V6.0-003

GT.M

V6.0-003 Release Notes

Contact Information

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

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

Legal Notice

Revision History		
Revision 1.1	28 January 2014	Added GTM-7811.
Revision 1.0	08 August 2013	V6.0-003 - First published version.

Table of Contents

V6.0-003

Overview

In V6.0-003, GT.M:

- Adds IPv6 to the previous support of IPv4 for TCP connections. SOCKET devices, replication, inbound connections managed by Internet superservers such as inetd, and the GNP protocol - all GT.M uses of TCP except the deprecated TCP devices - use TCP over IPv4 and IPv6. If your DNS provides both IPv4 and IPv6 addresses for the same name, you may need to add an environment variable to ensure that V6.0-003 communicates (for example, for replication) with older GT.M releases that do not support IPv6. Although V6.0-003 is a production release, FIS considers the functionality to support IPv6 to be field test grade functionality in a production release. Please review the description of GTM-6707 below.
- Maintains $KEY for terminal operation and introduces the [NO]EMPT[ERM] device parameter for terminal devices that allows an "erase" character to terminate a READ when there are no characters in the input buffer.
- Improves database throughput on AIX and Solaris.
- Improves critical section management.
- Improves deadlock detection.
- Changes the time stamp used for prior generation journal files to make operational management easier.

There are numerous smaller enhancements, performance enhancements, robustness improvements and bug fixes, described below.

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	

	UNIX	OpenVMS
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]
	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, GNU/Linux, and Solaris.

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.	-updok

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

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

denotes a new feature that requires updating the manuals.

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

denotes deprecated error messages.

denotes revised error messages.

Platforms

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

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

As of the publication date, FIS supports this release on the following hardware and operating system versions. Contact FIS for a current list of supported platforms.

Platform	Supported Versions	Notes
Hewlett-Packard Integrity IA64	11V3 (11.31)	-

Platform	Supported Versions	Notes
HP-UX		
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. 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.**
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):**

Platform	Supported Versions	Notes
		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 V6.0-003 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.0-003_arch (for example, /usr/lib/fis-gtm/V6.0-003_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V6.0-003_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 V6.0-003

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 V6.0-003 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 V6.0-003 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 V6.0-003.
- 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 V6.0-003.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

Stage 2: Database Files Upgrade

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

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

A V6 database supports global variable nodes up to 1 MiB and is not backward compatible. Use MUPIP DOWNGRADE -VERSION=V5 to downgrade a V6 database back to V5 format provided it meets the database downgrade requirements. For more information on downgrading a database, refer to Downgrading to V5 or V4.

Important

A V5 database that has been automatically upgraded to V6 can perform all GT.M V6.0-003 operations. However, that database can only grow to the maximum size of the version in which it was originally created. If the database is V5.0-000 through V5.3-003, the maximum size is 128Mi blocks. If the database is V5.4-000 through V5.5-000, the maximum size is 224Mi blocks. Only a database created with V6.0-000 or above (with a V6 MUPIP CREATE) can have a maximum database size of 992Mi blocks.

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

- Execute the MUPIP REORG -UPGRADE command again, or
- Execute the DSE CHANGE -FILEHEADER -FULLY_UPGRADED=1 command to stop the warnings.

Caution

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

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

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

For additional upgrade considerations, refer to Database Compatibility Notes.

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

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

- Upgrade your database files using in-place or traditional database upgrade procedure depending on your situation. For more information on in-place/traditional database upgrade, see Database Migration Technical Bulletin.
- Run the MUPIP REORG -UPGRADE command. This command upgrades all V4 blocks to V5 format.

Note

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

Database Compatibility Notes

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

Stage 3: Replication Instance File Upgrade

V6.0-003 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.0-003 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 V6.0-003, 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 V6.0-003. This is necessary because multi-line XECUTEs for triggers require a different internal storage format for triggers which makes triggers created in V5.4-000/V5.4-000A/V5.4-001 incompatible with V5.4-002/V5.4-002A/V5.4-002B/V5.5-000/V6.0-000/V6.0-001/V6.0-003.

To extract and reapply the trigger definitions on V6.0-003 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 V6.0-003 to reload your trigger definitions.

To extract and reload trigger definitions on a V6.0-003 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 V6.0-003.
6. Run **mumps -run %XCMD 'if $ztrigger("file","trigger_defs.trg")'** to reapply your trigger definitions.
7. Turn replication on.
8. Connect to the originating instance.

Note

Reloading triggers renumbers automatically generated trigger names.

Downgrading to V5 or V4

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

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

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

-VERSION specifies the desired version for the database header.

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

1. The database was created with a major version no greater than the target version.
2. The database does not contain any records that exceed the block size (spanning nodes).

3. The sizes of all the keys in database are less than 256 bytes.

4. There are no keys present in database with size greater than the Maximum-Key-Size specification the database header, that is, Maximum-Key-Size is assured.

5. The maximum Record size is small enough to accommodate key, overhead, and value within a block.

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

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

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

1. In a GT.M V6.0-003 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_V6.0-003_i686, then gtmprofile and gtmcshrc set $gtm_dist to /usr/lib/fis-gtm/gtm_V6.0-003_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

V6.0-003

Fixes and enhancements specific to V6.0-003 are:

Id	Prior Id	Category	Summary
GTM-4828	S9D01-002280	MUMPS	Prevent hangs during process rundown triggered by the delivery of an external signal
GTM-5708	S9F06-002553	MUMPS	The GT.M terminal device derive sets $KEY to the character or escape sequence terminating a READ
GTM-5710	S9F06-002554	MUMPS	Repetitive call-in invocations of gtm_init() no longer cause M stack overflow.
GTM-6707	C9K10-003329	MUMPS	IPv6 Support - Field Test
GTM-7370	-	MUPIP	MUPIP SET -JOURNAL adjusts journal alignsize or allocation only when the operation can complete without any error
GTM-7423	-	MUPIP	MUPIP issues a JNLORDBFLU error on failure to synchronize a journal file or database to disk
GTM-7430	-	MUPIP	MUPIP REORG -SELECT and -EXCLUDE correctly handle long global names
GTM-7500	-	MUMPS	See GTM-6707.
GTM-7519	-	MUPIP	Various MUPIP REORG -TRUNCATE fixes
GTM-7528	-	Other Utilities	Automcatic database initialization retry after bypass
GTM-7614	-	DB	Improved deadlock detection
GTM-7623	-	DB	Issue DBFILERR when appropriate
GTM-7627	-	MUPIP	Switching to new Journal files appends current system time instead of last modified time
GTM-7631	-	MUPIP	Exchange additional state information between Source Server and Receiver Server

Id	Prior Id	Category	Summary
GTM-7647	-	DB	Remove risk of SIG-11 when instance freeze is enabled and database write fails
GTM-7664	-	DB	More efficient transactions on AIX and Solaris
GTM-7675	-	MUMPS	Erase on an empty line terminates a READ and $KEY and $ZB hold the terminator.
GTM-7680	-	MUMPS	GT.M protects process creation from potential deadlock.
GTM-7688	-	MUPIP	Remove risk of a rare SIG-11 in REORG
GTM-7716	-	DB	Instance Freeze avoids possible ENO22 during database initialization
GTM-7731	-	Other Utilities	While printing to the syslog, GTMSECSHR defers the handling of signals that could lead to a process hang
GTM-7732	-	Other Utilities	GDE correctly stores and displays global namespaces containing an asterisk (*)
GTM-7733	-	MUMPS	GT.M tracing more accurately records the system and user CPU time for every profiled M line on Tru64 UNIX
GTM-7738	-	DB	Improved critical section locks
GTM-7747	-	MUMPS	While retrieving information about a file, GT.M defers the handling of signals that could lead to a process hang
GTM-7749	-	MUMPS	Missing return value in non-void M GTMJI call-in label results in default (per type) value returned on the Java side
GTM-7754	-	MUPIP	Replication Source Server performance improvements
GTM-7760	-	DB	SET does not garble values for globals with triggers
GTM-7761	-	MUMPS	Avoid TMPSTOREMAX errors for long concatenations with -DYNAMIC_LITERALS
GTM-7767	-	MUPIP	Receiver Server appropriately manages incoming messages from the Source Server during shutdown and flow control events

Id	Prior Id	Category	Summary
GTM-7770	-	MUMPS	Prevent a job parameter whose length is greater than 127 bytes from causing a SIG-11
GTM-7774	-	MUMPS	Prevent SIG-11 on Solaris when arguments exceed 239
GTM-7777	-	DB	New VIEW keywords: DBFLUSH, DBSYNC, and EPOCH
GTM-7785	-	MUPIP	Update Process does not issue false STRMSEQMISMTCH errors
GTM-7786	-	MUPIP	Transitional Source Server Modes
GTM-7788	-	Other Utilities	The gtmposix plug-in stat() function no longer errors out due to the incorrect call table argument count
GTM-7794	-	Other Utilities	A GTMSECSHR process does not block any interrupt signals regardless of the signal mask of the invoking process
GTM-7811	-	DB	Fix for MUPIP STOP arriving at a TRESTART involving a trigger.
GTM-7815	-	Other Utilities	Installation script correctly initializes the UTF-8 mode GT.M distribution
GTM-7817	-	MUMPS	Extrinsic function SIG-11 fix
GTM-7823	-	MUMPS	Linux core dumps enhanced

M-Database Access

- GT.M does comprehensive deadlock detection for both TP transactions and non-TP mini-transactions. Previously, GT.M's deadlock checking was less complete and might rarely and erroneously issue an inappropriate GTMASSERT; FIS encountered this issue in testing and has not received a customer report of such a GTMASSERT. (GTM-7614)
- GT.M issues a DBFILERR error upon encountering file system problems which prevent normal reading from the database file. Previously, due to changes made with the introduction of MUPIP REORG -TRUNCATE in V5.5-000, GT.M issued a misleading TPFAIL or non-TP failure error with code 'qqqq' in response to such problems. (GTM-7623)
- For a database region for which Instance Freeze is enabled, GT.M and MUPIP processes correctly freeze the instance in the unusual case a file system write to the database fails. Previously, this resulted in a segmentation violation (SIG-11) for the process. [UNIX] (GTM-7647)
- GT.M manages aspects of TP transactions and non-TP mini-transactions in a more efficient way on AIX and Solaris. In database intensive activity, the improvement on AIX could approach 30% and on Solaris, 10%. As database updates are just one part of a typical application, typical improvements in application throughput are likely to be less than these numbers (i.e., your mileage may vary). [UNIX] (GTM-7664)
- An Instance Freeze during database initialization works correctly. Previously, such an event could produce an EN022 error. [UNIX] (GTM-7716)
- GT.M more smoothly handles high contention for a database's principal critical section. Previously, limited benchmarking revealed erratic database access latencies in scenarios involving on the order of one thousand concurrent processes. [x86_64 Linux, Solaris] (GTM-7738)
- SET operations work correctly on global nodes with defined triggers. Previously, if any trigger node had the same unsubscripted ancestor as the destination node, SET sometimes placed a garbled value in the destination. [UNIX] (GTM-7760)
- VIEW accepts DBFLUSH, DBSYNC, and EPOCH. VIEW "DBFLUSH"[:REGION[:N]] writes modified cache blocks to the database file. By default, these command options operate on all regions under the current global directory. N specifies the number of blocks to write; by default, DBFLUSH writes all modified blocks. On UNIX, VIEW "DBSYNC"[:REGION] performs an fsync() system call on the database file. VIEW "EPOCH"[:REGION] flushes the database buffers and, if journaling is enabled, writes an EPOCH record. (GTM-7777) 
- GT.M appropriately handles a MUPIP STOP received by a process within a small window while processing a TRESTART while an performing an update involving a trigger. Previously, such an unlikely circumstance could cause the process to terminate with a segmentation fault (SIG-11). FIS encountered this issue in testing and has not received a customer report of such an event.[UNIX] (GTM-7811)

M-Other Than Database Access

- GT.M avoids certain unusual hangs during process rundown triggered by the delivery of an external signal. Previously, under very rare circumstances a GT.M process could fall into an indefinite sleep while handling a process-terminating signal, such as SIGTERM. [UNIX] (GTM-4828)
- The GT.M terminal device driver sets $KEY to the character or escape sequence terminating a READ, which is the same as the contents of $ZB. Previously, GT.M did not maintain $KEY for terminals. [UNIX](GTM-5708)
- GT.M ignores repetitive call-in invocations of gtm_init(). Note that gtm_init() call is done implicitly, not required, and best avoided. Previously, repeted call-in invocations of gtm_init() could cause M stack overflow[UNIX] (GTM-5710)
- GT.M supports both IPV4 and IPV6 communication for SOCKET devices, replication, backup, and GT.CM. FIS considers IPv6 support in V6.0-003 to be field test grade functionality, and intends to designate IPv6 support as production grade in a future release. Except for IPv6 support, FIS considers V6.0-003 to be a production grade GT.M release.
  - The CONNECT deviceparameter for an OPEN of a SOCKET device, or start of MUPIP REPLICATE -SOURCE , or GT.CM environment variable accepts both IPv4 and IPv6 addresses, optionally encapsulated by square-brackets ([]). For instance, the IP address for the loopback device can be "127.0.0.1", "::1", "[127.0.0.1]", or "[::1]".
  - MUPIP BACKUP and MUPIP RESTORE accept IPv6 host names after the tcp:// prefix, but not numeric IPv6 addresses.
  - When a name is specified, GT.M uses the protocol of the first address returned by DNS that is (a) supported by the operating system, and (b) for which a network interface exists.
  - When a port number is specified with the ZLISTEN deviceparameter, GT.M uses an IPv6 socket that accepts both IPv4 and IPv6 connections.
  - Any socket operation encountering an error retrieving host address information, or name or port information produces a GETADDRINFO or GETNAMEINFO error respectively.
  - When IPv6 sockets are passed via stdin/stdout (e.g., by inetd) to $PRINCIPAL, they are identified as SOCKET devices.

  The addition of IPv6 support can result in a Source Server of a GT.M V6.0-003 instance attempting to make an IPv6 connection to a Receiver Server from an instance running an older version of GT.M, if the same server name is used with both IPv4 and IPv6 instances. For environments where such circumstances arise, and where different server names are not used for IPv4 and IPv6 addresses, the environment variable gtm_ipv4_only can be set to"TRUE", "YES", or a non-zero integer to instruct GT.M that it should only use IPv4. Also, MUPIP REPLICATE -SOURCE -JNLPOOL -SHOW now displays the "Secondary INET Address" field in a human-readable format. Previously MUPIP displayed the address as a number in decimal and hexadecimal. [UNIX] (GTM-7500) (GTM-6707)

- Please refer to GTM-6707. (GTM-7500)
- The OPEN and USE commands support the [NO]EMPT[ERM] deviceparameter for terminal devices. When EMPTERM is specified, and a process executing a READ or READ # command receives an "Erase" character or character sequence with an empty input line, the command terminates and $KEY (and $ZB) hold the terminating character sequence. The default is NOEMPTERM. The gtm_principal_editing environment variable can specify an initial setting of [NO]EMPTERM. The TERMINFO specified by the current value of the TERM environment variable defines capnames values "kbs" and/or "kdch1" with character sequences for "Erase." If "kbs" or "kdch1" are multi-character values, you must also specify the ESCAPE or EDIT deviceparameters for EMPTERM recognition. The ERASE - a special terminal input character - also terminates a READ command with an empty input line and $KEY (and $ZB) hold the ERASE character. You can set the ERASE character to various values using the stty shell command. Typical values of the ERASE character are <CTRL-H> and <CTRL-?> .[UNIX](GTM-7675)
- GT.M protects process creation from potential deadlocks caused by asynchronous signals. GT.M process creation includes: JOB'd processes, PIPE device processes, gtmsecshr processes, and replication server, update and helper processes. Previously, there were very small windows where an asynchronous signal could cause a process creating a new process to hang indefinitely. [UNIX] (GTM-7680)
- The GT.M tracing facility more accurately records the system and user CPU time for every profiled M line. Previously, GT.M tracing did not account for an undocumented anomaly in the system service that provides the times, resulting in occasional erroneous values. [Tru64 UNIX] (GTM-7733)
- While retrieving information about a file, GT.M defers the handling of signals (such as SIGTERM) that could lead to a process hang. This closes a small window of time within which an interrupt to a GT.M process retrieving file meta-data from the operating system could cause it to hang. FIS encountered this issue in testing and has not received a customer report of such a error. [UNIX] (GTM-7747)
- When using GTMJI call-in interface, if an M label declared to have a non-void return type (as defined in the call-in table) does not specify a return value, the invoking Java routine receives a default (false for Boolean, 0 for primitive numeric, and null for non-primitive types) return value. Previously, a missing return value in the M label could potentially result in a segmentation fault or arbitrary value returned to Java. [IBM System p AIX, Sun SPARC Solaris, x86_64 GNU/Linux, x86 GNU/Linux] (GTM-7749)
- Long concatenations compile successfully with -DYNAMIC_LITERALS enabled. Previously, such compilations of concatenations involving 128 or more operands produced TMPSTOREMAX errors. [UNIX] (GTM-7761)
- The JOB command produces a JOBPARTOOLONG error if a jobparameter exceeds 255 characters. Previously, on UNIX jobparameters longer than 127 characters and on OpenVMS parameters longer than 255, except for DEFAULT which failed when longer than 197, caused either an access violation or a GTMASSERT (GTM-7770)
- The maximum number of arguments for functions and concatenate expressions is 253 on all platforms except for Solaris, where it is 239. Previously the documentation incorrectly stated that the

limit was 256 for all platforms, and exceeding 239 arguments on Solaris for these operations produced an access violation (SIG-11) or an inappropriate UNDEF error. (GTM-7774)

- GT.M safely executes extrinsic functions in an edge case where a routine loaded in memory that includes an address that is a multiple of 0x100000000, makes a call to a label within the routine but across that address. Previously, this could result in an access violation (SIG-11). [x86_64 Linux] (GTM-7817)
- GT.M core dumps now include the memory segments needed for prompt problem analysis. Previously core dumps could be missing required memory associated with huge pages, or due to inappropriate user configuration. [Linux] (GTM-7823)

Utilities-MUPIP

- MUPIP SET -JOURNAL adjusts journal alignsize or allocation only when the operation can complete without any error. Previously, MUPIP changed these values before completing its error checking, which could cause subsequent problems for journal operation. (GTM-7370)

- MUPIP issues a JNLORDBFLU error on failure to synchronize a journal file or database to disk. However, MUPIP RUNDOWN -OVERRIDE removes IPC resources, but leaves the database header fields intact for use by a subsequent MUPIP JOURNAL -ROLLBACK, MUPIP JOURNAL -RECOVER, or MUPIP RUNDOWN, depending on journal and/or replication status. Additionally, MUPIP RUNDOWN sends a MURNDWNOVRD informational message to the operator log on every use of the -OVERRIDE qualifier, which ignores potentially hazardous conditions that would otherwise result in guidance to use a MUUSERLBK, MUUSERECOV, or JNLORDBSYNC error; a brief description of the bypassed condition follows the MURNDWNOVRD message. [UNIX](GTM-7423)

- MUPIP REORG safely trims long global names specified with the -SELECT and -EXCLUDE qualifiers to a maximum of 31 characters. Previously, MUPIP REORG produced a fatal GTMASSERT error if given a name longer than the maximum key size in the database file header, and if the name, when trimmed, matched an existing global. Also, MUPIP REORG -RESUME works appropriately after an administrator reduces the maximum key size, where previously it produced an access violation (UNIX SIG-11 or OpenVMS ACCVIO). Additionally, MUPIP EXTRACT prints trimmed global names in RECORDSTAT messages. (GTM-7430)

- GT.M handles a database file extension after a MUPIP REORG -TRUNCATE correctly. Previously, when performing a file extension approximately concurrent with a TRUNCATE, a process might have encountered inappropriate TPFAIL or GVXXXXFAIL errors. FIS encountered this issue in testing and has not received a customer report of such an error. In addition, TRUNCATE no longer prints inappropriate GBLNOEXIST messages for "Global #t" and generally moves fewer root blocks. [UNIX] (GTM-7519)

- To simplify journal file maintenance for system administration, when renaming the current journal file as part of a journal file switch, GT.M uses the current date and time. Previously, GT.M used the date and time of the last update within the journal file. (GTM-7627)

- The Source Server and Receiver Server exchange and record information about their log files and process IDs. In addition, a FETCHRESYNC ROLLBACK, which does not have a separate log file, sends its current working directory to the Source Server log. Previously, the replication protocol did not exchange this state information. [UNIX] (GTM-7631) 

- REORG safely handles a rare concurrency issue associated with BG databases configured to have a small number of global buffers and MM databases of any size. Previously, REORG would produce an access violation (UNIX SIG-11 or OpenVMS ACCVIO) occasionally, while working on blocks within approximately 64KiB of the end of an MM database. The same could happen for BG databases with fewer than 384 global buffers (on 64-bit platforms) or with fewer than 512 (32-bit). (GTM-7688)

- The replication Source Server always operates in the catch-up mode and reacts more quickly to flow control messages from the Receiver Server and to shut down commands. Previously, it could have been slow to recognize Receiver Server requests to suspend sending information in order for the Replicating Instance to have a chance to catch up, and, under some circumstances, it could have been unresponsive to shut down requests. The Source Server also maintains a more even stream by checking the replication journal pool more frequently. In addition, on HP-UX, a Receiver Server reacts more quickly to shut down commands. Previously, under some circumstances, it could have been unresponsive to shut down requests. (GTM-7754)

- The Receiver Server manages incoming messages from the Source Server appropriately during shutdown and flow control events. Previously, in some rare cases, the Receiver Server could terminate with a segmentation violation (SIG-11) or hang indefinitely during these events. [UNIX] (GTM-7767)

- The replication Update Process does not issue false STRMSEQMISMTCH errors. Previously, in the event of an concurrent ONLINE ROLLBACK, Update Process issued false STRMSEQMISMTCH errors. [UNIX] (GTM-7785)

- The "MUPIP REPLICATE -SOURCE -ACTIVATE" command sets the source server to ACTIVE_REQUESTED mode, and the source server subsequently sets the mode to ACTIVE once it completes the transition. Similarly, the "MUPIP REPLICATE -SOURCE -DEACTIVATE" command sets the source server to PASSIVE_REQUESTED mode, and the source server subsequently sets the mode to PASSIVE once it completes the transition. The -ACTIVATE and -DEACTIVATE qualifiers produce an error message if they are specified while the source server is in ACTIVE_REQUESTED or PASSIVE_REQUESTED mode. The *_REQUESTED modes may be reported by the -CHECKHEALTH or -JNLPOOL -SHOW qualifiers. Previously, use of the -ACTIVATE and -DEACTIVATE qualifiers in rapid succession could lead to an inconsistent state in the source server. FIS encountered this issue in testing and has not received a customer report of such an event. (GTM-7786) 

Utilities-Other Than MUPIP

- LKE and DSE automatically retry database initialization if some other process removes the shared memory associated with the database while they are bypassing the normal interlocking of the initialization. If initialization does not succeed due to repeated shared memory removals, DSE/LKE moves on to the next regions' initialization. Previously, these operator utilities terminated with an error (REQROLLBACK/REQRECOV/REQRUNDOWN) when another process removed shared memory while they were trying a non-interlocked database initialization. [UNIX] (GTM-7528)

- While printing to the syslog, GTMSECSHR defers the handling of signals (such as SIGTERM) that could lead to a process hang. Previously, there was a very small chance that a signal could interrupt a GTMSECSHR process in the middle of writing to the syslog, thus causing the process to hang indefinitely. [UNIX] (GTM-7731)

- GDE now correctly stores and displays the namespaces specified as part of the -NAME qualifier. Previously in edge cases where * was specified in global ranges, it was possible for GDE to store the global mapping incorrectly. This meant GDE SHOW -MAPS as well as GDE SHOW -NAME produced incorrect results. An example that did not work before was if the name AG* mapped to region STAR1, the name AGZ* to region STAR2 and the name AGZ to region STAR1 (that is, there was an exception in the AGZ* name specification), GDE incorrectly mapped the name AGZ* to region STAR1. (GTM-7732)

- The gtmposix plug-in stat() function works with releases after V6.0-002. Previous versions of gtmposix contained an incorrect stat() function definition in the external call table, which, if used with current releases, produce an error return even for correct external routine invocations. [UNIX] (GTM-7788)

- A GTMSECSHR process does not block any interrupt signals (such as SIGTERM) regardless of the signal mask of the invoking process, though it ignores signals that are not material to gtmsecshr operation. Previously, a gtmsecshr process inherited the signal mask of the invoking GT.M process and could inappropriately become unresponsive to particular signals. The workaround was to let gtmsecshr timeout. [UNIX] (GTM-7794)

- TThe GT.M installation script "configure", which is in turn invoked by the "gtminstall" script, sets up the gtmsecshr-related files properly for both M and UTF-8 modes of operation. In the first packaging of V6.0-002, the gtminstall script configured gtmsecshr in a way that caused UTF-8 mode processes to attempt to start gtmsecshr in a way that violated the gtmsecshr security checks. A later repackaging of V6.0-002 corrected the issue.[UNIX] (GTM-7815)

Error and Other Messages

GETADDRINFO

Error in getting address info

Runtime error: This message indicates a problem converting a host name to an IP address.

Action: See associated TEXT message for more details. Check host names used for replication, backup, SOCKET devices, or GT.CM.

GETNAMEINFO

Error in getting name info

Runtime error: This message indicates a problem converting an IP address to a readable format.

Action: See associated TEXT message for more details. Report the error to your GT.M support channel.

INSTFRZDEFER

INSTFRZDEFER, Instance Freeze initiated by eeee error on region rrrrr deferred due to critical resource conflict.

Runtime informational message: eeee error encountered on region rrrrr triggered Instance Freeze mechanism to set the freeze, but couldn't do it due to a critical resource conflict. Any process can set the freeze later.

Action: None necessary.

JNLORDBFLU

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

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

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

MURNDWNOVRD

MURNDWNOVRD, OVERRIDE qualifier used with MUPIP RUNDOWN on database file dddd

Runtime informational message: This message notifies such usage of the OVERRIDE qualifier with a MUPIP RUNDOWN command that altered the code flow to bypass an otherwise imminent error.

Action: No action required. This message is primarily to facilitate analysis of database crashes and recovery procedures.

REGOPENFAIL

REGOPENFAIL, Failed to open region rrrr (dddd) due to conflicting database shutdown activity

LKE/DSE Error: Another process or processes repeatedly removed shared memory right after each one of the interlocking bypass. As a result of this, LKE/DSE failed to initialize region rrrr (database for dddd)

Action: Identify the process or processes that causes LKE/DSE to bypass interlocking mechanism by holding semaphore(s). To do that, follow RESRCINTRLCKBYPAS messages. They should tell you the PID of the process holding the semaphore. Check that process to see if it is stuck. If it is not stuck or have already terminated, it is likely that the database is being closed and opened abnormally fast. You may try running LKE/DSE again to achieve successful initialization.

REGOPENRETRY

REGOPENRETRY, Attempt to open region rrrr (dddd) using startup shortcut failed due to conflicting database shutdown. Retrying...

LKE/DSE Information: Another process removed shared memory right after the interlocking bypass, so LKE/DSE could not attach to the shared memory of region rrrr (database dddd). LKE/DSE is repeating database initialization for dddd.

Action: None necessary.

SOCKBIND

SOCKBIND, Error in binding TCP socket

Runtime error: This message indicates a problem binding a socket to a port.

Action: Check the associated ENO message for more details. Report the messages to your GT.M support channel..