of any user ever having encountered it; indeed, we were unable to create such a scenario in the GT.M development and testing environment. The worst consequences of such a scenario are errors within a process, or premature process termination - there is no potential for database damage or data loss from data written to the database encrypted with the wrong password. [UNIX] (GTM-6634)
- A GT.M process attempting to read a database block from disk to the global buffer pool retries indefinitely. Previously, failure to read a database block in 4 minutes caused GTMASSERT. (GTM-7192)
- GT.M writes idle EPOCHs, which ZSHOW "G" or DSE DUMP -FILEHEADER count in the JRI statistic, only when there is no update for approximately five (5) seconds. Previously, with multiple processes under low update activity, several processes might write idle EPOCHs due to their own inactivity, even though other processes were actually updating at a light rate and the database was not actually static. [UNIX] (GTM-7443)
- See GTM-7451. (GTM-7445)
- See GTM-7451. (GTM-7450)
- V6.0-001 brings changes to the database encryption plugin to provide more choices, to make it easier to use, and to fix bugs and misfeatures. If you are currently using encrypted databases, please note the following two important points regarding upward compatibility:

  1. If you use encrypted databases on AIX, the default symmetric cipher in the reference implementation of the plugin for AIX is now AES CFB. Should you wish to continue to use Blowfish CFB with V6.0-001 via the reference implementation of the plugin, you need to change a symbolic link post-installation, or provide an environment variable, as described below.

     

     Note

     To migrate a database from Blowfish CFB to AES CFB requires the data to be extracted and loaded into newly created database files. To reduce the time your application is unavailable to seconds to minutes, you can deploy your application in a Logical Multi-Site (LMS) configuration, and migrate using a rolling upgrade technique. Refer to the GT.M Administration and Operations Guide, UNIX Edition, for more complete documentation.

  2. If you have modified the encryption plugin, please note the change in the name of two API functions: from gtmcrypt_encode and gtmcrypt_decode, to gtmcrypt_encrypt and gtmcrypt_decrypt respectively. As there is no change to any function parameter, you can simply replace the function names. By exposing the same functionality with dual names, you can make your modified plugin work with both V6.0-000 and V6.0-001 - the former uses only the old names and the latter uses only the new names.

The choice of encryption plugin and cipher is effected in two ways:

1. If the environment variable gtm_crypt_plugin is defined and provides the path to a shared library relative to $gtm_dist/plugin, GT.M uses $gtm_dist/plugin/$gtm_crypt_plugin as the shared library providing the plugin.

2. If $gtm_crypt_plugin is not defined, GT.M expects $gtm_dist/plugin/libgtmcrypt.so to be a symbolic link to a shared library providing the plugin. The expected name of the actual shared library is libgtmcrypt_cryptlib_CIPHER.so (depending on your platform, the actual extension may differ from .so), for example, libgtmcrypt_openssl_AESCFB. GT.M cannot and does not ensure that the cipher is actually AES CFB as implemented by OpenSSL - GT.M uses CIPHER as salt for the hashed key in the database file header, and cryptlib is for your convenience, for example, for troubleshooting. Installing the GT.M distribution creates a default symbolic link.

When a GT.M process first opens a shared library providing an encryption plugin, it ensures that the library resides in $gtm_dist/plugin or a subdirectory thereof. This ensures that any library implementing an encryption plugin requires the same permissions to install, and is protected by the same access controls, as the GT.M installation itself.

On all platforms on which GT.M supports encryption, the distribution includes three reference implementations of the plugin, provided by the shared libraries libgtmcrypt_gcrypt_AES256CFB.so, libgtmcrypt_openssl_AES256CFB.so and libgtmcrypt_openssl_BLOWFISHCFB.so. On installation, platforms other than AIX, libgtmcrypt.so is a symbolic link to libgtmcrypt_gcrypt_AES256CFB.so; on AIX symbolic link is to libgtmcrypt_openssl_AESCFB.so. To use Blowfish CFB as the default, you should change the symbolic link after installing GT.M. To use AES CFB as the default, but retain Blowfish CFB for databases migrated from V6.0-000, you can set gtm_crypt_plugin when using those databases. For scripts intended to be portable between V6.0-000 and V6.0-001, you can safely set a value for gtm_crypt_plugin, which V6.0-000 ignores.

Enhancements to the helper scripts to accompany these changes are:

- An optional third command line parameter, in addition to cryptographic library (gcrypt or openssl) and build type (d or p), to build.sh allows you to specify the cipher (AES256CFB or BLOWFISHCFB) with which to build the encryption plugin, defaulting to AES256CFB if not specified.

- The install.sh script now requires a mandatory first parameter, which must be gcrypt or opensssl, and an optional second parameter, which defaults to AES256CFB if not specified.

- show_install_config.sh script reports the cryptographic library and cipher that a GT.M process would use, from $gtm_crypt_plugin, if it has a value and otherwise from the library linked to by libgtmcrypt.so.

- The gen_sym_hash.sh script uses the show_install_config.sh, script to identify the currently installed encryption configuration so that it can generate the appropriate cryptographic hash for the provided symmetric key.

GT.M V6.0-001 includes fixes to bugs and misfeatures pertaining to encrypted databases:

1. The reference implementation of the plugin now reports correct error messages. Previously it sometimes combined part of the text of a prior error with that of the current error, and also sometimes reported the incorrect region for an error.

2. MUPIP, DSE (and other utilities) now report encryption configuration related errors at process startup. Previously, they deferred reporting such errors until they actually needed to open an encrypted database file affected by the error.

3. The reference implementation of the plugin now parses the file pointed to by $gtm_dbkeys more robustly. Previously, in some rare cases, the parsing logic did not report errors correctly and did not manage allocated memory properly.

4. On unsuccessful completion, MUPIP LOAD now terminates with MUNOFINISH error message (in addition to the actual error messages that it reports). Previously, it terminated with a less helpful and undocumented message.

5. MUPIP INTEG now correctly exits with a non-zero status when encryption initialization fails for one or more regions. Previously, any errors during encryption initialization for a region resulted in a zero exit status.

The reference implementations of encryption plugins shipped with GT.M are now tested using AES CFB as implemented by OpenSSL (version >= 0.9.8) and libgcrypt (version >= 1.5).

Please remember that FIS neither endorses nor supports your use of any specific cryptographic library or cipher. Before using encrypted databases, please make sure you read and fully understand the Disclaimer in Chapter 12, Database Encryption, of the GT.M Administration and Operations Guide, UNIX Edition.

[UNIX] (GTM-7445) (GTM-7450) (GTM-7451)

- GT.M now resumes appropriately from an Instance Freeze. In V6.0-000, there was a small window in which a process resuming from an Instance Freeze might inappropriately issue an error. [UNIX](GTM-7468)
- GT.M now reports error messages which trigger an instance freeze before the freeze is set. Previously the freeze could encounter lock contention which would prevent the error from being reported until the freeze was lifted. [UNIX] (GTM-7471)
- custom_errors_sample.txt now includes two new messages: DBIOERR and JNLCNTRL. [UNIX] (GTM-7474)
- See GTM-7451. (GTM-7477)
- GT.M no longer logs a JNLACCESS message with ENOSPCQIODEFER. Previously, with Instance Freeze configured, if the journal file system ran out of space doing a non-critical write the JNLACCESS message triggered an Instance Freeze which would not be automatically cleared when space became available.[UNIX](GTM-7481)
- GT.M database errors check the Instance Freeze settings for the appropriate region. In V6.0-000, there were certain cases where an error could check the settings in the wrong region. [UNIX] (GTM-7493)
- GT.M invokes the current set of triggers for a global when trigger definitions change concurrently. Previously, in case a GT.M process had invoked a set of trigger definitions for a global as part of an update outside of a TSTART/TCOMMIT fence and those definitions changed concurrently, the first update (after the trigger definition change) to that global by the GT.M process incorrectly used the old set of trigger definitions. Subsequent updates would correctly use the revised triggers; only the first update used the out-of-date triggers. [UNIX](GTM-7495)
- When an Instance Freeze has stopped updates, GT.M and its utilities bypass database shutdown logic that typically occurs as part of process termination. In V6.0-000, exiting processes would hang if they encountered a frozen region. [UNIX] (GTM-7505)
- GT.M appropriately handles I/O errors that occur as it attempts to flush global buffers to database files at process exit. Previously, such an I/O error could cause process termination with a GTMASSERT (GTM-7517)
- An untimed LOCK waiting for a resource name owned by another process resumes waiting for the LOCK after a MUPIP INTRPT. In V5.5-000, the change to better manage the timeout when a process received a MUPIP INTRPT during a timed LOCK (GTM-5997) introduced a regression in untimed LOCKs that caused the process to continue past the LOCK command without ensuring it had acquired the specified resource(s). (GTM-7525)
- GT.M on Linux now supports using huge pages (depending on the CPU architecture, typically 2MiB or 4MiB rather than the default 4KiB) for shared memory segments for BG access and for the heap of mumps processes. Using huge pages reduces the need for page tables at the cost of a slight increase in total system virtual memory usage. While your mileage may vary, FIS believes that many large applications will benefit from configuring and using huge pages. If you do not configure your system for huge pages, GT.M continues to use the default page size. Note that using huge pages requires no change whatsoever at the application code level, or even to database management operations such as replication, backup, integ,

reorg, etc. - the changes discussed here pertain entirely to configuring system memory and the virtual memory manager to improve application performance.

To use huge pages:

- You must have a 32- or 64-bit x86 CPU, running a Linux kernel with huge pages enabled. All currently Supported Linux distributions appear to support huge pages; to confirm, use the command: grep hugetlbfs /proc/filesystems which should report: nodev hugetlbfs
- You must have libhugetlbfs.so installed in a standard location for system libraries. Installing the library using your Linux system's package manager should place it in a standard location. (Note that libhugetlbfs is not in Debian repositories and must be manually installed; GT.M on Debian releases is Supportable, not Supported.)
- You must have a sufficient number of huge pages available:
  - To reserve Huge Pages boot Linux with the hugepages=num_pages kernel boot parameter; or, shortly after bootup when unfragmented memory is still available, with the command: hugeadm --pool-pages-min DEFAULT:num_pages command
  - For subsequent on-demand allocation of Huge Pages, use: hugeadm --pool-pages-max DEFAULT:num_pages or set the value of /proc/sys/vm/nr_overcommit_hugepages. These delayed (from boot) actions do not guarantee availability of the requested number of huge pages; however, they are safe as, if a sufficient number of huge pages is not available, Linux simply uses traditional sized pages.
- To use huge pages for shared memory segments for the BG database access mode, both of the following are required:
  - Permit GT.M processes to use huge pages for shared memory segments (where available, FIS recommends 1, below; however not all file systems support extended attributes). Either:
    1. set the CAP_IPC_LOCK capability needs for your mumps, mupip and dse processes with a command such as setcap 'cap_ipc_lock+ep' $gtm_dist/mumps, or
    2. permit the group used by GT.M processes needs to use huge pages with, as root, echo gid >/proc/sys/vm/hugetlb_shm_group.
  - Set the environment variable HUGETLB_SHM for each process to yes.
- To use huge pages for process working space and dynamically linked code:
  - Set the environment variable HUGETLB_MORECORE for each process to yes.
- Although not required to use huge pages, your application is also likely to benefit from including the path to libhugetlbfs.so in the LD_PRELOAD environment variable.
- If you enable huge pages for all applications (by setting HUGETLB_MORECORE, HUGETLB_SHM, and LD_PRELOAD as discussed above in /etc/profile and/or /etc/csh.login), you may find it convenient to suppress warning messages from common applications that are not configured to take advantage of huge pages by also setting the environment variable HUGETLB_VERBOSE to zero (0).

At this time, huge pages cannot be used for MM databases; the text, data, or bss segments for each process; or for process stack.

Refer to the documentation of your Linux distribution for details. Other sources of information are:

- http://www.kernel.org/doc/Documentation/vm/hugetlbpage.txt
- http://lwn.net/Articles/374424/
- http://www.ibm.com/developerworks/wikis/display/LinuxP/libhuge+short+and+simple
- the HOWTO guide that comes with libhugetlbfs (http://sourceforge.net/projects/libhugetlbfs/files/)

Note

- In order to assure predictable performance, FIS strongly recommends that you invest the effort to appropriately tune your system before using huge pages in production. For example, use of on demand huge pages (over-committing) may result in delays while the kernel performs memory management activities to satisfy requests for memory - but this may or may not be material in your situation.
- Since the memory allocated by Linux for shared memory segments mapped with huge pages is rounded up to the next multiple of huge pages, there is potentially unused memory in each such shared memory segment. You can therefore increase any or all of the number of global buffers, journal buffers, and lock space to make use of this otherwise unused space. You can make this determination by looking at the size of shared memory segments using ipcs. Contact FIS GT.M support for a sample program to help you automate the estimate.
- Transparent huge pages may further improve virtual memory page table efficiency. Some Supported releases automatically set transparent_hugepages to "always"; others may require it to be set at or shortly after boot-up. Consult your Linux distribution's documentation.

We categorized this as a Database item in these Release Notes because we expect it to have the largest impact on shared pages used by BG access method database files, but it also affects process foot-print [Linux on x86/x86_64] (GTM-7529)

- GT.M now manages lookups of non-existent global variable roots (unsubscripted names) more carefully in very rare cases of heavy database contention. Starting in V5.4-000, such a rare event could cause incorrect results, and, in case of an update, create database damage reported by MUPIP as a DBINVGBL integrity error. (GTM-7534)
- The reference implementation of the encryption plugin reports "Incorrect password or error while obtaining password" as the secondary text to CRYPTKEYFETCHFAILED message in cases where GPGME provided a less descriptive error code. Previously, the encryption plugin reported "Incorrect password" as the secondary text even when the password was correct in the environment but an error occurred during retrieval of the password. Refer to the revised CRYPTKEYFETCHFAILED message in Message and Recovery manual for more details. [UNIX] (GTM-7546)
- Failures of non-critical writes intended to reduce the number of dirty global buffers caused by a (potentially transient) lack of space in the database file system do not report a DBIOERR in the system log, and do not initiate a freeze in environments configured for Instance Freeze. Previously, in such cases, GT.M reported a DBIOERR in the system log, and also initiated a freeze which required an explicit MUPIP REPLICATE -FREEZE=OFF action. Note that all critical database writes (those required to ensure durability) generated and continue to generate ENOSPC messages, and initiate freezes. Instances frozen with an ENOSPC error monitor available space, and automatically release the freeze when space becomes available. [UNIX] (GTM-7555)
- If a replicating secondary instance or a supplementary instance is configured for Instance Freeze, the Update Helper processes - Readers and Writers - honor Instance Freeze. Previously, the Update Helpers did not open the Journal Pool and as a result could neither initiate nor honor Instance Freeze. [UNIX] (GTM-7557)

- Database updates correctly handle heavy IO loads on OpenVMS. Previously, owing to a potential race condition in a very small window, in rare cases involving concurrent database updates, the database could potentially sustain structural damage, with the probability of damage increasing with update concurrency. FIS is not aware of this having been encountered at any customer site. [OpenVMS] (GTM-7564)

- GT.M appropriately handles $REFERENCE in certain unusual error conditions such as out-of-memory (MEMORY error mnemonic). In previous versions, if these unusual errors occurred while performing a database access, and the next global reference (either inside the error trap or in some routine to which the trap transferred control) relied on $REFERENCE or a naked reference, that subsequent reference could have been incorrect. (GTM-7571)

- When a processes that is part of an instance configured for instance freeze behavior encounters an error with journaling, it freezes the instance and invokes its own error trap even if it does not have the gtm_custom_errors environment variable set. Previously, under certain circumstances, such as being unable to create a new journal file because of directory permissions, a process in an Instance Freeze environment might turn journaling off. Since the operation of GT.M processes is controlled with environment variables, and GT.M relies on directory and file access permissions of the underlying operating system, it was, and remains, possible to set up abnormally configured systems where processes without appropriate values of environment variables can cause undesirable results such as turning off journaling in instances configured for instance freeze. [UNIX] (GTM-7575)

M-Other Than Database Access

- ZMESSAGE transforms two sets of error messages into SPCLZMSG errors. The internal error messages which should not be user visible and the error messages which are expected to be driven when their corresponding internal state is available. The list of such errors is as follows: (CTRLY, CTRLC, CTRAP, JOBINTRRQST, JOBINTRRETHROW, REPEATERROR, STACKCRIT, SPCLZMSG, TPRETRY, UNSOLCNTERR). Previously, passing to ZMESSAGE the IDs of some of the these errors could cause the process to terminate with a GTMASSERT. [UNIX](GTM-3357)

- When the UNIX environment variable gtm_side_effects (OpenVMS logical name GTM_SIDE_EFFECTS) is set to one (1) at process startup, GT.M generates code that performs left to right evaluation of actuallist arguments, function arguments, operands for non-Boolean binary operators, SET arguments where the target destination is an indirect subscripted glvn, and variable subscripts. When the environment variable is not set, or set to zero (0), GT.M retains its traditional behavior, which re-orders the evaluation of operands using rules intended to improve computational efficiency. This reordering assumes that functions have no side effects, and may generate unexpected behavior (x+$increment(x) is a pathological example). When gtm_side_effects is set to two (2), GT.M generates code with the left-to-right behavior, and also generates SIDEEFFECTEVAL warning messages for each construct that potentially generates different results depending on the order of evaluation. As extrinsic functions and external calls are opaque to the compiler at the point of their invocation, it cannot statically determine whether there is a real interaction. Therefore SIDEEFFECTEVAL warnings may be much more frequent than actual side effect interactions and the warning mode may be most useful as a diagnostic tool to investigate problematic or unexpected behavior in targeted code rather than for an audit of an entire application. Note that a string of concatenations in the same expression may generate more warnings than the code warrants. Other values of the environment variable are reserved for potential future use by FIS. It is important to note that gtm_side_effects affects the generated code, and must be in effect when code is compiled - the value when that compiled code is executed is irrelevant. Note also that XECUTE and auto-ZLINK, explicit ZLINK and ZCOMPILE all perform run-time compilation subject to the characteristic selected when the process started. Please be aware that programming style where one term of an expression changes a prior term in the same expression is an unsafe programming practice. The existing UNIX environment variable gtm_boolean (OpenVMS logical name GTM_BOOLEAN) may separately control short-circuit evaluation of Boolean expressions but a setting of 1 (or 2) for gtm_side_effects causes the same boolean evaluations as setting gtm_boolean to 1 (or 2). The differences in the compilation modes may include not only differences in results, but differences in flow of control.(GTM-6395)(GTM-3907) 

- Failure to close any database file at exit causes the process to return a non-success status code to the shell. Previously, such errors had no effect on the status returned to the shell. [UNIX] (GTM-3989)

- When a GT.M process within certain small but sensitive areas of the database code receives a terminal-related signal that would suspend the process because the terminal is unavailable for pending input or output, it now correctly sends itself a continue signal and defers the suspension until it reaches an area of the code where it is safe to suspend. The terminal related signals are TTIN, TTOU, and TSTP, and the continue signal is CONT. Previously, when a process in such a sensitive area received such a signal, it sent itself a continue signal but a problem in signal handling could result in an indefinite loop. If the process happened to hold critical resources this indefinite loop could cause resource starvation, blocking other processes needing resources. For example, a process unable to read a block within approximately four (4) minutes could terminate with a GTMASSERT. Since TTIN and TTOU result from terminal IO by processes running in the background, and TTSTP commonly results from typing Ctrl-Z to an interactive process, the workaround was to avoid these situations. Also, because a process cannot ignore a STOP signal, which causes it to suspend processing immediately, any processes waiting for a critical resource held by a suspended process send the CONT signal to the suspended process. Previously, a STOP signal suspending a GT.M process holding a critical resource could indefinitely block other processes. Note that FIS continues to recommend against suspending GT.M processes. [UNIX](GTM-4661)

- The VIEW command keyword, [NO]LOGT[PRESTART][=intexpr] allows a process to dynamically change the logging of TPRESTART messages to the operator log established at process startup by the environment variables gtm_tprestart_log_delta and gtm_tprestart_log.

  VIEW "NOLOGTPRESTART" turns off the logging of TPRESTART messages to the operator log.

  VIEW "LOGTPRESTART"[=intexpr] turns on logging of TPRESTART messages to the operator log. If no intexpr is specified, GT.M uses the value of environment variable gtm_tprestart_log_delta, if it is defined, and one otherwise (i.e., every transaction restart will be logged). A negative value of intexpr turns off the logging of TPRESTART messages.

  The TPRESTART message has a "local tn value" which may be helpful to FIS GT.M support in the event you contact us for support. It now has a 0x prefixed to it to indicate that it is a hexadecimal number. Previously, it was printed as a series of hexadecimal digits without the leading 0x. (GTM-5378)

- Loops involving indirection of the form @x@() where x is a global or local variable name execute faster when the subscripts vary with each iteration. Although this enhancement takes a different approach, FIS thanks Mikhail Ilyin and Komitex JSC for initially suggesting a refinement in this area. In addition, GT.M correctly evaluates nested indirection. Previously, if a subscripted indirect expression contained both nested indirection and multiple '@' characters, the result could have been incorrect. Also, the ZWRITE command accepts an argument of the form @x@(*), which previously gave a syntax error. (GTM-5896)

- The JOB command jobparameter CMD[LINE]=" <text>" passes the content of <text> to the $ZCMDLINE of the JOB'd process (The cmdline can be shown through the UNIX ps command). This is new functionality.[UNIX](GTM-6015)

- See GTM-3907.(GTM-6395)

- GT.M includes the %TRIM, %MPIECE, and %DSEWRAP utility programs and a new utility label for %XCMD called LOOP. The description of these utilities is as follows:

  1. **%TRIM**

     The %TRIM utility removes leading and trailing whitespace (spaces and tabs) from a string. You can use the %TRIM utility in Direct Mode or include it in a source application program in the following format:

     ```
     $$FUNC^%TRIM(exp)
     ```

     You can also use %TRIM as a command line utility to read from STDIN and write to STDOUT in the following format:

     ```
     %XCMD 'do ^%TRIM'
     ```

     Utility Labels

     The following labels invoke variations of %TRIM as an extrinsic function.

     FUNC(s): Returns a string after removing leading and trailing whitespaces from the argument.

     L(s): Returns a string after removing leading whitespaces from the argument.

     R(s): Returns a string after removing trailing whitespaces from the argument.

     Example:

     ```
     GTM>set strToTrim=$char(9,32)_"string with spaces and tabs"_$char(32,32,32) write $length(strToTrim)
      36 GTM>write "strToTrim=",?24,"""",strToTrim,"""",!,"$$L^%TRIM(strToTrim)=",?24,"""",$$L^
     ```

```
%TRIM(strToTrim),"""",!,"$$R^%TRIM(strToTrim)=",?24,"""",$$R^%TRIM(strToTrim),"""",!,"$$FUNC^
%TRIM(strToTrim)=",?24,"""",$$FUNC^%TRIM(strToTrim),"""$char(10)strToTrim= " string with spaces and tabs
 "$char(10)$$L^%TRIM(strToTrim)= "string with spaces and tabs "$char(10)$$R^%TRIM(strToTrim)= " string with
 spaces and tabs"$char(10)$$FUNC^%TRIM(strToTrim)="string with spaces and tabs"
```

This example invokes %TRIM as an extrinsic function and demonstrates the use of its L,R, and FUNC labels.

Example:

```
$ echo " GT.M Rocks! " | gtm -r %XCMD 'do ^%TRIM'$char(10)$char(10) GT.M Rocks!$char(10)$char(10)$
```

This example invokes %TRIM as a command line utility which reads STDIN and writes the trimmed output to STDOUT.

2. **%MPIECE**

The %MPIECE utility replaces one or more consecutive occurrences of the second argument in the first argument with one occurrence of the third argument. This lets $PIECE operate on the resulting string like UNIX awk.

You can use the %MPIECE utility in Direct Mode or include it in a source application program in the following format:

```
$$^%MPIECE(str,expr1,expr2)
```

If expr1 and expr2 are not specified, %MPIECE assumes expr1 to be one or more consecutive occurrences of whitespaces and expr2 to be one space.

%MPIECE removes all leading occurrences of expr1 from the result.

Utility Labels

$$SPLIT^%MPIECE(str,expr1): Invokes %MPIECE as an extrinsic function that returns an alias local array of string divided into pieces by expr1. If expr1 is not specified, MPIECE assumes expr1 to be one or more consecutive occurrences of whitespaces.

Example:

```
GTM>set strToSplit=" please split this string into six"$char(10)GTM>set piecestring=$$^%MPIECE(strToSplit,"
 ","|") zwrite strToSplit,piecestring write $length(piecestring,"|")$char(10)strToSplit="
 please split this string into six"$char(10)piecestring="please|split|this|string|into|
six$char(10)6$char(10)$char(10)GTM>set *fields=$$SPLIT^%MPIECE(strToSplit) zwrite fields
$char(10)fields(1)="please"$char(10)fields(2)="split"$char(10)fields(3)="this"$char(10)fields(4)="string"$char(10)
```

3. **%DSEWRAP**

The %DSEWRAP utility provides a programmatic interface that drives DSE either through a PIPE device or through generated command files. The current implementation only provides access to dumping the database file header.

Utility Labels

DUMP^%DSEWRAP(regions,.fdump,"fileheader","all") : Retrieve and parse the result of the DSE's DUMP -FILEHEADER -ALL command into the second parameter (passed by reference) for the regions contained in the local variable 'regions'. If invoked as an extrinsic function, %DSEWRAP returns the status of DUMP -FILEHEADER -ALL command.

The first parameter 'regions' can be undefined, "", "*" or "all" to mean all available regions.

The second parameter is a required passed-by-reference variable that the caller uses to retrieve data.

The third optional parameter defaults to DUMP -FILEHEADER. Using any other command dump command has not been tested.

The fourth optional parameter indicates the level of detail, -ALL, for the DUMP -FILEHEADER command. Fore more information on other -FILEHEADER qualifiers, please refer to the DSE chapter in the Administration and Operations Guide.

The format of the output array is fdump(<REGION NAME>,<FIELD NAME>). In the event of a field collision, dump^%DSEWRAP avoids overwriting existing data by creating number descendants.

The default $ETRAP handler for %DSEWRAP terminates the application if it detects an error during input validation. Application developers must define $ETRAP prior to calling %DSEWRAP.

Example:

```
$gtm -run ^%XCMD 'do dump^%DSEWRAP("DEFAULT",.dsefields,"","all") zwrite dsefield'
```

4. **LOOP%XCMD Utility Label**

```
LOOP^%XCMD [--before=/<XECUTE_code>/] --xec=/<XECUTE_code>/ [--after=/<XECUTE_code>/]
```

LOOP^%XCMD: XECUTEs the arguments specified in --xec=/arg1/ as GT.M code for each line of standard input that it reads. Each line is stored in the variable %l. It returns any error status (truncated to a single byte on UNIX) generated by that code.

--before=/arg0/ specifies the GT.M code that LOOP^%XCMD must XECUTE before executing --xec.

--after=/arg2/ specifies the GT.M code that LOOP^%XCMD must XECUTE after executing the last --xec.

For all qualifiers, always wrap GT.M code specified two forward slashes (/) to denote the start and end of the GT.M code. FIS strongly recommends enclosing the GT.M code within single quotation marks to prevent inappropriate expansion by the shell. LOOP^%XCMD's command line parser ignores these forward slashes.

Example:

```
$ ps -ef | $gtm_exe/mumps -run LOOP^%XCMD --before='/set user=$ztrnlnm("USER") write "Number of processes
 owned by ",user," : "/' --xec='/if %l[user,$increment(x)/' --after='/write x,\!/' $char(10) Number of
 processed owned by jdoe: 5$char(10)$$char(10)$char(10)$ cat somefile.txt | $gtm_exe/mumps -run LOOP^%XCMD
 --before='/write "Total number of lines : "/' --xec='/set total=$increment(x)/' --after='/write total,
\!/'$char(10)$char(10)Total number of lines: 9$char(10)$
```

(GTM-7160)

- The LOCK command now implements fairer access when multiple processes need the same lock resource. Previously, when multiple processes sought to get the same lock resource, there was no attempt to ensure fairness. Note that although the technique attempts to be fair, it does not guarantee that no processes will "jump the queue" especially when there is heavy contention for the same lock resource. Applications we expect to be typical should experience no change in overall behavior. (GTM-7254)

- The GT.M compiler, invoked with the MUMPS shell command and the GT.M ZCOMPILE command, returns a status of 1 after any error in compilation. Previously the compiler returned 255, which caused some shell services such as xargs to exit prematurely. Scripts or programs explicitly checking for 255 (rather than non-zero) must be revised. The workaround was to wrap the MUMPS command with a script that modified the return before returning to any utility that was sensitive to a

return status of 255. Note that unless invoked with the NOIGNORE qualifier, unless there are more than 150 compilation errors in a source file, the GT.M compiler still creates an object file with the compilation errors embedded in the code paths where they appeared.[UNIX](GTM-7281)

- GT.M properly handles TPFAIL errors inside a trigger by raising an appropriate error. Previously, under certain rare conditions, a TPFAIL in a trigger could lead to a GTMASSERT. Dealing with the implication of this issue led to changes in GT.M behavior under the following circumstances:

  - When set to a null or non-null string, the gtm_etrap environment variable provides an initial value for $ETRAP, overriding the default value of "B" for $ZTRAP. The GT.M install process now sets a default $ETRAP value of "Write:(0=$STACK) ""Error occurred: "",$ZStatus,!" in the gtmprofile file, which you can customize to suit your needs.

  - An error in a $ZTRAP, $ETRAP, device handler, ZSTEP or ZBREAK action string no longer writes error messages to stderr unless unwinding to direct mode. Note that for an interactive session stderr maps to the terminal/console session unless otherwise redirected. If there is no higher level error trap (the default $ZTRAP, set from $gtm_etrap, or by application code), an error causes a silent process exit. Note that the default $ZTRAP for GT.M in the base frame is "B", which takes the process to direct mode, which may be suitable for development, but not for production environments. Run-time compilation errors from indirection or XECUTEs maintain $STATUS and $ZSTATUS related information and cause nested error handling but do not provide information on the location of any error within the code fragment. Previously, GT.M wrote an ERRWxxx error to stderr followed by the error that occurred.

  - A TP RESTART, either implicit or explicit, while handling another error raises a TPRESTNESTERR error and engages nested error handling, which unstacks M virtual machine frames back to where the prior error occurred, unstacks that frame and rethrows the error.

  Although we do not expect these changes to have any impact on a production environment with appropriate error handling, you may be required to modify your debugging techniques for problems in indirection and XECUTE code. [UNIX] (GTM-7355)

- OPEN of a DECnet task as a GT.M I/O device releases all extra channels associated with the device. Previously, an extra channel remained open for each GT.M OPEN of a DECnet task until the process terminated. [OpenVMS] (GTM-7387)

- OPEN on a directory now yields a GTMEISDIR error in both READONLY and NOREADONLY modes along with the directory name which failed to open. UNIX directories contain metadata that is only available to the file system. Previously OPEN READONLY of a directory appeared to work but any subsequent READ yielded a EISDIR error and OPEN NOREADONLY on directories yielded EISDIR. Note that you can use the ZSEARCH() function to identify files in a directory, and you can call the POSIX stat() function to access metadata. The optional GT.M POSIX plug-in packages the stat() function for easy access from M application code. This change was made to make directory related errors more consistent. [UNIX] (GTM-7395)

- $ZMESSAGE() appropriately handles unused error codes by returning %SYSTEM-E-UNKNOWN. Previously, $ZMESSAGE() could produce a segmentation violation (SIG-11) for some unused error codes. Also more than one unused error codes could be incorrectly mapped into a single known error message.[UNIX] (GTM-7415)

- The $ZWRITE() function takes a single expression argument and returns that expression with the non-graphic characters represented in the $CHAR() format used by the ZWRITE command. Note that the non-graphic characters differ between M mode and UTF-8 mode. This is a new function.(GTM-7475)

- The FOR command correctly handles indirect control variables, as well as control variables whose subscripts contain an extrinsic function. Previously, due to a regression introduced in V54002B while addressing C9L03-003392, certain such cases would result in a segmentation violation (UNIX SIG-11 or VMS ACCVIO). In addition, FOR now safely issues an UNDEF error if the control variable is KILLed during a loop iteration. Previously, with NOUNDEF and an unsubscripted control variable,

this scenario resulted in an access violation on most platforms; on AIX, execution continued with indeterminate behavior. (GTM-7492)

- GT.M provides a C header file gtm_common_defs.h, in the distribution directory, which contains general purpose helper macros for use by external calls and plugins. (GTM-7494)
- GT.M no longer restricts indirection nesting to 32 levels and more robustly handles error conditions, TP restarts, and ZGOTO commands while executing indirect code. Previously, it restricted indirection nesting to 32 levels and the aforementioned occurrences during indirection could cause an access violation (UNIX SIG-11 or VMS ACCVIO) or result in unexpected INDMAXNEST messages. (GTM-7497)
- GT.M appropriately signs the results of exponential expressions. Previously, while assignments to a variable worked correctly, temporary representations were incorrectly negative.(GTM-7501)
- ZSHOW "G" accurately counts the LKS statistic (the number of successfully obtained LOCKs). Previously, LOCK logic incorrectly incremented LKS twice if the LOCK was not immediately available, that is: was held initially by another process. (GTM-7510)
- The gtmxc_types.h highlights the preferred types (gtm_xxx_t) and explicitly marks the old types (xc_xxx_t) as deprecated. It also includes definitions for the following entry points exported from libgtmshr:
  - void gtm_hiber_start(gtm_uint_t mssleep);
  - void gtm_hiber_start_wait_any(gtm_uint_t mssleep);
  - void gtm_start_timer(gtm_tid_t tid, gtm_int_t time_to_expir, void (*handler)(), gtm_int_t hdata_len, void *hdata);
  - void gtm_cancel_timer(gtm_tid_t tid);

  Where:

  - mssleep - milliseconds to sleep
  - tid - unique timer id value
  - time_to_expir - milliseconds until timer drives given handler
  - handler - function pointer to handler to be driven
  - hdata_len - 0 or length of data to pass to handler as a parameter
  - hdata - NULL or address of data to pass to handler as a parameter

  gtm_hiber_start() always sleeps until the time expires; gtm_hiber_start_wait_any() sleeps until the time expires or an interrupt by any signal (including another timer). gtm_start_timer() starts a timer but returns immediately (no sleeping) and drives the given handler when time expires unless the timer is canceled [UNIX](GTM-7515)
- $STACK(lvl) where lvl is greater than zero and less than or equal to $STACK(-1), reports "TRIGGER" for a stack level invoked by a trigger (UNIX only). Previously, $STACK() reported "DO" for this invocation. In addition, $STACK(lvl) reports "ZINTR" for a stack level invoked by MUPIP INTRPT. Previous, GT.M reported "XECUTE" for this invocation.(GTM-7516)
- $GET(), $NAME() and $ORDER() with an indirect first argument and a side effect expression for a subsequent argument, evaluate left-to right. Previously while the evaluation of the indirect glvn occurred before the subsequent argument, the evaluation of its result (the actual first argument) evaluated after the subsequent argument.(GTM-7535)

- When the target of a SET $EXTRACT()/$PIECE()/$ZEXTRACT()/$ZPIECE() command is an indirect subscripted glvn, SET evaluates side effect expressions only once. Previously, SET evaluated the subscripts twice, which inappropriately doubled any side effects.(GTM-7536)

- $ORDER() in the reverse direction and $ZPREVIOUS() return the correct result. Previously, if any global directory name-spaces shared a common prefix and if either of these commands was specified with an unsubscripted global name, an extension referred to as "name-level $ORDER()", the result could have been a global in an incorrect region. (GTM-7541)

- $ORDER() and $ZPREVIOUS() in which the first argument is an extended global reference work safely. Previously, if a GT.M process used either function in this manner, without having already done a non-extended global reference, an access violation (UNIX SIG-11 or VMS ACCVIO) could have occurred. (GTM-7544)

- A TP RESTART, either implicit or explicit, while executing $ZINTERRUPT in response to an interrupt (i.e., $ZININTERRUPT is 1), and while error processing is in effect (i.e, $ECODE'=""), raises a TPRESTNESTERR error and engages nested error handling, which unstacks M virtual machine frames back to where the incompletely handled error occurred, unstacks that frame and rethrows the error. Since GT.M version V5.4-000, GT.M has saved and restored error context ($ECODE, $STACK, etc.) over the execution of a $ZInterrupt invocation, but the RESTART exposes the underlying error. Previously, this situation caused a GTMASSERT.[UNIX](GTM-7550)

- GT.M handles M-locks inside TP transactions correctly. Previously, if an M-lock was used in a TP transaction and the lock name mapped to a database file where no other database references had occurred in that TP transaction, a TP RESTART could in rare cases cause the TP transaction to resume processing from the M line where the retry occurred instead of from where the TSTART happened. This in turn could result in application level database inconsistency, TPFAIL errors in GT.M and in rare cases database damage. A workaround for this was to ensure every lock usage inside a TP transaction is preceded by a global reference (e.g. $GET(), $DATA()) that maps to the exact same region. Note that FIS recommends against the use of LOCKs in TP transactions. (GTM-7552)

- GT.M recognizes >= as the syntactic equivalent of '< and <= as the syntactic equivalent of '>. Previously GT.M did not accept these non-standard binary operators. (GTM-7556)

Utilities-MUPIP

- MUPIP EXTRACT can redirect the database extract to the standard output stream if -STDOUT is specified. Also, MUPIP EXTRACT now issues an EXTRFILEXISTS error if the output file already exists and an EXTRACTFILERR if there is a problem (such as authorization) opening the output. Previously these errors did not have documented mnemonics. [UNIX] (GTM-4115)

- The GT.M replication connection between a Source Server and a Receiver Server and the GT.CM connection between a Client and a Server protect against communication issues when one side of the connection drops in a way that prevents the communication stack on the other side from receiving the dropped notification. When either side ceases to receive a heartbeat, it terminates the connection and then attempts to restore communications; the Receiver Server reverts to listening at its TCP port, and the Source Server makes an attempt to connect to the Receiver Server. Previously, in some cases, an abnormal shutdown of a GT.M replication or a GT.CM connection required operational recycling either or both sides of the connection. [UNIX] (GTM-7168)

- MUPIP JOURNAL -RECOVER -FORWARD works correctly for database files with the MM access method. Previously, it terminated with a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO) if the recovery caused a database file extension which was particularly common if -REDIRECT was also used. [UNIX] (GTM-7241)

- The new MUPIP SIZE command estimates and reports the size of global variables using a format that is similar to the one that appears at the end of the MUPIP INTEG -FULL report. In comparison with MUPIP INTEG -FAST -FULL, MUPIP SIZE provides the option of choosing any one of the three estimation techniques to estimate the size of global variables in a database file. These techniques vary in measurement speed and estimate accuracy. The format of the MUPIP SIZE command is:

```
MUPIP SI[ZE] [-h[euristic]=estimation_technique] [-s[elect]= global-name-list] [-r[egion]=region-list]
```

The optional qualifiers of MUPIP SIZE are:

-Heuristic=estimation_technique

Specifies the estimation technique that MUPIP SIZE should use to estimate the size of global variables. The format of the -HEURISTIC qualifier is:

```
-h[euristic]=sc[an][,level=<lvl>] | a[rsample][,samples=<smpls>] | i[mpsample][,samples=<smpls>]
```

- smpls is the number of samples and must be greater than zero (0)
- lvl is a positive or negative tree level designation and -(level of the root block) <= lvl <= (level of the root block)

estimation-technique is one of the following:

- *scan,level=<lvl>*

  Traverses the global variable tree and counts the actual number of records and blocks at levels from the root down to the level specified by lvl (default is 0, the data blocks). If the given level is non-negative, it is the lowest block level of the global for which the count is requested. So, 0 means all blocks, 1 means all index blocks, 2 means all index blocks of level 2 and above, and so on. SCAN counts a negative level from the root of the global tree where -1 means children of the root.

- *arsample,samples=<smpls>*

Uses acceptance/rejection sampling of random tree traversals to estimate the number of blocks at each level. It continues until the specified number of samples (default is 1,000) is accepted.

- *impsample,samples=<smpls>*

  Uses importance sampling of random tree traversals to weight each sample of the specified number of samples (default is 1,000) in order to estimate size of the tree at each level.

- If -HEURISTIC is not specified, MUPIP SIZE uses the ARSAMPLE,SAMPLE=1000 estimation technique.

Important

For large databases, MUPIP SIZE is faster than MUPIP INTEG -FAST -FULL. IMPSAMPLE is expected to be the fastest estimation technique, followed by ARSAMPLE and then SCAN.

In terms of accuracy, MUPIP INTEG -FAST -FULL is the most accurate.

-Select

Specifies the global variables on which MUPIP SIZE runs. If -SELECT is not specified, MUPIP SIZE selects all global variables.

The format of the SELECT qualifier is:

```
-s[elect]=global-name-list
```

global-name-list can be:

- A comma separated list of global variables.
- A range of global variables denoted by start:end syntax. For example, -select="g1:g4".
- A global variable with wildcards, for example, "g*" (the name must be escaped to avoid shell filename expansion)
- "*" to select all global variables.

-Region

Specifies the region on which MUPIP SIZE runs. If REGION is not specified, MUPIP SIZE selects all regions. The format of the REGION qualifier is:

```
-R[EGION]=region-list
```

Examples:

```
$ mupip size -heuristic="impsample,samples=2000" -select="y*" -region="AREG"
```

This example estimates the size of all global variable starting with "y". It uses importance sampling with 2000 samples on the region AREG.

```
$ mupip size -heuristic="scan,level=-1"
```

This example counts the number of blocks and records at 1 level below the root of the database tree.

```
$ mupip size -heuristic="arsample" -select="g1:g3"
```

This example estimates the size of global variables g1, g2 and g3 using accept/reject sampling with the default number of samples regardless of the region in which they reside.

Note

Apart from randomness caused by sampling heuristics, MUPIP SIZE also has randomness from concurrent updates because it does not use the snapshot technique that MUPIP INTEG uses.

[UNIX] (GTM-7292)

- The Receiver Server and Update Process now record the IPC identifiers associated with the Journal Pool and Receive Pool in the respective log files. Previously, no such information was logged by the Receiver Server and Update Process. [UNIX] (GTM-7356)
- MUPIP INTEG now exits with a MUNOTALLINTEG warning message if it did not find any integrity errors in the regions it checked, but could not check on one or more of the requested regions. For example, MUPIP cannot INTEG regions that are frozen, have an endian mismatch between the database and the machine (DBENDIAN message), have database format too old to support snapshot (SSV4NOALLOW message), etc. Previously, MUPIP INTEG exited with INTEGERRS in such situation, which was misleading as there were no errors found in the processed regions. (GTM-7427)
- MUPIP BACKUP displays GTM-E-BKUPRUNNING message if started when there is an already running BACKUP. Previously, it terminated with a less helpful and undocumented message. (GTM-7432)
- MUPIP REORG and the snapshot mechanism work to minimize the impact of a MUPIP INTEG -FAST on other processes. Previously GTM-7194 in V6.0-000 eliminated the writing of some, but not all, blocks that are not necessary for an INTEG -FAST. [UNIX] (GTM-7454)
- Online REORG works safely again on OpenVMS. Previously, due to a regression introduced in V6.0-000 while addressing GTM-6341, a concurrency issue could cause memory corruption and result in an OPCDEC error. [OpenVMS] (GTM-7455)
- The Receiver Server, on receiving MUPIP STOP after abnormal termination of the Update Server, resets IPC fields in the replication instance file header. Previously, in such cases, the Receiver Server removed IPC resources but did not reset them in the instance file header causing future startup commands to fail with REPLREQROLLBACK errors. The workaround was to do a MUPIP RUNDOWN -REG "*" before restarting the replication servers. [UNIX] (GTM-7458)
- MUPIP JOURNAL -ROLLBACK and MUPIP RUNDOWN support Instance Freeze configuration by opening the Replication Journal Pool if it already exists. Previously an error during a JOURNAL ROLLBACK or RUNDOWN did not set or respect Instance Freeze.[UNIX](GTM-7461)
- When run concurrently with MUPIP REORG ONLINE TRUNCATE, MUPIP BACKUP ONLINE generates correct backup files. Previously MUPIP BACKUP ONLINE could produce incomplete backup files. [UNIX](GTM-7476)
- When MUPIP JOURNAL EXTRACT, with neither the RECOVER nor ROLLBACK qualifier, needs to access the database file to determine collation information to use for the extracted data, a positive integer value of the environment variable gtm_extract_nocol instructs the MUPIP process to use the default collation if it is not able to read the database file. For example, the database file may be inaccessible, or the replication instance may be frozen with a critical section required for the access held by another process. In such a situation, the MUPIP process issues the DBCOLLREQ warning and proceeds with the extract using the default collation. Note that if default collation is used for a database with custom collation, the subscripts reported by MUPIP JOURNAL EXTRACT are those stored in the database, which may differ from those read

and written by application programs. Furthermore, the BG access method now always reserves 32 global buffers for read-only use, which ensures that non-dirty global buffers are always available. If gtm_extract_nocol is not set, or set to a value other than a positive integer, MUPIP JOURNAL EXTRACT exits with SETEXTRENV error if it encounters such a situation. Previously, MUPIP JOURNAL EXTRACT would hang if it encountered a frozen instance, or silently use default collation if it was unable to access the database file. [UNIX] (GTM-7478)

- The Source and Receiver Server shutdown processing in response to MUPIP REPLICATE commands appropriately manage shared memory and semaphores related to instance file and Journal/Receive Pool. Also, the REPLREQROLLBACK error message now provides additional detail. Previously, the server shutdown processing left the IPC resources temporarily unattached, which could allow a concurrent argument-less MUPIP RUNDOWN to treat them as abandoned and remove them, which, in turn, caused the in-progress shutdown command to exit with ENO22, and subsequent startup commands to exit with REPLREQROLLBACK. The workaround was to do a MUPIP RUNDOWN -REGION "*". [UNIX] (GTM-7485)

- Argumentless MUPIP RUNDOWN command issues MUJPOOLRNDWNFL and MURPOOLRNDWNFL messages with correct shared memory ID. Previously, it issued these messages with incorrect shared memory IDs and attempted to remove such memory segments if no process was attached to those segments which, in rare cases, could cause future replication startup commands to exit with REPLREQROLLBACK errors. [UNIX] (GTM-7486)

- The update process now handles Instance Freeze conditions while shutting down. In GT.M V6.0-000, the update process closed the journal pool early in its shutdown logic, which prevented it from handling custom errors and frozen instances appropriately, and could occasionally produce GTMASSERT failures. [UNIX] (GTM-7502)

- The update process operates correctly even in cases where it encounters an Instance Freeze. Previously, when an update process encountered an Instance Freeze, it would terminate abnormally with a GTMASSERT error after the freeze cleared. [UNIX] (GTM-7526)

- The source server, recover, and rollback processes issue a JNLFILRDOPN message when unable to open a journal file for read. Previously they issued a JNLFILOPN message. The new message allows for distinction between read-write and read-only use of the journal file, particularly in a gtm_custom_errors file. (GTM-7530)

- MUPIP no longer terminates with a segmentation fault (SIG-11) when performing a checkhealth while the replication instance is frozen, DSKNOSPCAVAIL is in the custom errors file, and no other processes are attached to the journal pool. [UNIX] (GTM-7538)

- The Source Server exits with one or more JNLFILOPN messages followed by a SEQNUMSEARCHTIMEOUT error after six failed attempts to access journal files in order to read a record needed to satisfy the corresponding Receiver Server. The JNLFILOPN messages contain information on the journal files and regions - the state of each journal file, the last sequence number read by the Source Server, etc. The first five failures generate REPL_WARN messages. Previously, the source server generated REPL_WARN messages every 10 seconds indefinitely and provided no information on which files were associated with the failure. [UNIX](GTM-7566)

Utilities-Other Than MUPIP

- The README, README.txt, and COPYING files in all GT.M distribution kits are now read-only. Previously, they had the write and/or execute permissions set. (GTM-7374)
- LKE SHOW -MEM now displays LOCKSPACEINFO message. Previously this message was only displayed after LOCKSPACEFULL errors. (GTM-7553)
- DBCERTIFY CERTIFY correctly handles a very rare case where the output file from dbcertify scan contains a global variable tree root block whose rightmost child tree path contains a block with just a *key record. Previously, this scenario failed with a DBCINTEGERR error. (GTM-7559)

Error and Other Messages

BKUPRUNNING

BKUPRUNNING, Process dddd is currently backing up region xxxx. Cannot start another backup.

MUPIP Error: MUPIP BACKUP -ONLINE only supports one backup at a time and this error indicates an attempt to start one before a previously started backup finished.

Action: Cancel the running BACKUP or reschedule this BACKUP to a time after the running BACKUP completes.

CRYPTBADCONFIG

CRYPTBADCONFIG, Could not retrieve data from encrypted file ffff due to bad encryption configuration. eeee

Runtime error: The error occurs when a GT.M utility program starts with a bad encryption configuration (like a bad password) and attempts to read a block corresponding to file ffff (either from memory or disk).

Action: Look at accompanying message (or prior messages related to encryption) for more details on what encryption configuration parameter is incorrect.

CRYPTDLNOOPEN Δ

Previous: *CRYPTDLNOOPEN, Error loading encryption library. eeee*

Revised: *CRYPTDLNOOPEN, Failed to load encryption library while opening encrypted file ffff. eeee*

CRYPTDLNOOPEN2

CRYPTDLNOOPEN, Failed to load encryption library dddd. Eeee

Run Time Error: GT.M failed to load the gtmcrypt plug-in or one of its related libraries during GT.M startup.

Action: Refer to the accompanying details (eeee). Verify that the gtmcrypt plug-in and related libraries are properly installed and the LD_LIBRARY_PATH and LIBPATH environment variables are properly set.

CRYPTHASHGENFAILED Δ

Previous: *CRYPTHASHGENFAILED, Error generating encryption hash. eeee*

Revised: *CRYPTHASHGENFAILED, Failed to generate cryptographic hash for symmetric key corresponding to file ffff. eeee*

CRYPTINIT Δ

Previous: *CRYPTINIT, Error initializing encryption library. eeee*

Revised: *CRYPTINIT, Failed to initialize encryption library while opening encrypted file ffff. eeee*

CRYPTINIT2

CRYPTINIT2, Failed to initialize encryption library during GT.M startup. eeee

Run Time Error: The gtmcrypt plug-in reports it is unable to initialize one or more of its related libraries during GT.M startup.

Action: Examine the detailed message (eeee) from the plug-in and take appropriate action.

CRYPTKEYFETCHFAILED Δ

Previous: *CRYPTKEYFETCHFAILED, Cannot obtain encryption key for ffff. eeee*

Revised: *CRYPTKEYFETCHFAILED, Failed to retrieve encryption key corresponding to file ffff. eeee*

CRYPTKEYFETCHFAILEDNF ⛔

CRYPTKEYFETCHFAILEDNF,Cannot obtain encryption key. xxxx

Obsolete Error: No longer issued by GT.M.

CRYPTINIT Δ

Previous: *CRYPTOPFAILED, Encrypt/Decrypt operation failed. eeee*

Revised: *CRYPTOPFAILED, Encrypt/Decrypt operation failed for file ffff. eeee*

DBCOLLREQ

DBCOLLREQ, JOURNAL EXTRACT proceeding without collation information for globals in database. eeeee fffff .

MUPIP Warning: This is MUPIP JOURNAL EXTRACT Warning. This indicates that MUPIP process uses the default collation as it is not able to read the database file fffff because of eeeee

Action: Be aware that if the EXTRACT contains variables with alternative collation that this extract will represent them as stored rather than as used by the application. Attempting to LOAD such an EXTRACT will produce incorrect results.

GTMEISDIR

GTMEISDIR, dddd : Is a directory

Runtime Error: The file dddd opened for reading is a directory. Directories cannot be opened for reading.

Action: Check the argument to the OPEN for the appropriate file and its path.

JNLCNTRL Δ

Previous: *Journal control unsynchronized. Journaling closed for xxxx.*

Revised: *Journal control unsynchronized for ffff.*

JNLEXTEND Δ

JNLEXTEND, Journal file extension error for file xxxx.

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

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

JNLFILRDOPN

JNLFILRDOPN, Error opening journal file xxxx for read for database file yyyy

MUPIP Error: This indicates that GT.M was unable to open journal file xxxx in read-only mode for the specified database file. The Source Server exits with a JNLFILRDOPN message after six failed attempts to open journal files.

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

MUNOTALLINTEG

MUNOTALLINTEG, At least one region skipped. See the earlier messages

MUPIP Warning: The INTEG report is incomplete because MUPIP could not access all of the selected regions.

Action: If appropriate, correct the issue(s) that caused INTEG to skip one or more regions

MUSIZEFAIL

MUSIZEFAIL: MUPIP SIZE : failed. Failure code: xxxx.

MUPIP Error: This error indicates that *MUPIP SIZE* command encountered a database error with failure code xxxx.

Action: Report this error to the group responsible for database integrity within your organization.

MUSIZEINVARG

MUSIZEINVARG, MUPIP SIZE : Invalid parameter value for: xxxx

MUPIP Error: This indicates that *MUPIP SIZE* encountered a qualifier or parameter xxxx with an invalid value

Action: Review the proper syntax for *MUPIP SIZE*. Refer to the Administration and Operations Guide or the online help for the *MUPIP SIZE command.*

NOTALLDBRNDWN

NOTALLDBRNDWN, Not all regions were successfully rundown

Run time error message: This message indicates an error while running down the database. It could be caused by various conditions such as running out of disk space or IO error.

Action: Look at the previous error messages to identify the cause of this error

REQROLLBACK

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

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

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

SEQNUMSEARCHTIMEOUT

SEQNUMSEARCHTIMEOUT, Timed out trying to find sequence number ssss in Journal File(s). See above messages for details. Source server exiting

MUPIP Error: The Source Server was unable to access a journal record corresponding to sequence number ssss as requested by the Replicating instance.

Action: Check to see that the journal chain is intact and all the file in the Source Server has authorization to all the files in the chain. Review the accompanying message for more information.

SETEXTRENV

SETEXTRENV, Database files are missing or Instance is frozen; supply the database files, wait for the freeze to lift or define gtm_extract_nocol to extract possibly incorrect collation

MUPIP Error: It indicates that gtm_extract_nocol environment variable needs to be defined to run MUPIP JOURNAL EXTRACT completion if Instance is Frozen or Database files are missing.

Action: If the you know there are no variables with alternate collation or if the EXTRACT is for analysis rather than a LOAD, define gtm_extract_nocol to a positive value and reissue the command. Otherwise correct the condition before reissuing the EXTRACT.

SIDEEFFECTEVAL

SIDEEFFECTEVAL, Extrinsic ($$), External call ($&) or $INCREMENT() with potential side effects in actuallist, function arguments, non-Boolean binary operands or subscripts

Compile Time Warning: A side effect expression appeared to the right of a global or local variable (glvn) in an order within an outer expression where the side effect might modify the glvn. Setting the gtm_side_effects environment variable in UNIX or logical name in OpenVMS to 2 (two) activates this check.

Action: Analyze the effect(s) of the side effect expressions, which are $INCREMENT(). extrinsics ($$), external calls ($& or $ZCALL()) as to whether they modify a glvn earlier in the expression. If they do, the setting of gtm_side_effects modifies the behavior and you either need to modify the code to eliminate the side effect interaction or be sure to select the behavior you desire.

SPCLZMSG

SPCLZMSG, The following error message cannot be driven through ZMESSAGE

Run time error: The specified error code is not allowed to be driven through *ZMESSAGE*

Action: Make sure ZMESSAGE is not driving any prohibited error message.

TPRESTNESTERR

TPRESTNESTERR, TP restart signaled while handing error - treated as nested error - Use TROLLBACK in error handler to avoid this

Runtime error: GT.M does not allow a TP restart to occur either explicitly or implicitly while doing error processing ($ECODE is not NULL). Note that if a $ZINTERRUPT interrupts an error handler, this restriction then extends to the $ZInterrupt code even though $ECODE is temporarily nullified for the duration of the $ZInterrupt.

Action: Doing a TROLLBACK in the error handler before attempting to set any globals avoids this error.