GT.M

Release Notes

V7.0-002

Empowering
the Financial World

FIS

Contact Information

GT.M Group
Fidelity National Information Services, Inc.
200 Campus Drive
Collegeville, PA 19426
United States of America

GT.M Support for customers: gtmsupport@fisglobal.com
Automated attendant for 24 hour support: +1 (484) 302-3248
Switchboard: +1 (484) 302-3160
Website: http://fis-gtm.com

Legal Notice

Revision History		
Revision 1.0	28 March 2022	V7.0-002

Table of Contents

V7.0-002

Overview

V7.0-002 provides an option to proactively split blocks to reduce restarts, and an ISV that specifies a limit beyond which a memory request by a process produces a warning.

V7.0-002 also includes other fixes and enhancements including changes aimed at making management of backups and replication more straightforward. For more information, refer to the Change History section.

Items marked with ✓ document new or different capabilities.

Please pay special attention to the items marked with the symbols ✓ as those document items that have a possible impact on existing code, practice or process. Please be sure to recompile all objects to ensure all the updates are in place.

Note

While FIS keeps message IDs and mnemonics quite stable, messages texts change more frequently as we strive to improve them, especially in response to user feedback. Please ensure you review any automated scripting that parses GT.M messages.

Conventions

This document uses the following conventions:

Flag/Qualifiers	-
Program Names or Functions	upper case. For example, MUPIP BACKUP
Examples	lower case. For example: mupip backup -database ACN,HIST /backup
Reference Number	A reference number is used to track software enhancements and support requests. It is enclosed between parentheses ().
Platform Identifier	Where an item affects only specific platforms, the platforms are listed in square brackets, e.g., [AIX]

Note

The term UNIX refers to the general sense of all platforms on which GT.M uses a POSIX API. As of this date, this includes: AIX and GNU/Linux on x86 (32- and 64-bits).

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

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
originating instance or primary instance	-rootprimary	originating instance or originating primary instance.	-updok (recommended)

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
		Within the context of a replication connection between two instances, an originating instance is referred to as source instance or source side. For example, in an B<-A->C replication configuration, A is the source instance for B and C.	-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 adopted IEC standard Prefixes for binary multiples. This document therefore uses prefixes Ki, Mi and Ti (e.g., 1MiB for 1,048,576 bytes). Over time, we'll update all GT.M documentation to this standard.

✓ denotes a new feature that requires updating the manuals.

✓ denotes a new feature or an enhancement that may not be upward compatible and may affect an existing application.

⛔ denotes deprecated messages.

Δ denotes revised messages.

⊕ denotes added messages.

Platforms

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

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

As of the publication date, FIS supports this release on the hardware and operating system versions below. Contact FIS for a current list of Supported platforms. The reference implementation of the encryption plugin has its own additional requirements, should you opt to use it as included with GT.M.

Platform	Supported Versions	Notes
IBM Power Systems AIX	7.1 TL 5, 7.2 TL 3	Only 64-bit versions of AIX with POWER7 as the minimum required CPU architecture level are Supported. While GT.M supports both UTF-8 mode and M mode on this platform, there are problems with the AIX ICU utilities that prevent FIS from testing 4-byte UTF-8 characters as comprehensively on this platform as we do on others. Running GT.M on AIX 7.1 requires APAR IZ87564, a fix for the POW() function, to be applied. To verify that this fix has been installed, execute **instfix -ik IZ87564.** Only the AIX jfs2 filesystem is Supported. Other filesystems, such as jfs1 are Supportable, but not Supported. FIS strongly recommends use of the jfs2 filesystem on AIX; use jfs1 only for existing databases not yet migrated to a jfs2 filesystem.
x86_64 GNU/Linux	Red Hat Enterprise Linux 7.9, 8.5; Ubuntu 18.04 LTS, and 20.04 LTS	To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware. GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6.32 or later), glibc (version 2.12 or later) and ncurses (version 5.7 or later). Due to build optimization and library incompatibilities, GT.M versions older than V6.2-000 are incompatible with glibc 2.24 and up. This incompatibility has not been reported by a customer, but was observed on internal test systems that use the latest Linux software distributions from Fedora (26), Debian (unstable), and Ubuntu (17.10). In internal testing, processes either hung or encountered a segmentation violation (SIG-11) during operation. Customers upgrading to Linux distributions that utilize glibc 2.24+ must upgrade their GT.M version at the same time as or before the OS upgrade. GT.M requires a compatible version of the libtinfo library. On Red Hat, the ncurses-libs and ncurses-compat-libs packages contain the libtinfo library. On Debian/Ubuntu, libtinfo5 and libncurses5 packages contain the libtinfo library. If any of these packages is not already installed on your system, please install using an appropriate package manager. To support the optional WRITE /TLS fifth argument (the ability to provide / override options in the tlsid section of the encryption configuration file), the reference implementation of the encryption plugin requires libconfig 1.4.x. Only the ext4 and xfs filesystems are Supported. Other filesystems are Supportable, but not Supported. Furthermore, if you use the NODEFER_ALLOCATE feature, FIS strongly recommends that you use xfs. If you must use NODEFER_ALLOCATE with ext4, you **must** ensure that your kernel includes commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3). The Red Hat Bugzilla identifier for

Platform	Supported Versions	Notes
		the bug is 1213487. With NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs. Ubuntu 21.10 is Supportable.

Note

x86 GNU/Linux is Supportable but not Supported. The 32-bit version of GT.M runs on either 32- or 64-bit x86 platforms; we expect the x86_64 GNU/Linux version of GT.M to be preferable on 64-bit hardware. Running a 32-bit GT.M on a 64-bit GNU/Linux requires 32-bit libraries to be installed. The CPU must have an instruction set equivalent to 586 (Pentium) or better. If you are using the 32-bit GT.M, please also refer to the notes above on x86_64 GNU/Linux.

Important

GT.M has been Supportable (but not Supported) on x86 GNU/Linux platform since V7.0-000. V7.0-002 is the last GT.M release that is Supportable on this platform. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.

Platform support lifecycle

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

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

32- vs. 64-bit platforms

The same application code runs on both 32-bit and 64-bit platforms; however there are operational differences between them (for example, auto-relink and the ability to use GT.M object code from shared libraries exist only on 64-bit platforms). Please note that:

- You must compile the application code separately for each platform. Even though the M source code is the same, the generated object modules are different - the object code differs between x86 and x86_64.
- Parameter-types that interface GT.M with non-M code using C calling conventions must match the data-types on their target platforms. Mostly, these parameters are for call-ins, external calls, internationalization (collation) and environment translation, and are listed in the tables below. Note that most addresses on 64-bit platforms are 8 bytes long and require 8 byte alignment in structures whereas all addresses on 32-bit platforms are 4 bytes long and require 4-byte alignment in structures.

Call-ins and External Calls

Parameter type	32-Bit	64-bit	Remarks
gtm_long_t	4-byte (32-bit)	8-byte (64-bit)	gtm_long_t is much the same as the C language long type.
gtm_ulong_t	4-byte	8-byte	gtm_ulong_t is much the same as the C language unsigned long type.

Parameter type	32-Bit	64-bit	Remarks
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.

Additional Installation Instructions

To install GT.M, see the "Installing GT.M" section in the GT.M Administration and Operations Guide. For minimal down time, upgrade a current replicating instance and restart replication. Once that replicating instance is current, switch it to become the originating instance. Upgrade the prior originating instance to become a replicating instance, and perform a switchover when you want it to resume an originating primary role.

Caution

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

- FIS strongly recommends installing each version of GT.M in a separate (new) directory, rather than overwriting a previously installed version. If you have a legitimate need to overwrite an existing GT.M installation with a new version, you must first shut down all processes using the old version. FIS suggests installing GT.M V7.0-002 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V7.0-002_arch (for example, /usr/lib/fis-gtm/V7.0-002_x86 on 32-bit Linux systems). A location such as /opt/fis-gtm/V7.0-002_arch would also be appropriate. Note that the ***arch*** suffix is especially important if you plan to install 32- and 64-bit versions of the same release of GT.M on the same system.

- Use the appropriate MUPIP command (e.g. ROLLBACK, RECOVER, RUNDOWN) of the old GT.M version to ensure all database files are cleanly closed.

- Make sure gtmsecshr is not running. If gtmsecshr is running, first stop all GT.M processes including the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP *pid_of_gtmsecshr***.

- Starting with V6.2-000, GT.M no longer supports the use of the deprecated $gtm_dbkeys and the master key file it points to for database encryption. To convert master files to the libconfig format, please click ▼ to download the CONVDBKEYS.m program and follow instructions in the comments near the top of the program file. You can also download CONVDBKEYS.m from http://tinco.pair.com/bhaskar/gtm/doc/articles/downloadables/CONVDBKEYS.m. If you are using $gtm_dbkeys for database encryption, please convert master key files to libconfig format immediately after upgrading to V6.2-000 or later. Also, modify your environment scripts to include the use of gtmcrypt_config environment variable.

Recompile

- Recompile all M and C source files.

Rebuild Shared Libraries or Images

- Rebuild all Shared Libraries after recompiling all M and C source files.

- If your application is not using object code shared using GT.M's auto-relink functionality, please consider using it.

Compiling the Reference Implementation Plugin

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

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

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

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

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

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

3. Follow the instructions in the README.

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

- Define the gtm_dist environment variable to point to the absolute path for the directory where you have GT.M installed
- Copy and paste the commands from the README to compile and install the encryption plugin with the permissions defined at install time

Caution

These are separate steps to compile the encryption plugin for GT.M versions V5.3-004 through V6.3-000 when OpenSSL 1.1 is installed and OpenSSL 1.0.x libraries are still available.

- Download the most recent OpenSSL 1.0.x version
- Compile and install (default installs to /usr/local/ssl)

  ```
  ./config && make install
  ```

- Adjust the configuration : Move the newly installed libraries out of the way

  ```
  mv /usr/local/ssl/lib /usr/local/ssl/lib.donotuse
  ```

- Adjust the configuration : Create another /usr/local/ssl/lib and symlink the existing 1.0.x library into it as the default. This ensures that the encryption plugin is compiled using the compatible OpenSSL 1.0.x library. Adjust the path below as necessary.

  ```
  mkdir /usr/local/ssl/lib && ln -s /path/to/existing/libssl.so.1.0.x /usr/local/ssl/libssl.so
  ```

- Recompile the encryption plugin following the above directions.
- Remove /usr/local/ssl/lib.donotuse to avoid future complications.

Upgrading to GT.M V7.0-002

There are two upgrade paths available when you upgrade to GT.M V7.0-002:

- **V7 Upgrade Path 1**: Perform a MUPIP EXTRACT -FREEZE on an existing V6 database and then perform MUPIP LOAD on an empty V7 database using replication to minimize downtime. You can also use the MERGE command to move data from V6 to the new V7 database; as with MUPIP EXTRACT, you need to keep the source data in a stable state. If you are using triggers, extract the triggers from the V6 database and load them in the new V7 database.
- **V7 Upgrade Path 2**: Continue using your V6 databases with GT.M V7.0-002.

Choose V7 Upgrade Path 1 if you anticipate a database file to grow up to 16Gi blocks. Note that the maximum size of a V7 database file having 8KiB block size is 114688GiB (8KiB*16Gi).

Choose V7 Upgrade Path 2 if you do not anticipate a database file to grow beyond the V6 database limit of 994Mi blocks. Note that the maximum size of a V6 database file having 8KiB block size is 7936GiB (8KiB*992Mi).

Other that the new maximum database file size that comes with V7 Upgrade Path 1, there is no difference between V7 Upgrade Path 1 and V7 Upgrade Path 2. You can choose V7 Upgrade Path 2 first and then later choose V7 Upgrade Path 1 if a need arises.

Note

To upgrade your database from V5 to V7, you need to first upgrade your database from V5 to a V6 database and then choose an appropriate V7 upgrade path. Refer to the appropriate V6 release notes for the database upgrade instructions or consult your GT.M support channel.

Important

In an upcoming release, the FIS GT.M team plans to provide in-place database upgrade with minimal downtime.

Managing M mode and UTF-8 mode

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

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

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

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

- Actual files for GT.M executable programs (mumps, mupip, dse, lke, and so on) are in the parent directory, that is, the location specified for installation.
- Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for UTF-8 mode in the utf8 subdirectory, and one compiled without support for UTF-8 mode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install UTF-8 mode support. Note that on 64-bit versions of GT.M, the object code is in shared libraries, rather than individual files in the directory.
- The utf8 subdirectory has files called mumps, mupip, dse, lke, and so on, which are relative symbolic links to the executables in the parent directory (for example, mumps is the symbolic link ../mumps).
- When a shell process sources the file gtmprofile, the behavior is as follows:
  - If $gtm_chset is "m", "M" or undefined, there is no change from the previous GT.M versions to the value of the environment variable $gtmroutines.
  - If $gtm_chset is "UTF-8" (the check is case-insensitive),
    - $gtm_dist is set to the utf8 subdirectory (that is, if GT.M is installed in /usr/lib/fis-gtm/gtm_V7.0-002_i686, then gtmprofile sets $gtm_dist to /usr/lib/fis-gtm/gtm_V7.0-002_i686/utf8).
    - On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

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

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

Setting the environment variable TERM

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

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

- GT.M uses the following terminfo capabilities. The full variable name is followed by the capname in parenthesis:

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

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

Installing Compression Libraries

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

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

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

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

Change History

V7.0-002

Fixes and enhancements specific to V7.0-002:

Id	Prior Id	Category	Summary
GTM-F132372	GTM-7456	DB	The replication Update process and associated helper process record database-related statistics ✅
GTM-F135292	GTM-9151	Language	Optional second argument to $ZJOBEXAM() to control its output ✅
GTM-DE201305	GTM-9290	Admin	MUPIP BACKUP works if environment variables used in segment to database file mapping are not defined
GTM-F135393	GTM-9393	Language	$ZMALLOCLIM ISV sets up a trappable warning of high memory usage by a process ✅
GTM-F135414	GTM-9426	DB	🔴 Proactive Database Block Splitting ✅
GTM-F135415	GTM-9428	Other	Defer a TLS renegotiation when a prior TLS renegotiation is pending
GTM-DE201378	GTM-9455	Language	Prevent $[Z]CHAR() represenions from generationg results longer than the maximum string length
GTM-DE201380	GTM-9457	Language	SOCKET device commands defend against large deviceparameter arguments
GTM-DE201381	GTM-9458	Admin	MUPIP LOAD -FORMAT=BINARY uses only data length in checking for maximum length
GTM-DE201386	GTM-9465	Language	Address issues which can result in a segmentation violation (SIG-11) or similar failure
GTM-DE201388	GTM-9468	Language	Better handling of literal arguments that cause numeric overflow
GTM-DE201389	GTM-9469	Language	🔴 Pattern match better guards against deep recursion
GTM-DE201390	GTM-9470	Language	🔴 CTRAP only recognizes characters with ASCII codes 0-31 inclusive ✅
GTM-DE201392	GTM-9472	Other	Routine Management asynchronous event protection
GTM-DE201393	GTM-9474	Language	@x@y indirection ignores comments in x and handles empty string arguments appropriately
GTM-F135433	GTM-9480	Language	SET $ZTRAP produces appropriate result
GTM-F135435	GTM-9482	Other	Routine shared object integrity check & repair
GTM-F135842	-	Admin	🔴 MUPIP BACKUP supports user specified order ✅
GTM-DE222430	-	Other	Prevent fatal errors from disconnect/hangup ✅
GTM-DE201825	-	Admin	The configure script removes semsstat2, ftok, and getuid

Database

- The replication update process and its helper processes record statistics using the STATSDB facility for reporting using ^%YGBLSTAT. GT.M reports the WRL, PRG, WFL, and WHE statistics as part of the YGBLSTAT output. See the GT.M Administration & Operations Guide for more details. Previously, the replication update process and its helper processes did not record statistics for STATSDB. (GTM-F132372) ✓

- By default, GT.M proactively splits blocks when it detects significant restarts in an attempt to reduce overall restarts. MUPIP SET -FILE mumps.dat -PROBLKSPLIT=N where N is 0 disables proactive splitting, as do very large values of N. Values of N greater than 0 adjust the lower limit for the number of records below which GT.M does not consider splitting a block. While this is behavior is aimed at reducing restarts, because it reduces data density, it may also increase the number of blocks and even the tree depth. (GTM-F135414) ✓ ✓

Language

- $ZJOBEXAM() recognizes an optional second argument of an expr that evaluates to a string as described for the argument of ZSHOW specifying one or more codes determining the nature of the information produced by the function. If the argument is missing or empty, GT.M operates as if it was a "*" and produces the associated context. This provides a way to suppress content that might contain PNI. Previously, $ZJOBEXAM() always produced a full context. (GTM-F135292) ✓

- $ZMALL[OCLIM] provides a way for a process to limit its process memory. When the value is zero (0), GT.M imposes no limit, although the OS still does. A positive value specifies a byte limit with a minimum of 2.5MB. A value of minus one (-1) provides a value of half the system imposed limit if any. When a request for additional memory exceeds the limit, GT.M does the expansion and then produces a trappable MALLOCCRIT warning. By default, some later request for memory is likely to produce a fatal MEMORY error, unless subsequent to MALLOCCRIT, a limit has been reestablished by SET ZMALLOCLIM to the same or higher limit, but one not exceeding any system limit. Note that GT.M allocates memory from the OS in large blocks so the interaction of $ZMALLOCLIM with memory growth is not exact. In the case of a MEMORY error, GT.M makes an attempt to marshal available memory to enable as graceful a termination as possible. Note that independent of this mechanism, the OS may kill the process without recourse if it determines the greed of the process for memory jeopardizes the viability of the system. When the integer byte value specified in a SET $ZMALLOCLIN=intexpr or, at process startup, by the $gtm_malloc_limit environment variable, specifies a positive value, GT.M uses the smaller of that value and any OS defined amount for the value of $ZMALLOCLIM. GT.M does not give errors or messages about its choice for $ZMALLOCLIM between a specified value and some other more appropriate value, so if the application needs to verify the outcome, it should examine the resulting ISV value. MEMORY errors are fatal and terminate the process. Previously, fatal MEMORY errors were not preceded by a trappable MALLOCCRIT warning. (GTM-F135393) ✓

- Operations that replace non-graphic characters with $[Z]CHAR() representations give a MAXSTRLEN error when the result would exceed 1MiB. Previously such operations ensured they had adequate space for the result, but could pass strings to other operations that might have no provision for dealing with strings longer than 1MiB. (GTM-DE201378)

- USE SOCKET handles the ATTACH, DETACH, CONNECT, SOCKET and ZLISTEN deviceparameters appropriately; previously, certain arguments for these deviceparameters could cause a segmentation violation(SIG-11). (GTM-DE201380)

- Address issues that can result in a segmentation violation (SIG-11) or similar failure.

  GT.M appropriately detects divide by zero; previously there were some unusual combinations of calculations which produced a SIGINTDIV (SIG-8), which could not be trapped and therefore terminated the process.

  The GT.M compiler appropriately handles optimization of literal FALSE post conditionals; previously these could cause segmentation violations (SIG-11). The GT.M compiler deals with argument-less BREAK and NEW commands combined with certain patterns of local variable use, and also with NEW commands having indirect arguments after a FOR command. Previously these could cause a segmentation violation (SIG-11).

  The VIEW command for "NOISOLATION" handles various edge cases and reports a VIEWARGTOOLONG error when a list of global names exceeds 1024 bytes. Previously, some malformed global names could cause a segmentation violation (SIG-11) and using a string over 1024 bytes would cause a GTM assertion failure.

  The ZSHOW command operates as expected when directing the output to a MUMPS local. Previously repeatedly running the ZSHOW command in an indefinite loop with the output directed to a MUMPS local could result in a segmentation violation (SIG-11).

  The $FNUMBER() and $JUSTIFY() functions work as expected. Previously, results requiring long lengths could result in a segmentation violation (SIG-11).

  The GT.M compiler protects against an indirect argument combined with a missing closing parenthesis in a $INCREMENT() function; previously some syntax with that characteristic produced a segmentation violation (SIG-11).

  $ZSYSLOG() ignores Format Ascii Output (FAO) directives. Previously, $ZSYSLOG() did not ignore such directives resulting in a segmentation violation (SIG-11).

The USE command works as expected when the indirect device name contains an ill formatted string. Previously certain strings could cause a segmentation violation (SIG-11).
GT.M ignores unknown device class mnemonic spaces. Previously, if a mnemonic space other than "SOCKET" or "PIPE" was used, a subsequent OPEN resulted in a segmentation violation (SIG-11).(GTM-DE201386)

- $[Z]CHAR(1E48) and similar numeric literal overflows produce a NUMOFLOW error at compile time; previously an optimization inappropriately treated this as $[Z]CHAR(0). (GTM-DE201388)

- GT.M pattern match reports a PATMAXLEN error for a deeply nested pattern containing many indefinite ranges (. symbols). While we think it unlikely, it is possible that existing complex patterns may now report this error. Previously such a deeply nested pattern terminated with a segmentation violation (SIG-11). (GTM-DE201389) ✓

- CTRAP characters other than CTRL-x, where x corresponds to the non-graphic characters represented by $CHAR(n) with n 0 to 31 inclusive, have no effect, and do not show up in ZSHOW "D" output. On the keyboard those characters correspond to <CTRL-@> to CTRL-_> with the bulk of the range being <CTRL-A> to <CTRL-Z>. Note that CTRAP=$CHAR(3) has a different semantic than the other CTRAP characters, in that, when enabled and $PRINCIPAL is a terminal device type, <CTRL-C> can interrupt at any time, including when $PRINCIPAL'=$IO. GT.M only recognizes the other CTRAP characters when they appear in input to a READ command. Previously, characters outside the range with codes 0-31 showed up in ZSHOW "D" output with a modulo 32 $CHAR() representation and caused corresponding CTRAP recognition. (GTM-DE201390) ✓ ✓

- Indirection of the form @x@y ignores any comment at the end of the value of x; previously such a comment stopped the evaluation so the subscripts after the second @ were ignored. In addition, if the first (x) expression evaluates to the empty string, GT.M produces a VAREXPECTED error; previously that scenario produced incorrect results, sub-optimal errors or, in the worst case damage to an item at the top of the heap. Also, if the second (y) expression is the empty string, it no longer has the potential to lead to incorrect results or sub-optimal error reports. (GTM-DE201393)

- SET $ZTRAP=$ETRAP works correctly; previously it tended to result in $ZTRAP="" regardless of the prior value of $ETRAP, but could also produce garbage values, while less likely, other assignments to $ZTRAP could also go awry. The workaround was to NEW $ETRAP, assign a the value of $ETRAP to a temporary, the empty string to $ETRAP and $ZTRAP to the value of the temporary. (GTM-F135433)

System Administration

- When the Database segment to file mapping uses an environment variable, and the environment variable is not defined, MUPIP BACKUP uses the environment variable name itself in constructing the file name, as does MUPIP CREATE. Previously MUPIP CREATE created database with undefined environment variable name as the database name, but MUPIP BACKUP failed to backup that existing database. (GTM-DE201305)
- MUPIP LOAD -FORMAT=BINARY checks the record length against the data for a record; starting with V6.0-000, GT.M defines record length as the data in a node, but the utility in question still inappropriately included the key in its length check. (GTM-DE201381)
- When MUPIP BACKUP arguments specify a list, the utility processes regions in the listed order, or, for names expanded by wildcard ("*"), alphabetically. Previously, MUPIP BACKUP ignored any user-specified order of regions, and processed regions in FTOK order, which tends to change with changes in operational conditions within the underlying file system.(GTM-F135842) ✓✓
- The configure script checks for and removes semsstat2, ftok, and getuid files while installing GT.M in an existing directory where a GT.M installation already exists. Note that GT.M versions starting from V6.3-014 no longer require these files. Previously, the configure script did not remove these obsolete executable files.(GTM-DE201825)

Other

- The Source Server defers a TLS renegotiation when a prior TLS renegotiation is pending. Previously, the Source Server logged the REPLWARN message, closed the replication connection, and attempted to re-establish the replication connection with the Receiver Server.(GTM-F135415)
- GT.M provides protection from asynchronous events during routine management. Previously, asynchronous events like MUPIP STOP at inopportune times could cause a segmentation violation (SIG-11). This was only seen in development and has never been reported by a customer.(GTM-DE201392)
- GT.M detects and recovers from certain integrity errors in the auto-relink control structures. GT.M also issues RLNKRECNFL and RLNKINTEGINFO messages to the system log to report an integrity check. Previously, GT.M terminated with an segmentation violation (SIG-11) or a GT.M assert when it encountered such integrity errors. (GTM-F135435)
- GT.M handles SIGHUP appropriately when $PRINCIPAL has HUPENABLE set; in V7.0-001 due to changes in deferred event handling, error handing could encounter a GTMASSERT2. In addition, a TERMHANGUP error implicitly sets the device to NOHUPENABLE, so should a process anticipate multiple disconnects/hangups, it should explicitly issue a USE $PRINCIPAL:HUPENABLE. Also, ZSHOW "D" displays the HUPENABLE state for $PRINCIPAL. (GTM-DE222430) ✓

Error and Other Messages

MALLOCCRIT

MALLOCCRIT, Memory allocation critical due to request for bbbb bytes from aaaa

All GT.M Components Warning: Indicates a GT.M process exceeded the memory allocation threshold established with $ZMALLOCLIM with a request for bbbb bytes. The address aaaa gives a location in a GT.M executable, likely only useful to your GT.M support channel.

Action: Consider diagnosing the process behavior. For example, look for a resource leak, or a more resource efficient approach. The size of the request may be helpful in indicating how aggressively the process is growing. The MALLOCRIT invokes the error handler, and may need special handling to resume execution at the point it was detected. By default, some later request for memory is likely to produce a fatal MEMORY error, unless a subsequent SET $ZMALLOCLIM reestablishes the same or higher limit not exceeding any system limit. MEMORY errors are fatal and terminate the process. Independent of this mechanism, the OS may kill the process without recourse if it determines the greed of the process for memory jeopardizes the viability of the system.

RLNKINTEGINFO

RLNKINTEGINFO, Integrity check completed successfully: xxxx -- called from module yyyy at line zzzz

Run Time Information: Indicates relinkctl integrity check completed successfully by performing the action described by xxxx. GT.M issues this message after RLNKRECNFL, otherwise, it means the relinkctl integrity check failed. This message was called from the yyyy module at zzzz line, and it goes to the operator log.

Action: If the integrity check result succeeds, the process can continue, otherwise, GT.M generates a trappable error, right after this message. Contact your GT.M support channel to discuss what might have caused the issue.

RLNKRECNFL

RLNKRECNFL, Conflict on relinkctl file rrrr for $ZROUTINES directory dddd, running an integrity check

Run Time Warning: Indicates a process encountered an issue attempting to attach to a routine object in dynamically linked library associated with directory dddd, and initiated an integrity check of the library control structures associated with relinkctl file rrrr. GT.M sends this message to the operator log.

Action: If the check finds no problem or can correct any abnormality it finds (look for the RLNKINTEGINFO message in the operator log, to have more information about the integrity check result), the process can continue, otherwise, GT.M generates a trappable error. Contact your GT.M support channel to discuss what might have caused the issue.

VIEWARGTOOLONG

VIEWARGTOOLONG, The argument length LLLL to VIEW command vvvv exceeds the maximum mmmm

Run Time Error: An argument to the VIEW command vvvv has a length LLLL bytes that exceeds the maximum supported length of mmmm.

Action: Reduce the length of the VIEW command argument to no more than mmmm bytes.