



V6.3-000



# GT.M

**Release Notes**



## 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](mailto: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

Copyright ©2016 Fidelity National Information Services, Inc. and/or its subsidiaries. All Rights Reserved.

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








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

































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

Revision History		
Revision 1.1	11 April 2016	<ul style="list-style-type: none"><li>Added V6.3-000 messages that got missed in revision 1.0 due to a documentation error.</li><li>Added the release note for GTM-8511.</li><li>In Additional Information for GTM-8296 - %PEEKBYNAME(), added an entry for the epoch taper set region parameter.</li></ul>
Revision 1.0	29 March 2016	V6.3-000



# Table of Contents

V6.3-000 .....	1
Overview .....	1
Conventions .....	2
Platforms .....	4
Platform support lifecycle .....	7
32- vs. 64-bit platforms .....	7
Call-ins and External Calls .....	7
Internationalization (Collation) .....	8
Environment Translation .....	8
Recompile .....	8
Rebuild Shared Libraries or Images .....	8
Additional Installation Instructions .....	9
.....	9
Upgrading to GT.M V6.3-000 .....	10
Stage 1: Global Directory Upgrade .....	11
Stage 2: Database Files Upgrade .....	11
Stage 3: Replication Instance File Upgrade .....	13
Stage 4: Journal Files Upgrade .....	14
Stage 5: Trigger Definitions Upgrade .....	14
Downgrading to V5 or V4 .....	15
Managing M mode and UTF-8 mode .....	16
Setting the environment variable TERM .....	18
Installing Compression Libraries .....	18
Change History .....	19
V6.3-000 .....	19
Database .....	25
Language .....	29
System Administration .....	35
Other .....	47
More Information .....	49
Additional information for GTM-7291 - MUPIP JOURNAL -ROLLBACK qualifiers .....	49
Additional Information for GTM-8296 - %PEEKBYNAME() .....	50
Examples: .....	51
LISTALL^%PEEKBYNAME .....	51
LIST^%PEEKBYNAME(.output) .....	51
Labels for Selected Fields .....	51
Error and Other Messages .....	57
CRYPTJNLMISMATCH  .....	57
CRYPTKEYRELEASEFAILED  .....	57
CRYPTNOKEY  .....	57
ENCRYPTCONFLT  .....	57
EXTRINTEGRITY  .....	58
GDINVALID  .....	58
INVLINKTMPDIR  .....	58

INVLOCALE 	58
INVZWRITECHAR 	59
IOEOF 	59
JNLDBSEQNOMATCH 	59
JNLPOOLRECOVERY 	59
JOBLVN2LONG 	60
JOBLVNDETAIL 	60
MULTIPROCLATCH 	60
MUIPSET2BIG 	60
MUIPSET2SML 	61
MUPJNLINTERRUPT 	61
MUREENCRYPTEND 	61
MUREENCRYPTSTART 	61
MUREENCRYPTV4NOALLOW 	61
NLRESTORE 	62
NOMORESEMCNT 	62
NONTPRESTART 	62
NOPRINCIO 	63
NOTALLJNLLEN 	63
NOTALLREPLON 	63
PBNINVALID 	64
PBNNOFIELD 	64
PBNNOPARM 	64
PBNPARAMREQ 	64
PBNUNSUPSTRUCT 	64
RELOAD 	65
REPLLOGOPN 	65
REPLSTATEOFF 	65
RESRCINTRLCKBYPAS 	65
SETQUALPROB 	66
TPRESTART 	66
TRIGINVCHSET 	67

---

## V6.3-000

---

### Overview

V6.3-000 brings significant enhancements to GT.M's use of encryption. One defensive technique is to reduce the "surface" available to an attacker. V6.3-000 reduces the surface in several ways.

An attacker with the wherewithal for a brute-force attack on encryption can in theory benefit from the voluminous, long-lived, and structurally similar data in a typical application database, such as financial transactions and medical records. One component of a traditional layered defense-in-depth is to change encryption keys regularly. Enabling encryption keys for database files to be changed "on the fly" while a database is in use with V6.3-000 (GTM-6310) operationally simplifies the changing of the keys, and makes key changes less prone to human error. The prior technique required database regions to be extracted and loaded into newly created database files with keys different from those of their predecessors. Context-sensitive initialization vectors (IVs) in database, journal, extract and bytestream backup files (GTM-8117) further reduce the surface for a brute-force attack.

A properly configured Transport Layer Security (TLS; formerly known as SSL) session is required to secure a TCP connection. However, an attacker that can record a TCP session, and with the wherewithal for a brute force attack, or with more affordable future computing power, can in theory retroactively break into and eavesdrop on the recorded session. Periodically renegotiating the session key (GTM-8302) means that an attacker who succeeds in breaking a key can only eavesdrop on that part of the session - every renegotiation generates a new key that must be separately broken.

Note that GT.M continues to include no cryptographic software - cryptographic functionality is provided by your choice of independent, non-GT.M, cryptographic software that GT.M access through a plugin. Distributions of GT.M since the introduction of database encryption have included the source code for reference implementations of the plugin as tested by FIS in the GT.M development environment against versions of popular encryption packages noted in the release notes for each GT.M release. In V6.3-000 (GTM-8361), GT.M includes the source code of the encryption plugin, but not pre-compiled binaries, because the wide range of versions of cryptographic software across Supported platforms made it infeasible for us to provide a single binary that was guaranteed to run with the robustness we require of GT.M.

V6.3-000 brings a number of useful enhancements, as well as other improvements. For example:

- Parallelization speeds MUPIP JOURNAL RECOVER/ROLLBACK operations (GTM-5007).
- For a replicated database even of an application that does not use transaction processing, MUPIP JOURNAL -ROLLBACK -FORWARD applies updates from a set of journal files to the restored backup of a multi-region database, bringing it to the same state that MUPIP JOURNAL -ROLLBACK -BACKWARD would when performed on the original database, providing the same consistency across regions that the MUPIP JOURNAL -ROLLBACK provides (GTM-7291).
- Faster database exit, especially with large numbers of processes and databases with many regions (GTM-6301).

- Evaluation of certain string literal operations during compilation rather than execution (GTM-7762 and GTM-8404).
- Concurrent access by more than 32K processes to a database file (GTM-8137).
- Significant performance improvements for certain UTF-8 mode use cases (GTM-8352)

Effective V6.3-000, we are changing the organization of core information in the release notes for each GT.M version. Instead of M - Database Access, M - Other than Database Access, Utilities - MUPIP, and Utilities-Other than MUPIP, we have the following sections:

- Database - the core of GT.M; items we believe are of interest to all users
- Language - language features; primarily of interest to programmers
- System Administration - MUPIP and GDE; primarily of interest to administration and operations staff
- Other - DSE, LKE, and changes potentially of interest to a smaller subset of users than the sections above

As always, the release bring numerous smaller enhancements, and fixes. See the Change History below.

*Please note that messages are not part of the GT.M API whose stability we strive to maintain. The enhancements and fixes in this release bring more changes to messages, including in some cases the order of messages, than a typical GT.M release does. Make sure that you review any automated scripting that parses GT.M messages.*

---

## Conventions

This document uses the following conventions:

<b>Flag/Qualifiers</b>	-
<b>Program Names or Functions</b>	upper case. For example, MUPIP BACKUP
<b>Examples</b>	lower case. For example: \$char(10) mupip backup - database ACN,HIST /backup
<b>Reference Number</b>	A reference number is used to track software \$char(10) enhancements and support requests. \$char(10) It is enclosed between parentheses ().
<b>Platform Identifier</b>	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.  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.	-updnok
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). 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 may not be upward compatible and may affect an existing application.

⊘ denotes deprecated messages.

⚠ 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.

GT.M runs on a variety of UNIX/Linux implementations. 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 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.

Note: V6.3-000 is the last release on which FIS plans to test the reference implementation of the encryption plugin with the Blowfish algorithm. When database encryption was first introduced, usable implementations of AES did not exist on all GT.M platforms. Although FIS neither recommends nor recommends against the use of any specific cipher, we test the plugin against what we expect to be in common use.

Platform	Supported Versions	Notes
Hewlett-Packard Integrity IA64 HP-UX	-	<i>V6.2-002A was the last GT.M release for this platform, which is no longer Supported. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.</i>
Hewlett-Packard Alpha/AXP OpenVMS	-	<i>V6.2-001 was the last GT.M release for this platform, which is no longer supported. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.</i>
IBM System p AIX	6.1, 7.1	Only 64-bit versions of AIX 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.

Platform	Supported Versions	Notes
		<p>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 <b>instfix -ik IZ87564</b>.</p> <p>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.</p> <p>Effective the next release, FIS intends to require POWER6 as the minimum CPU architecture level on this for AIX.</p>
Oracle (Sun) SPARC Solaris	-	<p><i>V6.2-002A was the last GT.M release for this platform, which is no longer Supported. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.</i></p>
x86_64 GNU/Linux	Red Hat Enterprise Linux 6 and 7; Ubuntu 12.04 LTS 14.04 LTS; SuSE Linux Enterprise Server 11	<p>To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware.</p> <p>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).</p> <p>To install GT.M with Unicode (UTF-8) support on RHEL 6, in response to the installation question <b>Should an ICU version other than the default be used? (y or n)</b> please respond <b>y</b> and then specify the ICU version (for example, respond 4.2) to the subsequent prompt <b>Enter ICU version (ICU version 3.6 or later required. Enter as major-ver.minor-ver):</b></p> <p>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):</p> <ul style="list-style-type: none"> <li>• Find the directory where libncurses.so is installed on your system.</li> <li>• Change to that directory and make a symbolic link to libncurses.so.&lt;ver&gt; from libtinfo.so.&lt;ver&gt;. 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.</li> </ul> <p>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. As this is a higher level than that distributed with Red Hat Enterprise Linux 6 or Ubuntu 12.04 LTS, in order to use this feature</p>

Platform	Supported Versions	Notes
		<p>of WRITE/TLS on those platforms with the reference implementation, please install libconfig 1.4.x, including the header files, and recompile the reference implementation of the encryption plugin.</p> <p>A bug in the Linux 3.13 kernels used in Ubuntu 14.04 LTS (<a href="https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1502168">https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1502168</a>) affects GT.M operation. As newer kernels do not exhibit this misbehavior, FIS recommends that you follow the Ubuntu LTS Enablement Stack procedure (<a href="https://wiki.ubuntu.com/Kernel/LTSEnablementStack">https://wiki.ubuntu.com/Kernel/LTSEnablementStack</a>) and use newer kernels to avoid the behavior until such time as the bug is fixed in the 3.13 kernels.</p> <p>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 <i>must</i> ensure that your kernel includes commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at <a href="https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3">https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3</a>) is in your kernel. The Red Hat Bugzilla identifier for the bug is 1213487. With NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs.</p> <p>In the future, FIS intends:</p> <ul style="list-style-type: none"> <li>• effective July 1, 2017, to require the then current level of 7 (e.g, 7.2) as the minimum supported level of Red Hat Enterprise Linux;</li> <li>• effective July 1, 2017, to require 16.04 LTS, assuming it is released as anticipated in April 2016, as the minimum supported level of Ubuntu Linux; and</li> <li>• effective October 1, 2016, to no longer consider SuSE Linux Enterprise Server to be Supported (it will remain Supportable to the extent that Linux distributions other than Supported ones are considered Supportable, as noted above).</li> </ul> <p>If these will cause you hardship, please contact your FIS account manager or your GT.M support channel.</p>
x86 GNU/Linux	Red Hat Enterprise Linux 6 and 7	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. Running a 32-bit GT.M on a 64-bit GNU/Linux requires 32-bit libraries to be installed. The

Platform	Supported Versions	Notes
		<p>CPU must have an instruction set equivalent to 586 (Pentium) or better. Also, refer to the notes above on the 64-bit version.</p> <p>Please also refer to the notes above on x86_64 GNU/Linux.</p>

## 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 GT.M object code in 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.
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
<code>gtm_descriptor</code> in <code>gtm_descript.h</code>	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
<code>gtm_string_t</code> type in <code>gtmxc_types.h</code>	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 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 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 again 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 V6.3-000 in a Filesystem Hierarchy Standard compliant location such as `/usr/lib/fis-gtm/V6.3-000_arch` (for example, `/usr/lib/fis-gtm/V6.3-000_x86` on 32-bit Linux systems). A location such as `/opt/fis-gtm/V6.3-000_arch` would also be appropriate. Note that the **arch** suffix is especially important if you plan to install 32- and 64-bit versions of the same release of GT.M on the same system.
- Use the MUPIP RUNDOWN command of the old GT.M version to ensure all database files are cleanly closed.
- 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. Also, modify your environment scripts to include the use of `gtmcrypt_config` environment variable.

## Compiling the Reference Implementation Plugin

If you plan to use database encryption and TLS replication, 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 GT.M is installed
  - Copy and paste the commands from the README to compile and install the encryption plugin with the permissions defined at install time

---

## Upgrading to GT.M V6.3-000

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

GT.M upgrade procedure for V6.3-000 consists of 5 stages:

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

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V6.3-000 depends on your GT.M upgrade history and your current version.



## Stage 1: Global Directory Upgrade

FIS strongly recommends you back up your Global Directory file before upgrading. There is no single-step method for downgrading a Global Directory file 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 V6.3-000.
- Execute the EXIT command. This command automatically upgrades the Global Directory.

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

1. Open your Global Directory with the GDE utility program on the 32-bit platform.
2. On GT.M versions that support SHOW -COMMAND, execute SHOW -COMMAND -FILE=file-name. This command stores the current Global Directory settings in file-name.
3. On GT.M versions that do not support GDE SHOW -COMMAND, execute the SHOW -ALL command. Use the information from the output to create an appropriate command file or use it as a guide to manually enter commands in GDE.
4. Open GDE on the 64-bit platform. If you have a command file from 2. or 3., execute @file-name and then run the EXIT command. These commands automatically create the Global Directory. Otherwise use the GDE output from the old Global Directory and apply the settings in the new environment.

An analogous procedure applies in the reverse direction.

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

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

- Open the global directory with the GDE utility program of V6.3-000.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

## Stage 2: Database Files Upgrade

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

A V6 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 database created with V6 supports up to 992Ki blocks and is not backward compatible. V6 databases that take advantage of V6 limits on key size and records size cannot be downgraded. 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 V6.3-000 operations. However, that database can only grow to the maximum size of the version in which it was originally created. A database created on V5.0-000 through V5.3-003 has maximum size of 128Mi blocks. A database created on V5.4-000 through V5.5-000 has a maximum size of 224Mi blocks. Only a database created with V6.0-000 or above (with a V6 MUPIP CREATE) has a maximum database size of 992Mi blocks.



### Important

In order to perform a database downgrade you must perform a MUPIP INTEG -NOONLINE. If the duration of the MUPIP INTEG will exceed the time allotted for an upgrade you should rely on a rolling upgrade scheme using replication.

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 syslog 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.
- For information on downgrading a database upgraded from V6 to V5, refer to: Downgrading to V5 or V4.

### Stage 3: Replication Instance File Upgrade

V6.3-000 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.3-000 requires new replication instance files if you are upgrading from any version prior to V5.5-000. Instructions for creating new replication instance files are in the Database Replication chapter

of the GT.M Administration and Operations Guide. 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 described in the Database Replication chapter of the GT.M Administration and Operations Guide 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 V6.3-000 and you have database triggers defined in V6.2-000 or earlier, you need to ensure that your trigger definitions are wholesome in the older version and then run MUPIP TRIGGER -UPGRADE. If you have doubts about the wholesomeness of the trigger definitions in the old version use the instructions below to capture the definitions delete

them in the old version (-\*), run MUPIP TRIGGER -UPGRADE in V6.3-000 and then reload them as described below.

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 V6.3-000 or if you find your prior version trigger definitions have problems. For versions V5.4-000/V5.4-000A/V5.4-001 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/V6.3-000.

To extract and reapply the trigger definitions on V6.3-000 using MUPIP TRIGGER:

1. Using the old version, execute a command like **mupip trigger -select="" trigger\_defs.trg**. 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. Using V6.3-000, run **mupip trigger -triggerfile=trigger\_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.3-000 replicating instance using \$ZTRIGGER():

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



### Note

Reloading triggers rennumbers 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.

To verify that your database meets all of the above requirements, execute `MUPIP INTEG -NOONLINE`. Note that the integrity check requires the use of `-NOONLINE` to ensure no concurrent updates invalidate the above requirements. Once assured that 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 V6.3-000 environment:
  - a. Execute `MUPIP SET -VERSION=v4` so that GT.M writes updates blocks in V4 format.
  - b. Execute `MUPIP REORG -DOWNGRADE` to convert all blocks from V6 format to V4 format.
2. Bring down all V6 GT.M processes and execute `MUPIP RUNDOWN -FILE` on each database file to ensure that there are no processes accessing the database files.
3. Execute `MUPIP DOWNGRADE -VERSION=V4` to change the database file header from V6 to V4.
4. Restore or recreate all the V4 global directory files.
5. Your database is now successfully downgraded to V4.

---

## Managing M mode and UTF-8 mode

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

On a system that has ICU installed, GT.M optionally installs support for both M mode and UTF-8 mode, including a `utf8` subdirectory of the directory where GT.M is installed. From the same source

file, depending upon the value of the environment variable `gtm_chset`, the GT.M compiler generates an object file either for M mode or UTF-8 mode. GT.M generates a new object file when it finds both a source and an object file, and the object predates the source file and was generated with the same setting of `$gtm_chset/$ZCHset`. A GT.M process generates an error if it encounters an object file generated with a different setting of `$gtm_chset/$ZCHset` than that processes' current value.

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

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

- Actual files for GT.M executable programs (`mumps`, `mupip`, `dse`, `lke`, and so on) are in the parent directory, that is, the location specified for installation.
- Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for Unicode in the `utf8` subdirectory, and one compiled without support for Unicode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install Unicode support. 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_V6.3-000_i686`, then `gtmprofile` sets `$gtm_dist` to `/usr/lib/fis-gtm/gtm_V6.3-000_i686/utf8`).
    - On platforms where the object files have not been placed in a `libgtmutil.so` shared library, the last element of `$gtmroutines` is `$gtm_dist($gtm_dist/..)` so that the source files in the parent directory for utility programs are matched with object files in the `utf8` subdirectory. On platforms where the object files are in `libgtmutil.so`, that shared library is the one with the object files compiled in the mode for the process.

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

Although GT.M uses ICU, 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(smkn), 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.

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.



---

## Change History

---

### V6.3-000

Fixes and enhancements specific to V6.3-000 are:

Id	Prior Id	Category	Summary
GTM-5007	C9D07-002329	Admin	MUPIP JOURNAL parallelization option, sorted lost and broken transaction files, and change in default of -verify ✔
GTM-5726	S9F07-002557	Admin	See GTM-7768
GTM-6114	C9I01-002943	Language	ZMESSAGE handles indirection
GTM-6301	C9I12-003057	DB	Cleaner and quicker database cleanup on process exit
GTM-6310	C9I12-003066	Admin	Change database file encryption keys "on the fly" while databases are in use ✔
GTM-6317	C9J01-003070	Admin	See GTM-8137
GTM-6388	C9J05-003131	Admin	MUPIP EXTRACT and MUPIP JOURNAL - EXTRACT performance improvement
GTM-6858	S9L07-002825	Admin	Building ICU on AIX no longer required ✔
GTM-6928	-	Admin	ZEROBACKLOG qualifier to shut down Source Server when backlog is zero ✔
GTM-7060	-	Language	Better behavior when a USE command puts \$X beyond WIDTH ✔
GTM-7199	-	Other	DSE ALL -CLEARCORRUPT clears the CORRUPT flag on all regions
GTM-7291	-	Admin	MUPIP JOURNAL -ROLLBACK - FORWARD and enhancements to MUPIP JOURNAL -ROLLBACK -BACKWARD ✔
GTM-7375	-	Admin	More tolerant ICU specification ✔
GTM-7604	-	Language	\$ZWRITE() can go from, as well as to, ZWRITE format ✔
GTM-7608	-	DB	Processes with read-only access leave shared memory intact for databases not shutdown in an orderly fashion

Id	Prior Id	Category	Summary
GTM-7658	-	Other	Enhancements and fixes to ^%RI utility ✔
GTM-7762	-	Language	Compiler evaluation of some operations and functions containing string literals ✔
GTM-7768	-	Admin	Improved Source Server communication management
GTM-7809	-	Admin	Addition critical section tuning options ✔
GTM-7831	-	Admin	Online Rollback avoids inappropriate Instance Freeze on DBDANGER
GTM-7838	-	Admin	Replication Flow Control adjustment
GTM-8009	-	Other	Prevent inappropriate file sharing after process creation
GTM-8020	-	Admin	MUPIP LOAD accepts unquoted negative values
GTM-8022	-	Other	^%GI accepts records up to the maximum string length ✔
GTM-8028	-	Other	Encryption plugin loads correctly for UTF-8 mode processes with standard GT.M installation directory structure
GTM-8034	-	Language	On the JOB command, enable naming of both output and error when appending JOBPID ✔
GTM-8038	-	Language	\$ZCONVERT(...,"T") works in M mode
GTM-8076	-	Other	Journal Pool Recovery
GTM-8112	-	Language	Appropriate handling by the ZSYSTEM command of a very long SHELL environment variable
GTM-8117	-	DB	Better initialization vector available for encrypted databases ✔
GTM-8137	-	Admin	Allowing more than 32Ki processes to access the database and journal pool ✔
GTM-8190	-	DB	Optional logging of non-tp restart conflicts ✔
GTM-8215	-	Language	\$ZCONVERT() raises ICUERROR if ICU library is unavailable

Id	Prior Id	Category	Summary
GTM-8223	-	Admin	MUPIP LOAD recognizes a wider range of labels and also DOS-formatted records ✔
GTM-8225	-	Admin	GT.M uses the default zlib on AIX ✔
GTM-8296	-	Language	Version-agnostic invocation of \$ZPEEK() with ^%PEEKBYNAME wrapper ✔
GTM-8297	-	Language	JOBLVN2LONG message contains length information
GTM-8302	-	Language	TLS Socket renegotiation and other features for server implementation ✔
GTM-8326	-	Admin	Allow journal pool size to be greater than 2GB
GTM-8336	-	DB	Performance improvement initializing encryption for databases with multiple encrypted regions
GTM-8340	-	DB	Receiver, Update Process, and Update Writer Helper Improvements ✔
GTM-8342	-	Admin	Better MUPIP TRIGGER delete confirmation interaction ✔
GTM-8352	-	Language	Improved UTF-8 character handling for \$EXTRACT() and \$FIND() ✔
GTM-8361	-	Admin	Reference implementation of encryption plugin included only as source code ✔
GTM-8389	-	Language	Source references in triggers outside of explicit TP use any available cached source ✔
GTM-8394	-	Admin	Add robustness to ROLLBACK/RECOVER for an operationally odd case
GTM-8395	-	Other	Restore patience for startup of gtmsecshr
GTM-8397	-	Language	Ensure proper delivery of any process termination messages
GTM-8399	-	DB	Better deletion of malformed trigger representations
GTM-8403	-	Other	The %MPIECE utility NEWs all local variables

Id	Prior Id	Category	Summary
GTM-8404	-	Language	Calls to \$TEXT() that are known at compile time are treated as literal strings 🟢
GTM-8407	-	Language	Correct indirection for ZSHOW
GTM-8410	-	Admin	Fix Source Server journal switch race
GTM-8416	-	Other	Better alignment of \$HOROLOG and journal time stamps
GTM-8417	-	Language	Fix for ZPRINT edge case involving triggers, indirection and source file instability
GTM-8420	-	Admin	See GTM-7658
GTM-8421	-	Admin	Receiver Server started with -autorollback continues when an online rollback does not change the state of the database
GTM-8422	-	Admin	See GTM-6388
GTM-8423	-	Admin	The Receiver Server appropriately handles transactions larger than 2MiB
GTM-8425	-	Admin	Fix an evil interaction between the Update Process and Online Rollback
GTM-8428	-	Admin	MUPIP INTEG correctly reports large values for things it counts
GTM-8429	-	Other	Correction to open source builds
GTM-8431	-	Language	\$TEXT() accepts ^GTM\$DMOD and ^GTM\$CI as arguments
GTM-8433	-	Language	ZSHOW expr:gvn output format change and ^%ZSHOWVTOLCL utility 🟢
GTM-8435	-	Language	Prevent a possible hang due to an externally initiated termination
GTM-8440	-	Language	Fix to handling of over-long device names
GTM-8441	-	Other	gtminstall script qualifier fixes
GTM-8443	-	Admin	Fix epoch taper rare edge case
GTM-8448	-	Language	Better handling of MUPIP STOPS
GTM-8450	-	Language	Prevent time overflows in \$ZGETJPI()
GTM-8457	-	DB	Fix for an edge case in trigger delete

Id	Prior Id	Category	Summary
GTM-8458	-	Other	Address possible mis-ordering of DSE output
GTM-8461	-	Admin	Source Server reliably updates the resync sequence number in the replication instance file
GTM-8464	-	Admin	MUPIP EXTRACT on encrypted database creates valid binary extract files regardless of mapping
GTM-8465	-	Language	JOB with PASSCURLVN appropriately handles alias containers containing KILL'd alias variables
GTM-8468	-	Admin	Improved Source Server management of journal files
GTM-8470	-	Admin	See GTM-7831
GTM-8471	-	Other	gtmsecshr does not trigger false ARCTLMAXLOW warnings
GTM-8475	-	Language	Prevent a case where \$ZSEARCH() could hang indefinitely
GTM-8476	-	Language	Fix for error handling in triggers on spanning nodes or spanning regions
GTM-8477	-	Language	GT.M no longer issues an inappropriate TLVLZERO error
GTM-8478	-	Admin	"Sending REPL_RENEG_COMPLETE" in Source Server log file
GTM-8479	-	Admin	MUPIP REORG really does respond to gtm_poollimit
GTM-8480	-	Admin	Minimize size of result from MUPIP BACKUP -DATABASE 🟢
GTM-8481	-	Admin	MUPIP INTEG handles a flood of DBTNTOLG errors from the same global correctly
GTM-8483	-	Language	MUPIP JOURNAL -EXTRACT leaves journal files in a wholesome state
GTM-8484	-	Language	M compilations, trigger compilations, and online integrity checks no longer SIG-11 in rare cases

Id	Prior Id	Category	Summary
GTM-8485	-	Admin	MUPIP JOURNAL correctly issues FILEPARSE errors for invalid paths
GTM-8486	-	Language	ZROSYNTAX errors do not cause segmentation violations
GTM-8487	-	Admin	No more inappropriate REPLINSTMATCH errors
GTM-8489	-	DB	Prevent inappropriate Source Sever termination
GTM-8490	-	Language	VIEW command does not accept 0 and 1 as arguments
GTM-8491	-	Language	Improved \$ZSEARCH on AIX for non-wildcard searches
GTM-8494	-	Admin	Better maintenance of an Instance file by MUPIP BACKUP and MUPIP REPLIC
GTM-8495	-	Other	Improve DSE key interpretation of damaged blocks under some circumstances
GTM-8499	-	DB	Unusual sequence of KILLs does not result in process termination with SIG-11
GTM-8500	-	DB	Deadlock recovery for the internal locks
GTM-8501	-	DB	Fix for NOCHLEFT error during process shutdown
GTM-8502	-	Admin	MUPIP standalone commands use the same caution as MUPIP RUNDOWN 🟢
GTM-8506	-	DB	Prevent trigger problem from causing a deadlock
GTM-8507	-	Admin	Fix rare case that inappropriately terminated replication between two SI nodes
GTM-8511	-	Other	Allow DSE to map resource even if they don't match the file header
GTM-8512	-	Language	JOB command uses \$zgbldir as the default value for the command parameter GBLDIR
GTM-8514	-	Other	--xec is optional for %XCMD

---

## Database

- Processes execute substantially less logic when exiting database files opened with the BG access method. Previously process exit involved more checking, which could significantly slow the exit and consume CPU resources potentially usable by other processes, behavior which was most visible when a large number of processes concurrently attempted to exit databases with large numbers of regions and large numbers of global buffers per region. (GTM-6301)
- A process with read-only access run on an otherwise unaccessed database that was not shutdown in an orderly fashion, due to for example a kill -9, attached processes exceeding 32Ki, etc., leaves shared memory intact. Previously, such a process could cause the loss of the shared resource and, under some circumstances, updates it contained. (GTM-7608)
- GT.M uses non-zero, context-sensitive, initialization vectors (IVs) when it encrypts or decrypts databases, journal files, binary extracts and bytestream backups; previously it used empty (all zeros or "NULL\_IV") initialization vectors.

MUPIP EXTRACT -FORMAT=BIN accepts a -NULL\_IV qualifier and generates extracts in the following formats:

- Level 6 for extracts that include no encrypted region (i.e., no data in the extract is encrypted). This is unchanged from prior releases.
- Level 8 for extracts that include at least one encrypted region (i.e., some or all data is encrypted), and the extract was generated either (a) with the -NULL\_IV flag, or (b) from database regions that are encrypted with null IVs, to generate a file with null IVs. MUPIP LOAD from GT.M V6.2-002/-002A can load level 8 extracts (but see the discussion of GTM-8360 in the V6.2-002 release notes on extracts that mixed encrypted and unencrypted regions in GT.M releases prior to V6.2-002)
- Level 9 for extracts that include at least one encrypted region, and the extract uses IVs - i.e., MUPIP LOAD only from V6.3-000 and future releases will be able to accept this format.

MUPIP BACKUP -BYTESTREAM in V6.3-000 uses level format 8 for an encrypted region and level format 9 for an unencrypted one (i.e., the first 5 characters of the label are "GDSV8" or "GDSV9", respectively). Bytestream backups created from a database with a particular minor version can only be restored onto a database with the same minor version. In case of a minor version mismatch GT.M issues the MUPRESTERR error with a descriptive addendum.

Journal and database file header dumps, produced using MUPIP JOURNAL -SHOW=HEADER and DSE DUMP -FILE -ALL, respectively, report whether an all-zero IV (NULL\_IV) is in use.

Additionally, ZSHOW "D" displays the name of the encryption key and IV type for each file device that has its input, output, or both streams using encryption.

Please note:

## Database

- For databases created with prior versions that used an all-zero IV, GT.M continues to provide an empty IV for database blocks and journal records (see the release note for GTM-6310 on upgrading to non-zero IVs).
- To downgrade a database that uses non-zero IVs to a database that uses zero IVs, extract the contents with MUPIP EXTRACT and use MUPIP LOAD to load the contents into a database created with a prior GT.M version that uses zero IVs. If the MUPIP LOAD targets a prior version of GT.M, the extract needs to either specify FORMAT=ZWR or FORMAT=BIN -NULL\_IV.
- While GT.M releases that provide non-zero IVs can use databases from versions of GT.M that use zero IVs, prior versions of GT.M that use zero IVs cannot process files that have non-zero IVs (e.g., while MUPIP LOAD can process binary extracts in the prior GDS level 8 format, previous GT.M versions cannot process GDS level format 9 binary extracts).  
(GTM-8117) 🟢
- The `gtm_nontprestart_log_delta` and `gtm_nontprestart_log_first` environment variables control whether and how GT.M sends to the syslog NONTPRESTART messages that supply information on non-TP "mini transaction" restarts. If `$gtm_nontprestart_log_delta` is a non-zero integer, GT.M uses the value as a sampling interval for the messages, so a value of one (1) produces a report for every non-TP restart, a value of two (2) means a report for every other non-TP restart, etc. If `$gtm_nontprestart_log_delta` is defined as described and `$gtm_nontprestart_log_first` is also a non-zero integer, GT.M reports the first `$gtm_nontprestart_log_first` non-TP restarts before reporting samples as defined by `$gtm_nontprestart_log_delta`. Previously, GT.M did not provide detailed information for non-TP mini-transactions, but it did provide cumulative restart non-TP mini-transaction restarts using the NR0, NR1, NR2, NR3 fields in the ZSHOW "G" output. In addition, if the restart occurs due to a conflict in the global directory tree, the TPRESTART message reports "\*DIR" for the `gbl` field. Previously, in that case, it displayed only "^" with no global variable name. TPRESTART messages include the subscript of the contested global. Previously, it reported global names without any subscript information. `VIEW [NO]LOGN[ONTP][:intexpr]` allows a process to dynamically change the logging to the syslog of NONTPRESTART messages established at process startup by the environment variables `gtm_nontprestart_log_delta` and `gtm_nontprestart_log_first`. `VIEW "NOLOGNONTP"` turns off the logging of NONTPRESTART messages to the syslog. `VIEW "LOGNONTP[:intexpr]"` turns on logging of NONTPRESTART messages to the syslog. If no `intexpr` is specified, GT.M uses the value of environment variable `gtm_nontprestart_log_delta`, if it is defined, and one (1) otherwise (that is, log every transaction restart). A negative value of `intexpr` turns off the logging of NONTPRESTART messages. Note that it is not possible to perform the operations of `gtm_nontprestart_log_first` with `VIEW "LOGNONTP[:intexpr]"`. (GTM-8190) 🟢
- Encryption initialization is faster for databases with multiple encrypted regions, with the improvement more noticeable as the number of regions increases. To facilitate this performance improvement, GT.M requires the encryption configuration file (referred to by `$gtmencrypt_config`) to specifically associate the key used by each database file with that database file name, raising an error if an entry mapping the key to the file does not exist. Previously, GT.M accepted any key with a hash matching that in the database file header, even a key that was not specifically mapped to a database file name that used it. This enhancement benefits from changes to the API of the encryption plugin.  
(GTM-8336)







- Update helper writer processes increase replication throughput, decrease backlog, and improve manageability:
  - Update writer helper processes start flush timers, participate in epoch tapering, and perform timed epochs. Previously writer helpers only flushed dirty blocks.
  - The Update Process no longer starts or restarts flush timers on a database if there is another process with a timer, e.g. an update writer helper. Previously the Update Process typically would establish a timer, preempting writer helpers from performing them.
  - The Receiver Server actively notifies the Update Process when there is work to be done. Previously the update process periodically polled for updates when the receive pool backlog reached zero, potentially leading to short periods where the backlog increased.
  - MUPIP REPLICATE -RECEIVER -SHUTDOWN shuts down update helper processes left running after the receiver shutdown abnormally. Previously a prior MUPIP REPLICATE -RECEIVER -SHUTDOWN -HELPERS was required to terminate each such process. (GTM-8340) 🟢
- \$ZTRIGGER() and MUPIP TRIGGER delete all, "-\*", operations delete malformed trigger records, identifying such malformed trigger records with TRIGDEFBAD warnings. Previously trigger delete ignored some malformed triggers making them undeletable. In some cases, attempts to view (SELECT) these malformed triggers failed, making them invisible as well. The workaround was to extract all triggers to a file, add a trigger for each global variable with a trigger, followed by a delete all, "-\*". This deleted all triggers and restored the desired triggers in one transaction. (GTM-8399)
- \$ZTRIGGER() and MUPIP TRIGGER appropriately delete triggers as specified; previously under rare circumstances, they silently failed to do so. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8457)
- Processes that have read-write access to a journaled database file, but have not yet performed any updates to it (i.e., processes that have not yet opened the journal file for that database file), cooperate correctly with processes that have read-write access and have performed updates. Previously, it was possible for the former to pick up all available slots for flushing dirty buffers (the default being two slots), resulting in an out-of-design situation where dirty buffers did not get flushed in a timely manner, in turn resulting in the Source Server terminating with a SEQNUMSEARCHTIMEOUT error. Additionally, the Source Server issues an alert every 50 seconds (instead of 10 seconds previously). (GTM-8489)
- KILL of a global outside of a TP transaction following a large TP transaction with certain characteristics by the same process works correctly; previously this unusual sequence could terminate a process with a segmentation violation (SIG-11). Database structural integrity was not at risk from this behavior. (GTM-8499)
- GT.M does not internally deadlock in the presence of certain rare, asynchronous events, as it previously could in releases V6.0-003 through V6.2-002A. This issue was only observed in the GT.M development environment, and was never reported by a user. Note that the design of GT.M precludes deadlocks in normal usage. (GTM-8500)

## Database

- MUPIP, LKE, and DSE no longer issue NOCHLEFT fatal errors while handling errors in database disconnect which occur during process shutdown. (GTM-8501)
- GT.M processes no longer deadlock when a trigger causes a fatal error, under certain unusual conditions. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8506)

---

## Language

- ZMESSAGE accepts an indirect argument. Previously this syntax produced a GTMASSERT. (GTM-6114)
- In an M mode process, a WRITE to a sequential device after a SET \$X to a value greater than the device WIDTH or a reduction in WIDTH to less than the current \$X acts as if the first character caused \$X to exceed the WIDTH inducing an immediate WRAP, if WRAP is enabled. Previously, \$X exceeding WIDTH in an M mode process caused a SYSTEM-E-ENO14, Bad address error. (GTM-7060) 
- \$ZWRITE() accepts a second intexpr which specifies the direction of the conversion. No value or a zero value converts the first argument to ZWRITE format and a non-zero value converts the first argument from ZWRITE format to a string with embedded non-graphic characters. In addition, \$ZWRITE() now takes the canonical or non-canonical nature of its first argument into account; previously it attempted to treat non-canonical numeric values as numbers rather than strings, which caused errors and inappropriate results. Also if all its arguments are literals, \$ZWRITE() evaluates to a literal constant at compile time. Previously \$ZWRITE only accepted a single argument, converting only to ZWRITE format, and always evaluated at run time. (GTM-7604) 
- The GT.M compiler resolves some expressions involving string literals at compile time, reducing both code size and run time CPU requirements. Currently the compiler does this for concatenation operations, \$[Z]ASCII(), \$[Z]EXTRACT(), \$[Z]PIECE(), and \$ZSUBSTR(). For code compiled in UTF-8 mode, this means that the compiler can report BADCHAR warnings for strings that are not legal UTF-8 strings, where previously it did not. The compiler flags these as warnings, rather than as errors, since the possible difference between the compile time and run time setting of [NO]BADCHAR dictates that the compiler cannot with certainty determine whether strings containing illegal UTF-8 characters are actually errors (for example, they may contain binary data). When the compiler reports BADCHAR warnings, it defers the computation to run time, instead of attempting to precompute the result. Note that sometimes a single character with an invalid encoding can cause multiple warnings, for example, when a string that is not a valid UTF-8 string appears inside a function such as \$PIECE(). Regardless of any compile time warnings, there is no change to run time functionality of any M application code as a result of this change. (GTM-7762) 
- Using VIEW "JOBPID":1 does not issue an error when the JOB command parameters ERROR and OUTPUT point to the same file; previously, while the effect of the VIEW command was not documented, using the same file name for ERROR and OUTPUT and VIEW "JOBPID":1 resulted in an error. (GTM-8034) 
- \$ZCONVERT(...,"T") works in M mode; previously it only worked in UTF-8 mode. (GTM-8038)
- Processes no longer terminate with a segmentation violation (SIG-11) when:
  - executing the ZSYSTEM command with an overly long value of the environment variable SHELL;  
and

## Language

- initializing the reference implementation of the encryption plugin with an overly long value of the environment variable GNUPGHOME.

These issues were only observed in the GT.M development environment, and were never reported by a user. (GTM-8112)

- `$ZCONVERT()` raises an `ICUERROR` if GT.M is unable to correctly access the ICU library; previously the process terminated with a `GTMASERT`. It also runs faster than it did previously. (GTM-8215)
- `%PEEKBYNAME()` provides a stable interface to `$ZPEEK()` that uses control structure field mnemonics. `$ZPEEK()` provides a read-only mechanism to access selected fields in selected control structures in the address space of a process, including process private memory, database shared memory segments and Journal Pools. Although application code can call `$ZPEEK()` directly, such direct access must use numeric arguments that can vary from release to release. Access using `%PEEKBYNAME` makes application code more stable across GT.M releases. For more information, refer to Additional Information for `GTM-8296- %PEEKBYNAME()`. (GTM-8296) ✓
- If the `JOB` command finds the `ZWRITE` representation of an lvn consisting of a its full name, its subscripts, corresponding value, quotes and the "=" sign, exceeding 1MiB, it produces a `JOBLVN2LONG` error in both `JOB'ing` and `JOB'd process'` error stream. This message includes the attempted length and maximum limit. Previously, the `JOB'ing` process would issue a `JOBLVN2LONG` error without the lvn length details, and `JOB'd process` would issue a `JOBLVNDetail` with the length information which the `JOBLVN2LONG` currently displays. (GTM-8297)
- The `WRITE /TLS("renegotiate"[, [tlsid][, options]])` command on a server socket allows applications to request a TLS renegotiation. In the command:
  - The second and fourth arguments are unspecified.
  - `tlsid` refers to the name of a section in the configuration file specified by the `gtmencrypt_config` environment variable. If `tlsid` is not specified, GT.M creates a virtual section by appending "-RENEGOTIATE" to the `tlsid` used to enable TLS on the socket. If no section named `tlsid` is present in the configuration file, GT.M creates a virtual section with that name for the life of the process.
  - Supported configuration file options passed in the command are (case-sensitive): `verify-depth`, `verify-level`, `verify-mode`, `session-id-hex`, and `CAfile`.
  - When `tlsid` is specified, any options in the command take precedence over options of the same name specified in the configuration file section.

The `WRITE /TLS("renegotiate",...)` command ignores options other than those listed above. The options remain in effect for the socket after the renegotiation. Any virtual section remains available in the current process.

Renegotiation requires the suspension of application communication and the application must read all pending data before initiating a renegotiation. This means that in the communication protocol used, both parties must be at a known state when renegotiating keys. For example, in

GT.M replication, one party sends a renegotiation request and waits for an acknowledgement before initiating the renegotiation.

The configuration file can specify options, although the `WRITE /TLS("renegotiate",...)` command can override them. Note that configuration file options may be useful even without the renegotiate command.

- The `CAfile` option when specified for a server connection either in a `tlsid` level configuration file section or for the renegotiate command allows the server to inform the client of acceptable certificate authorities via the OpenSSL function `SSL_set_client_CA_list()`. The determinant definition for the acceptable list of certificate authorities sent to the client comes in descending order of priority from the one specified by the `WRITE /TLS("renegotiate",...)` command, the one specified by the `CAfile` value in the `tlsid` section used to establish the TLS connection, and finally that specified at the `tls` level.
- The `verify-level` option takes a string value to specify any additional certificate verification in addition to the basic OpenSSL verification. The only value currently accepted is "CHECK" which requests additional checks on the results of the basic OpenSSL certificate verification. A leading exclamation mark ("!") disables a `verify-level` option. The `verify-level` options specified at lower levels are merged with those options already specified at higher levels. CHECK is enabled by default for all TLS connections.
- The `session-id-hex` option takes a string value which is used to set the `SSL_session_id` context for server sockets, which may be specified in the `tlsid` section of a config file or on `WRITE /TLS("RENEGOTIATE",...)`. See the OpenSSL man page for `SSL_set_session_id_context` for usage details. The value should consist of hexadecimal digits representing the desired value. Application code can call the `%UTF2HEX` utility routine to translate a character string to the corresponding string of hexadecimal digits. If neither the command or the associated `tlsid` section in the configuration file specify a `session-id-hex` option when creating the socket, GT.M uses the current `tlsid`, translated into hexadecimal digits.

The default for the configuration file option `verify-mode` is `SSL_VERIFY_PEER`. Previously it was `SSL_VERIFY_NONE` for TLS enabled sockets.

Including "SESSION" in the fourth argument of `$ZSOCKET "TLS"` returns information related to SSL sessions including information about renegotiations. The following is an example of the information returned in addition to the information previously returned:

```
|S:RENSEC:1,RENTOT:1,SESSID:A9EB18B4731B2E4ABA572C8386213
4C67C9561597D5FAF47CDD5B866B77215FF,SESEXP:Thu Jun 4 21:07:11 2015
```

where "|S:" denotes this piece contains session information, "RENSEC:" indicates whether secure renegotiation is available (1) or not (0), "RENTOT:" gives the current total number of renegotiations done on this socket, "SESSID:" shows the session id in hexadecimal, and "SESEXP:" indicates when the session expires in the local timezone.


## Language




Including "OPTIONS" in the fourth argument of \$ZSOCKET "TLS" now returns the verify mode after the TLS options (that is, "|O:hexdigits") as a comma and two hexadecimal digits. The verify mode values are defined in openssl/ssl.h.

Including "ALL" in the fourth argument of \$ZSOCKET "TLS" returns all available information.

A successful WRITE for a TLS enabled socket with an argument larger than the ZBFSIZE for the socket is not considered an error. Previously, such a WRITE indicated an error with the following \$DEVICE value: "1,Unknown error -1" or if IOERROR="TRAP", \$ZSTATUS included:

```
%GTM-E-SOCKWRITE, Write to a socket failed,%GTM-I-TEXT, Unknown error -1
```

(GTM-8302) 

- To improve performance of certain UTF-8 string functions, when a string 32-bytes or longer is passed as the first parameter to \$EXTRACT() and \$FIND() calls that have three parameters, GT.M caches information regarding transitions between blocks of ASCII characters (\$CHAR(0) through \$CHAR(127)) and non-ASCII UTF-8 characters (\$CHAR(128) & up). Actual improvement in application performance can range from unobservable to dramatic depending on the prevalence of this use case. As there are a couple of tuning parameters for this optimization (with defaults that are reasonable in our judgement), if you have code that benefits from this optimization, please contact your GT.M support channel for help in experimenting with the tuning the parameters. (GTM-8352) 
- When a process accesses a trigger definition with ZPRINT or \$TEXT() outside of a TP transaction and it has previously fetched the definition with no intervening execution of a revised trigger definition, GT.M returns information cached by that prior access. Previously, if another process had deleted or modified the definition after its last source access and with no intervening execution, ZPRINT gave a TRIGNAMF error and \$TEXT() returned an empty string. Note that accessing the source code within a TP transaction (either implicit or explicit) always uses the current definition. (GTM-8389) 
- When interrupted by a signal requesting termination, such as SIGTERM, while writing to the principal device, GT.M prints that all subsequent rundown messages correctly. Previously, there was a small window in writing to the principal device when an external signal could result in corrupt rundown messages. This issue was discovered in the GT.M development environment and was never reported by any user. (GTM-8397)
- Calls to \$TEXT() for which the compiler can determine the source code line at compile time it compiles as literal strings in the object code. Specifically, this optimization occurs when the entryref refers to a line in the current routine, and no component of the entryref uses indirection. (GTM-8404) 
- ZSHOW accepts atomic indirection in the second expression of its argument. Previously it incorrectly gave an error for such an attempt. (GTM-8407)
- ZPRINT handles an unusual case when it appears within trigger code and also within an XECUTE or indirection and the target routine has become empty (has zero length). Previously, such

circumstances intermittently produced a ZLINKFILE error. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8417)

- \$TEXT() returns the empty string for ^GTM\$DMOD and ^GTM\$CI; previously these \$TEXT() arguments, which represent respectively the base frames for mumps -direct and a call-in and could be derived from \$STACK(), \$ZPOSITION or ZSHOW "S", produced an RPARENMISSING error. (GTM-8431)
- ZSHOW expr:gvn stores continuations of information that do not fit in the maximum record size as immediate descendants, using ordinal subscripts starting at one (1), of the node holding the beginning of the information. This facilitates identifying when multiple nodes need to be reassembled to find the entire item. In addition, the ^%ZSHOWVTOLCL utility program restores ZSHOW "V":gvn data into its original local variables. The utility needs to be invoked with \$ECODE set to the empty string ("") and cannot restore a local variable with the same name (%ZSHOWvbase) as the parameter in its formal list. Other ZSHOW information typically fits within common database record size limitations. Previously, ZSHOW expr:gvn stored continuation nodes at the same level as other nodes. Although this change is not backward compatible, it facilitates automated restoration even of nodes exceeding the maximum record size of the global, which was not the case previously (GTM-8433) 🟢
- When a GT.M process receives a MUIP STOP while doing a READ on a SEQUENTIAL, FIFO, or PIPE device in NOFIXED, M mode, the process terminates. Previously, this combination could cause the process to hang. (GTM-8435)
- GT.M produces an appropriate error for an inappropriately long device name in the argument of OPEN, USE and CLOSE commands; previously such an argument could cause a segmentation violation (SIG-11). (GTM-8440)
- When interrupted by a signal requesting termination, such as SIGTERM GT.M handles the termination appropriately. In GT.M V6.2-002 and V6.2-002A, if the interrupt happened during certain small windows of execution, it was possible for the process to terminate abnormally, hang, or exhibit other incorrect behavior under rare conditions. This issue was discovered in the GT.M development environment and was never reported by any user. (GTM-8448)
- \$ZGETJPI() reports large values appropriately; previously large times could wrap. (GTM-8450)
- A JOB command with PASSCURLVN correctly passes alias containers that contain KILL'ed original aliases. Previously, JOB command with PASSCURLVN could fail with INDEXTRACHARS error in such a case. (GTM-8465)
- \$ZSEARCH() avoids a possible deadlock that could cause a process to hang indefinitely. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8475)
- GT.M handles errors from triggers on globals that span nodes or regions when the error handler does not clear \$ECODE. Previously, when a trigger's error handler did not clear \$ECODE for an update to a global that spanned regions or nodes, GT.M issued a fatal GTMASSERT2 error. (GTM-8476)

## Language

- GT.M no longer issues an inappropriate TLVLZERO error when executing a TCOMMIT command, as it previously did under certain very rare conditions. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8477)
- MUPIP JOURNAL -EXTRACT leaves journal files in a wholesome state when run where a corresponding database was not shutdown in an orderly fashion, due to for example a kill -9, attached processes exceeding 32Ki, etc. Previously, under these somewhat unusual conditions MUPIP JOURNAL -EXTRACT left the journal files in a state where they could not be applied to recover the database. (GTM-8483)
- M compilations, trigger compilations, and online integrity checks no longer result in segmentation faults (SIG-11) in rare cases. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8484)
- GT.M correctly issues a ZROSYNTAX error when transitioning from a \$ZROUTINES with auto-relink enabled to a \$ZROUTINES that has non-existent directory paths. Previously, GT.M terminated with a segmentation violation (SIG-11). (GTM-8486)
- GT.M treats VIEW 1 and VIEW 0 as errors. Previously, it treated these constructs as VIEW "GDSCERT":1 and VIEW "GDSCERT":0 commands respectively. (GTM-8490)
- On AIX, for non-wildcard searches, \$ZSEARCH() exhibits performance similar to that of other platforms. The correction made with GTM-8237 in V6.2-002 exposed decreased and erratic performance on AIX of the POSIX service used to perform such searches. [AIX] (GTM-8491)
- Unless overridden with a GBLDIR jobparameter, processes started with the JOB command use the global directory specified by the \$ZGBLDIR of the process executing the JOB command (as documented). Previously, these processes incorrectly used the value of the environment variable gtmgbldir in the environment of the parent executing the JOB command. As a side effect, since the process executing the JOB command would have previously validated the path to the global directory, a JOB'd processes does not terminate with a segmentation violation (SIG-11), as it previously did, if \$gtmgbldir is longer than GT.M's limit of 255 bytes for the path to the global directory. You should check all occurrences in your application where it executes JOB commands without GBLDIR jobparameters when its values of \$ZGBLDIR and \$gtmgbldir differ. In addition, processes no longer terminate with a segmentation violation (SIG-11) when: executing the ZSYSTEM command with an overly long value of the environment variable SHELL; and initializing the reference implementation of the encryption plugin with an overly long value of the environment variable GNUPGHOME. These issues were only observed in the GT.M development environment, and were never reported by a user. (GTM-8512)



---

## System Administration

- MUPIP JOURNAL commands (e.g. ROLLBACK, RECOVER, VERIFY, SHOW etc.) accept the -PARA[LLEL][=n] qualifier to specify the number of parallel threads (for backward processing) and parallel processes (for forward processing). Omitting the qualifier or specifying a value of one (1) defaults to a single process with no threads. Omitting the value or specifying a value of zero (0) specifies one thread or process per region. A value greater than one (1) specifies the maximum number of concurrent threads or processes MUPIP should use, although it never uses more than one per region. If the number of regions exceeds the specified value, MUPIP allocates threads or processes in an order determined by timestamps in the journal records. The environment variable gtm\_mupjnl\_parallel provides a value when the command has no explicit -PARALLEL qualifier; when defined with no value gtm\_mupjnl\_parallel acts like -PARALLEL with no value. When the -PARALLEL qualifier (or the gtm\_mupjnl\_parallel environment variable) specifies the use of parallel processes in the forward phase of a MUPIP JOURNAL command, MUPIP may create temporary shared memory segments and/or extract files (corresponding to -extract or -losttrans or -brokentrans qualifiers) and clean these up at the end of the command; however an abnormal termination such as a kill -9 might cause these to be orphaned.

Journal extract files (created by specifying one of -extract or -brokentrans or -losttrans to a MUPIP JOURNAL command) contain journal records sorted by sequence number (token\_seq/jsnum) then by update order (updnum) for all regions which were replicated/journaled - in other words: in the exact order their corresponding updates happened in time. In prior versions of GT.M, these files were not sorted which meant applying lost transaction files (for example) required first sorting the file to correspond to the update-order before applying them.

In addition, a MUPIP JOURNAL -SHOW=HEADER has default of -NOVERIFY if no other action qualifiers (-EXTRACT, -RECOVER, -ROLLBACK, -SHOW) are specified. This speeds up the command in the default case (no verification of the entire journal file occurs). Note that specifying -VERIFY explicitly still does the verification as requested. (GTM-5007) 🟢

- See GTM-7768. (GTM-5726)
- With the -ENCRYPT flag, MUPIP REORG changes the encryption key of a database, or encrypts an unencrypted database, while the database continues to be used by applications. Whether or not the prior encryption uses non-zero initialization vectors (IVs), database blocks encrypted with the new key use non-zero IVs (see GTM-8117). The syntax is:

```
MUPIP REORG -ENCR[YPT]=<key> -REGION <region-list>
```

where <key> is a key provided by MUPIP to the encryption plugin. The reference implementation of the plugin expects a key with the specified name in the encryption configuration file identified by \$gtmencrypt\_config. The configuration file must contain an entry in the database section for each database file mapping to a region specified in <region-list> that names the specified key as its key. The -ENCRYPT flag is incompatible with all other command line flags of MUPIP REORG, and performs no operation other than changing the encryption key. If the specified key is already the

encryption key of a database region, MUPIP REORG -ENCRYPT moves on the next region after displaying a message (on stderr, where MUPIP operations send their output).

As MUPIP REORG -ENCRYPT must read, re-encrypt, and write every encrypted block in each database file, its operation will take a material amount of time on the databases of typical applications, and furthermore will add an additional IO load to the system on which it runs. You can use the environment variable `gtm_poollimit` to ameliorate, but not eliminate, the impact, at the cost of extending execution times. To minimize impact on production instances, FIS recommends running this operation on replicating secondary instances, rather than on originating primary instances.

MUPIP REORG -ENCRYPT switches the journal file for each database region when it begins operating on it, and again when it completes, and also records messages in the syslog for both events. Note that the detailed journal extract format is now level 8.

As is the case under normal operation when MUPIP REORG -ENCRYPT is not active, journaled databases are protected against system crashes when MUPIP REORG -ENCRYPT is in operation: MUPIP JOURNAL ROLLBACK / RECOVER recovers journaled database regions (databases that use NOBEFORE journaling continue to require FORWARD RECOVER / ROLLBACK).

Since a database file utilizes two keys while MUPIP REORG -ENCRYPT is underway, the database section of the configuration file provides for a single database file entry to specify multiple keys. For example, if the keys of database regions CUST and HIST, mapping to database files `cust.dat` and `hist.dat` in directory `/var/myApp/prod`, are to be changed from `key1` to `key2` using the command:

```
MUPIP REORG -ENCRYPT=key2 -REGION CUST,HIST
```

then the database section of the configuration file must at least have the following entries:

```
database: {
  keys: ({
    dat: "/var/myApp/cust.dat";
    key: "key1";
  }, {
    dat: "/var/myApp/cust.dat";
    key: "key2";
  }, {
    dat: "/var/myApp/hist.dat";
    key: "key1";
  }, {
    dat: "/var/myApp/hist.dat";
    key: "key2";
  })
};
```

In other words, each database file entry can have multiple keys, and a key can be associated with multiple database files. With a configuration file that has multiple keys associated with the same database file, MUPIP CREATE uses the last entry. Other database operations use whichever key has a hash matching one in the database file header, reporting an error if no key matches. To improve

efficiency when opening databases, you can delete entries for keys that are no longer used from the configuration file.

MUPIP REORG -ENCR[YPT] can encrypt an unencrypted database only if the following command:

```
MUPIP SET -ENCRYPTABLE -REGION <region-list>
```

has previously marked the database "encryptable".

The command requires standalone access to the database. Just as encrypted databases use global buffers in pairs (for encrypted and unencrypted versions of blocks), a database marked as encryptable has global buffers allocated in pairs (i.e., the actual number of global buffers is twice the number reported by DSE DUMP -FILEHEADER) and requires correspondingly larger shared memory segments. To revert unencrypted but encryptable databases back to "unencryptable" state, use the command:

```
MUPIP SET -NOENCRYPTABLE -REGION <region-list>
```

The above command also requires standalone access, and the result depends on the state of the database. It:

- is a no-op if the database is encrypted;
- is disallowed if the database is partially (re)encrypted; and
- prohibits encryption if the database is not encrypted.


Under normal operation, a database file has only one key. Upon starting a MUPIP REORG -ENCRYPT to change the key, there are two keys, both of whose hashes GT.M stores in the database file header. With a MUPIP REORG -ENCRYPT operation underway to change the key, normal database operation can continue, except for another MUPIP REORG -ENCRYPT or MUPIP EXTRACT in binary format. Other MUPIP operations, such as MUPIP BACKUP and MUPIP EXTRACT in ZWR format can occur. A MUPIP REORG -ENCRYPT operation can resume after an interruption, either unintentional, such as after a system crash and recovery, or intentional, after an explicit MUPIP STOP of the MUPIP REORG -ENCRYPT process. To resume the REORG operation, reissue the original command, including the key parameter. (Note that supplying a key other than the one used in the original command results in an error.)




After the MUPIP REORG -ENCRYPT process completes, subsequent MUPIP REORG -ENCRYPT operations on the same region(s) are disallowed until the following command is run:

```
MUPIP SET -ENCRYPTIONCOMPLETE -REGION <region-list>
```

The reason to block subsequent MUPIP REORG -ENCRYPT operations upon completion of one is to provide time for a backup of the entire database before enabling further key changes. MUPIP SET -ENCRYPTIONCOMPLETE reports an error for any database region for which MUPIP REORG -ENCRYPT has not completed.

The above changes necessitated certain alterations of the encryption plugin API (for other API changes, see GTM-8336).

(GTM-6310) 

- See GTM-8137. (GTM-6317)
- MUPIP EXTRACT and MUPIP JOURNAL -EXTRACT are faster. While improvements you see will depend on the circumstances of your configuration and environment, in testing in the GT.M development environment, GO and ZWR formatted extracts completed in as much as three to four times faster than before. (GTM-6388)
- On AIX, GT.M uses the AIX provided ICU libraries when `gtm_icu_version` is not defined. Previously GT.M required users to build the ICU libraries. [AIX] (GTM-6858) 
- With the `-ZEROBACKLOG` qualifier, a MUPIP REPLICATE `-SOURCE -SHUTDOWN` command shuts down the Source Server either when the backlog goes to zero, or the timeout expires, whichever occurs first. (GTM-6928) 
- MUPIP JOURNAL -ROLLBACK recognizes the `-FORWARD` qualifier to specify it apply update records in journal files to backed up copies of database files in order to bring them to the same state that MUPIP JOURNAL -ROLLBACK -BACKWARD "" would bring crashed database files. Unlike MUPIP JOURNAL -ROLLBACK -BACKWARD, MUPIP JOURNAL -ROLLBACK -FORWARD accepts either a comma-delimited list of region names or "", indicating the journal files associated with all regions in the current Global Directory. A MUPIP JOURNAL -ROLLBACK -FORWARD "" command does what a MUPIP JOURNAL -RECOVER -FORWARD "" would do except that the -ROLLBACK also updates sequence number related fields in the database file header, and ensures update serialization across regions - MUPIP JOURNAL -RECOVER can leave one database region ahead of another region and cannot ensure database Consistency across regions, whereas MUPIP JOURNAL -ROLLBACK can ensure Consistency. Databases recovered with MUPIP JOURNAL -ROLLBACK can therefore be used in replicated instances. Note that, in the context of -RECOVER and -ROLLBACK, the "" indicates the use of all the appropriate journal files in all the replicated regions and the quotes prevent inappropriate expansion by the OS shell. MUPIP JOURNAL -ROLLBACK -FORWARD leaves the journaling state turned off in database files (as does MUPIP JOURNAL -RECOVER -FORWARD), which in turn means that replication is also turned off; journaling should be re-enabled, and replication turned on, before database files are used in environments where they can be updated, but can be left off if subsequent access is read-only. After a MUPIP JOURNAL -ROLLBACK -FORWARD, the replication instance file needs to be recreated as part of turning replication on in the recovered database. MUPIP JOURNAL -ROLLBACK -FORWARD can use both before-image and nobefore-image journal files. See the More Information section for details on qualifier use. In addition, MUPIP JOURNAL -ROLLBACK -BACKWARD output always includes the RLBKJNSEQ message; previously the output only included this if the rollback changed the database (i.e., the operation was not a no-op). Also, MUPIP JOURNAL -RECOVER -BACKWARD -BEFORE works even when -SINCE is not specified; previously such a command terminated with a JNLTMQUAL1 error. For more information, refer to Additional Information for GTM-7291 - MUPIP JOURNAL -ROLLBACK qualifiers. (GTM-7291) 
- GT.M accepts all ICU versions reported by "icu-config -version" for `gtm_icu_version`. Previously GT.M required the format of `gtm_icu_version` to be `<ICU Major Version> "." <ICU Minor Version>`

with no trailing digits, and required ICU versions 49 & higher (i.e., with the new ICU numbering scheme) to be specified as 4.9, 5.0, etc. (GTM-7375) 🟢

- An improvement in database replication communication between Source and Receiver Servers reduces delays. Previously, a busy Source Server, for example, one searching through journal files to find the transaction at which to resume communication with a replicating secondary instance with a large backlog, could ignore communications long enough for the Receiver Server to close the connection and begin a reconnection cycle, adding to the persistence of the backlog. Note that a Source Server started with `-JNLFILEONLY` always reads from the journal files. (GTM-5726) (GTM-7768)
- MUPIP SET recognizes the `-SPIN_SLEEP_LIMIT=n` where n is decimal maximum number rounded up to the nearest power of two and turned into a hexadecimal mask that determines the maximum number of nanoseconds for processes to sleep while waiting to obtain critical sections for shared resources, principally those involving databases. The default is zero (0) which causes the process to return control to the UNIX kernel to be rescheduled with no explicit delay. When the value is non-zero, the process waits for a random value between zero (0) and the maximum value permitted by the mask. DSE CHANGE `-FILEHEADER -SPIN_SLEEP_MASK` provides a means to directly set the hexadecimal value of the mask, which appears in DSE DUMP `-FILEHEADER` output with the label "Spin sleep time mask." Previously, processes always rescheduled themselves with no delay. In addition, MUPIP SET (and DSE) accept a 0 value for `-SLEEP_SPIN_COUNT`, which eliminates the sleep loop in the mutual exclusion (mutex) facility so processes go straight from a hard spin to a queued wait. Previously, the `-SLEEP_SPIN_COUNT` qualifier was not recognized by MUPIP SET and processes that exhausted the hard spin count always did at least one (1) rescheduling operation. Except on the advice of your GT.M support channel, FIS recommends leaving the default values unchanged in production environments, until you have data from testing and benchmarking that demonstrates a benefit from a change. If none of its qualifiers are out of the range from minimum to maximum, MUPIP SET processes all qualifiers for each region applying them only if all are appropriate for the region and otherwise warning of any issues. Previously, MUPIP SET stopped at the first error and if some qualifiers required standalone access and others did not, applied only those requiring standalone access and silently ignored those that did not. If you have scripting that checks for correct completion of a MUPIP SET command, no changes are required to accommodate this change. However, if your scripting checks for and takes actions based on errors reported by MUPIP SET, you should test your script and revise it as needed. (GTM-7809) 🟢
- MUPIP JOURNAL ROLLBACK fixes DBDANGER situations and therefore neither issues DBDANGER messages to syslog nor freezes an instance. Previously, MUPIP JOURNAL ROLLBACK could send DBDANGER messages to syslog even though it would fix the problem, and, when using the Instance Freeze facility with a DBDANGER in the custom errors file, a ROLLBACK issuing a DBDANGER message would freeze until manually unfrozen. (GTM-7831)
- GT.M replication uses the operating system and network to perform flow control. Previously, the Source and Receiver Servers performed flow control, as a consequence of which the Receiver Server could flood the socket connection with flow control messages, leading to a large number of log messages and an occasional hang in replication processing. (GTM-7838)

- MUPIP LOAD accepts negative values in ZWR format input such as that produced by %GO. Previously, such a value caused a LOADFILERR. The workaround was to use GO format or edit the file to add a pair of double-quotes around the negative value. (GTM-8020)
- GT.M can be configured to permit more than 32,767 processes to concurrently access a database file. In a replicated environment, to permit one or more database files to be concurrently accessed by more than 32,767 updating processes also requires the replication instance file to be configured to permit concurrent access by more than 32,767 processes. The default behavior is to limit the number of processes accessing a database file or instance file to 32,767. This permission is effected by the QDBRUNDOWN flags in database file headers and in replication instance files. When an open database file or replication instance file with QDBRUNDOWN set is first concurrently accessed by more than 32,767 processes, GT.M (a) logs a NOMORESEMCNT message in the system log, and (b) stops counting the number of attached processes. This means that GT.M cannot recognize when the number of attached processes reaches zero (0) in order to release the corresponding shared resources, and therefore requires explicit manual clean up of resources for an orderly shutdown. Previously, exceeding 32Ki attachments to shared resources caused a DBFILERR, CRITSEMFAIL error with an ERANGE status for database and replication instance files, as it still does when QDBRUNDOWN is not set.

Except in application configurations that require it, FIS recommends not setting QDBRUNDOWN. Not setting QDBRUNDOWN allows GT.M to clean up resources, instead of putting the burden on the operational procedures. Where GT.M cannot perform an orderly shutdown, an explicit, clean shutdown must be performed as follows:

- Replicated instances require a MUPIP JOURNAL -ROLLBACK -BACKWARD "" executed after the MUPIP REPLICATE SOURCE -SHUTDOWN command (remember that even instances receiving a replication stream have one or more Source Servers).
- Database files that are journaled but not part of a replication instance require a MUPIP JOURNAL -RECOVER -BACKWARD command.
- Database files that are not journaled (and hence not replicated) require a MUPIP RUNDOWN command.

MUPIP REPLICATE -INSTANCE\_CREATE -[NO]QDBRUNDOWN controls the QDBRUNDOWN setting of a replication instance file when it is created. For an existing replication instance file, requiring stand-alone access (i.e., the instance must not have an existing Journal Pool), MUPIP REPLICATE -EDITINSTANCE -[NO]QDBRUNDOWN controls the QDBRUNDOWN setting. Specifying -QDBRUNDOWN turns it ON, and -NOQDBRUNDOWN turns it OFF. MUPIP REPLICATE -EDITINSTANCE -SHOW displays the current QDBRUNDOWN setting of an instance file as "HDR Quick database rundown is active", reported as TRUE or FALSE. Note that QDBRUNDOWN is an existing setting in database files. See the release note for GTM-8296 on accessing the QDBRUNDOWN setting using %PEEKBYNAME().

In support of this enhancement:

- As all processes must use the same setting of QDBRUNDOWN, changing the QDBRUNDOWN setting requires standalone access; previously it did not for database files, and was not meaningful for replication instance files.
- MUPIP RUNDOWN issues a DBRDONLY error if it encounters a database file with read-only permissions (no read-write permission) and the database shared memory indicates that the number of attached processes exceeded 32Ki at some point after it was opened. Databases that exceed the 32Ki counter need a process with read-write ability to perform the required rundown/recover/rollback.

The work to develop this enhancement also addressed several issues. These issues were only observed in the GT.M development environment, and were never reported by a user.

- Endian conversion works correctly in the rare case that the database is opened by a process after MUPIP ENDIANCVT starts converting the endianness of a database, and before it blocks other processes from opening the database. Previously, this could result in a database with structural damage requiring manual repair.
- MUPIP JOURNAL -SHOW=HEADER on a journal file of a supplementary instance reports correct stream sequence numbers (displayed as "**Stream *i* : Start RegSeqno**" or "**Stream *i* : End RegSeqno**" where *i* identifies a stream, from 0 through 15). Previously the stream sequence numbers were not correctly maintained if the instance had helper writer processes (spawned off by the Receiver Server when started with MUPIP REPLICATE -RECEIVER -HELPERS).
- When it encounters interprocess communication (IPC) shared memory & semaphores for a database file for which it does not have read-write permissions and which is currently open by no processes with read-write access, a MUPIP RUNDOWN with no arguments issues a DBRDONLY error message, and leaves both semaphores and shared memory intact. Previously, in this case it removed the semaphores while leaving the shared memory intact, an out-of-design condition which could result in errors to other processes accessing that database file. The workaround to clean up the out-of-design state, was to open and close the database with a read-write process (the recommended technique), or run MUPIP RUNDOWN as root (not recommended, as GT.M processes should be run as root only as a last resort).
- MUPIP RUNDOWN on a database file on which updates are frozen (e.g., using MUPIP FREEZE -ON) preserves the freeze. Previously if a frozen database file had a corresponding shared memory segment, MUPIP RUNDOWN would release the freeze.
- MUPIP RUNDOWN with no arguments detaches from shared memory segments as it moves from one Journal Pool to another. Previously, if the gtm\_custom\_errors environment variable of an instance was set to a non-null value and MUPIP RUNDOWN encountered a Journal Pool shared memory segment corresponding to the receiving side of a replication connection, it would report a MURPOOLRNDWNFL error on that Journal Pool but not detach from the shared memory segment before moving on to the next journal pool. Encountering a large number of such Journal Pools could potentially cause MUPIP RUNDOWN to exhaust available address space on a 32-bit architecture, and hit an address space limit, if configured, on a 64-bit architecture.

- MUPIP RUNDOWN works correctly on database files using the MM access method. Previously it incorrectly issued an "Invalid argument" message if it also issued a DBNAMEMISMATCH error on that MM database.
- A MUPIP SET -JOURNAL command that requires standalone access to the database file (for example turning replication on or off) works correctly. In GT.M versions V6.2-000 to V6.2-002A, it could in rare cases terminate abnormally with a segmentation fault (SIG-11).
- GT.M correctly maintains database shared memory in some rare cases when a database file has QDBRUNDOWN enabled and processes with read-only access open the database file BEFORE processes with read-write access. Previously, it was possible for an associated shared memory segment with updates not yet applied to the database file on disk to be inadvertently deleted.
- GT.M logs the LASTWRITERBYPAS message in the system log (syslog) file only once per database file for the time that it stays open. Previously, in rare cases it was possible for this message to be issued more than once.
- Helper processes (started with the GT.M replication Update Process) work correctly if they need to issue a LASTWRITERBYPAS message to the syslog. Previously they terminated abnormally with a segmentation fault (SIG-11).  
(GTM-8137) 🟢
- MUPIP LOAD accepts a broader range of labels; label second lines containing "ZWR", "GLO", or the pattern produced by MUPIP EXTRACT and %GO automatically determine a format. Starting with V6.2-001, LOAD required the second line be either the exact format produced by MUPIP EXTRACT and ^%GO or under some conditions have a second line ending in "; ZWR" or "; GLO". In addition, MUPIP LOAD accepts files with DOS style termination. For -FORMAT={ZWR|GO} files not produced by MUPIP EXTRACT or %GO, the first line of the label must contain the case insensitive text "UTF-8" for UTF-8 mode files and the second line should contain the case insensitive text "ZWR" for zwr format or "GLO" for GO format and the two label lines must contain a total of more than 10 characters. (GTM-8223) 🟢
- MUPIP replication compression supports the use of the IBM provided zlib library for AIX. Previously MUPIP replication compression required a custom compiled library. [AIX] (GTM-8225) 🟢
- MUPIP REPLIC -BUFFSIZE=<bytes> has a maximum of 4294967288 (4GiB-8) for both -SOURCE and -RECEIVER; previously it had a maximum of 2147483647 (2GiB-1). (GTM-8326)
- MUPIP TRIGGER -TRIGGERFILE presents a delete all confirmation prompt once, "Please enter Y or N:" (case insensitive). For any other response, MUPIP prompts again. Previously, every time MUPIP TRIGGER -TRIGGERFILE had to restart its transaction, it would repeat the prompt. Also, previously MUPIP TRIGGER -TRIGGERFILE would treat any response other than Y (case insensitive) as NO.  
(GTM-8342) 🟢
- The distribution contains a source tarball for the reference encryption plugin, but no binaries. If you wish to use the reference encryption plugin, you must follow the instructions to compile it from the source in the tarball. Previously the distribution included binaries. Although not backwards compatible, we took this step because variations in encryption libraries meant that we could not



provide a single binary that was guaranteed to run across Supported platforms with the robustness we require of GT.M. Note that when compiling the encryption plug-in, you should always review the makefile to ensure that all required dependencies in the makefile are installed on your system, and edit as needed to ensure that the locations of the header and library paths are correct for your system. (GTM-8361) ✓

- Repeated invocations of MUPIP JOURNAL ROLLBACK/RECOVER work correctly in case prior invocations terminated with certain errors. Previously, if at least two ROLLBACK or RECOVER commands terminated incompletely (say due to a GTM-E-MEMORY error) AND a MUPIP SET JOURNAL occurred between the interrupted commands, a subsequent ROLLBACK or RECOVER command could terminate abnormally with a GTMASSERT error even after correcting the cause of the original errors (say by raising the ulimit memory setting to avoid GTM-E-MEMORY errors). FIS recommends against unnecessarily changing journal files in the middle of operational processes such as ROLLBACK or RECOVER. (GTM-8394)
- The Source Server correctly identifies the current journal file in the presence of a concurrent journal file switch. Previously, on rare occasion (made somewhat less rare by specifying the -JNLFILEONLY option to the Source Server) the Source Server would fail to identify the current journal file, issue a NOPREVLINK error, and exit. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8410)
- See GTM-7658. (GTM-8420)
- A Receiver Server started with -autorollback remains active even if an online rollback does not change the state of the database. Previously a Receiver Server started with -autorollback would shutdown if an automatic online rollback did not change the state of the database. (GTM-8421)
- See GTM-6388. (GTM-8422)
- The Receiver Server (MUPIP) correctly handles replication record conversion for large transactions from prior GT.M releases. A regression in GT.M V6.2-000 caused the Receiver Server to hang when the conversion size of records from a single transaction replicated from V5.4-002B or earlier exceeded 2MiB. (GTM-8423)
- The Update Process receiving a replication stream works correctly in certain edge cases when a concurrent online rollback runs on the instance. Previously, the Update Process could terminate with a GTMASSERT error or hang and block further updates for potentially arbitrary periods of time. This was only encountered in the GT.M development environment, and was never reported by a user. (GTM-8425)
- MUPIP INTEG appropriately reports large values; previously large amounts could overflow and cause erroneous reports. (GTM-8428)
- Epoch tapering performs properly for a rare edge case. Previously, on very rare occasions, epoch tapering could encounter a divide-by-zero error. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8443)
- The Source Server updates the resync sequence number in the replication instance file every 60 seconds. Internal testing pointed to a few timing windows where the resync sequence number was not updated for over 90 seconds. (GTM-8461)

## System Administration

- MUPIP EXTRACT on an encrypted database instance creates valid binary extract files regardless of the region mapping. Previously, MUPIP EXTRACTs on instances with multiple regions pointing to the same database file could result in invalid extract files, that MUPIP LOAD could not process. The workaround was to use a global directory with regions merged so that each database file was referenced in just one segment. (GTM-8464)
- When a Source Server starts replicating updates from a newer generation journal file, it closes any prior generation of that journal file that it has open. Previously, it closed open journal files only when switching to the Journal Pool, which was problematic for a Source Server started with the -jnlfileonly option. The workaround was to shut down and restart the Source Server. Additionally, the Source Server more efficiently handles the case where it has a large number of journal files open - a common situation when its receiving secondary instance starts with a significant backlog, forcing the Source Server to chain through large numbers of prior generation journal files. (GTM-8468)
- See GTM-7831. (GTM-8470)
- When completing a TLS renegotiation, the source server places a "Sending REPL\_RENEG\_COMPLETE" in its log file after a "REPL\_RENEG\_ACK received" message and before a "Sent REPL\_RENEG\_COMPLETE" message; previously it did not record this event. (GTM-8478)
- MUPIP REORG limits its global buffer usage to the value specified by the gtm\_poollimit environment variable, or, by default, if gtm\_poollimit is not defined, to 64 buffers. Previously MUPIP REORG did not restrict its use of global buffers although the documentation stated that it did. We are aware of some issues with gtm\_poollimit and recommend the following until the next release: avoid using it for processes that make extended global references or run MUPIP TRIGGER. (GTM-8479)
- MUPIP BACKUP -DATABASE now attempts to preserve, and potentially restore, sparseness in database files. Previously, MUPIP BACKUP -DATABASE resulted in backup database files that were fully allocated. In the case of large but sparse database files, this could produce backup database files that used substantially more storage than the original database files. To remove sparseness from backup database files, use the fallocate command on Linux, or copy them to another location using the cp command on AIX. (GTM-8480) 🟢
- MUPIP INTEG -FAST handles certain obscure integrity conditions correctly. Previously, these conditions could result in a REGSSFAIL error (AIX) or KILLBYSIGSINFO1 (signal 11) fatal error and core dump (Linux). MUPIP INTEG without the -FAST qualifier was not affected. (GTM-8481)
- MUPIP JOURNAL correctly issues FILEPARSE errors for invalid paths; previously FILEPARSE errors would result in a SIG-11. (GTM-8485)
- Shutting down all Source Server processes when GT.M processes are still accessing a database file does not generate inappropriate REPLINSTMATCH errors. (GTM-8487)
- MUPIP BACKUP -ONLINE creates a usable backup of the replication instance file on a secondary instance. Previously, in rare cases, it could create a backup instance file that would cause the secondary to be out of sync with the currently replicating primary. Also, the replication instance file header on disk stores the current journal sequence number whenever a new history record gets added to the instance file. Previously this was not maintained which meant a MUPIP REPLIC -EDIT -SHOW command on the instance file potentially returned stale information on a live replicated

environment. These issues were only observed in the GT.M development environment, and never reported by a user. (GTM-8494)

- MUPIP commands that need standalone access, for example, MUPIP SET -REPLICATION=ON, issue a MUUSERLBK error on a crashed replication-enabled database, and MUUSERECOV error in case of a non-replicated-but-journaled database. Previously, the MUPIP commands would attempt a rundown regardless of the replication and journaling state of the database, potentially resulting in an unrecoverable database. Also MUPIP JOURNAL -RECOVER -FORWARD and MUPIP JOURNAL -ROLLBACK -FORWARD now issue a MUUSERECOV or MUUSERLBK error in case the database shared memory segment exists. In this case, only a MUPIP JOURNAL -RECOVER -BACKWARD or MUPIP JOURNAL -ROLLBACK -BACKWARD can correctly flush the updates from shared memory to the database file on disk. Previously MUPIP JOURNAL -RECOVER -FORWARD used to proceed with the recovery potentially creating a corrupt database file. Also, MUPIP RUNDOWN issues a MUUSERLBK error on a crashed replication-enabled database even if the database has NOBEFORE\_IMAGE journaling. Previously, it used to issue a MUUSERECOV error in this case. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8502) 🟢
- Replication from an originating primary instance A to business continuity instance B and a supplementary instance P to an instance downstream Q (i.e., B<-A->P->Q) is more robust. Previously, if there was at least one switchover between A and B while the P->Q link was operating (i.e., from B<-A->P->Q to A<-B->P->Q) followed by a disruption in the P->Q replication connection, the automatic reconnect incorrectly concluded that P and Q were out of sync and the receiver server on Q incorrectly terminated. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8507)



---

## Other

- The DSE ALL -CLEARCORRUPT qualifier sets the CORRUPT\_FILE file header to FALSE for all GDS regions. Use the ALL -CLEARCORRUPT qualifier only after receiving instructions from your GT.M support channel. Previously, there was no single command to clear the CORRUPT\_FILE to FALSE for all regions. (GTM-7199)
- ^%RI correctly responds to a yes answer to the "Formfeed delimited <No>?" question, correctly places its output even if the output is sent to a directory without specifying a trailing slash (/), appropriately restores the characteristics of \$PRINCIPAL that it adjusts, and accepts "DOS" line terminations (<CR><LF> instead of <LF>) at the end of input file lines. Previously it ignored a "YES" form-feed answer, prepended the directory name to the routines, changed the characteristics of the principal device and retained <CR>s at the end of input file lines, which resulted in compilation errors.(GTM-7658) 🟢
- Created processes inherit only those open files that they should inherit. Previously unintended file sharing with non-GT.M executables could cause rare file handling errors. Non-GT.M executables can be run via ZSYSTEM, PIPE, MUPIP replication filters, and the gtm\_procstuck\_exec environment variable. (GTM-8009)
- ^%GI accepts records up to the maximum string length (currently 1MiB). Previously it was limited to 8KiB for ZWR format and 2044 bytes for GO format. (GTM-8022) 🟢
- The GT.M encryption plugin works correctly when \$gtm\_dist points to the utf8 subdirectory of the GT.M installation, as it should for UTF-8 mode processes. Previously, pointing \$gtm\_dist to the utf8 subdirectory resulted in the encryption plugin failing to load with a CRYPTDLNOOPEN2 error. The workaround was to replace the symbolic link plugin in the utf8 directory with a copy of the plugin directory from the main GT.M installation directory. This was originally fixed in V6.2-000, but was inadvertently omitted from the release notes. (GTM-8028)
- In the event that a process detects a certain class of inconsistencies in the Journal Pool, it generates a core file (but does not terminate), and forces replication to continue from the journal files. This causes replication to reset to a known state. Replication resumes from the Journal Pool once a Source Server process determines that the updates it needs to replicate are in the Journal Pool. (GTM-8076)
- A GT.M process waits for 500 ms before re-attempting to start gtmsecshr. As a consequence of a regression introduced in a previous version, while processing a timed event, GT.M waited only 3 milliseconds between attempts which could generate unnecessary syslog messages. (GTM-8395)
- The %MPIECE utility NEWs all local variables except its arguments. Previously it could disrupt the state of a caller's local variables. (GTM-8403)
- Journal record time stamps are more closely aligned with \$HOROLOG and the timekeeping used for the HANG function. Previously there was some evidence that a global SET, HANG 1, global SET might occasionally yield two journal records in with the same time stamp. (GTM-8416)

## Other

- The build dependencies in the source code for GT.M released under a free / open source software license support a greater variety of environments. GT.M V6.2-002/-002A builds failed in some build environments with the error: "fatal error: xfer\_desc.i: No such file or directory". (GTM-8429)
- The gtminstall script now honors the --copyenv, --copyexec and --group-restriction options; previously these options caused the script to fail. (GTM-8441)
- DSE manages output so it appears in the intended order. In versions, from GT.M V6.2-000 to GT.M V6.2-002A inclusive, when stdout and stderr for DSE invocation were assigned to the same file or terminal, DSE sometimes presented output in an incorrect order. There was no issue if stderr and stdout were directed to different destinations - output always went to stderr. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8458)
- When starting gtmsecshr, GT.M clears environment variables that gtmsecshr does not need. Previously, it set them to the empty string (""), which could result in benign ARCTLMAXLOW warnings in the syslog. (GTM-8471)
- DSE interprets the keys on damaged blocks appropriately; previously moving between a good block and a damaged block could cause DSE to report the wrong key interpretation. This issue was discovered in the GT.M development environment and was never reported by any user. (GTM-8495)
- DSE uses the original values used to create shared memory in case other values have been written to the file header. Previously mismatched values between shared memory and the file header prevented DSE from attaching to an active database. Note that such a scenario would require explicit abuse or misuse of DSE, and could not happen accidentally. Please remember that:
  - As a low level database repair tool of last resort, DSE assumes a knowledgeable user, and does no edit checking of input values. Do not use DSE to make routine changes, and do not use DSE to change a parameter if you can accomplish the same goal with MUPIP. As the normal system administration and operations tool, MUPIP has the ability to change parameters you might normally need to change, and it does check that input values are reasonable.
  - Changing fileheader parameters with DSE should normally be performed with stand-alone access. Change fileheader parameters on an open database only under the guidance of an expert GT.M support channel.  
This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8511)
- The --xec command line option of the %XCMD utility program is optional. Previously, it was required. (GTM-8514)

---

## More Information

---

### Additional information for GTM-7291 - MUPIP JOURNAL -ROLLBACK qualifiers

Except as detailed below, qualifiers previously supported for MUPIP JOURNAL -ROLLBACK -BACKWARD are supported with MUPIP JOURNAL -ROLLBACK -FORWARD.

The -BEFORE time qualifier applies to MUPIP JOURNAL -ROLLBACK, both -FORWARD and -BACKWARD. As for MUPIP JOURNAL -RECOVER, the -BEFORE qualifier specifies the time at which ROLLBACK stops applying updates to the database in its forward processing phase (i.e., no journal records with update times after the -BEFORE time are applied to the database). If -BEFORE (time-based) and -FETCHRESYNC/-RESYNC (sequence-number-based) are specified in the same MUPIP JOURNAL -ROLLBACK command, the qualifier that corresponds to an earlier database state or point in time prevails i.e. if the update corresponding to the sequence number obtained through the -FETCHRESYNC command happened at a later time relative to the -BEFORE qualifier, -BEFORE prevails and vice versa.

The -CHAIN qualifier applies to MUPIP JOURNAL -ROLLBACK -FORWARD just as it does to MUPIP JOURNAL -RECOVER -FORWARD.

Unlike MUPIP JOURNAL -RECOVER -FORWARD, MUPIP JOURNAL -ROLLBACK -FORWARD, accepts only -CHECKTN, which is the default, but does not accept -NOCHECKTN.

-FENCES=NONE and FENCES=ALWAYS are not permitted for MUPIP JOURNAL -ROLLBACK (with -BACKWARD or -FORWARD); ROLLBACK supports -FENCES=PROCESS (the default option). Previously MUPIP JOURNAL -ROLLBACK -BACKWARD allowed -FENCES=NONE or -FENCES=ALWAYS which could cause incomplete transactions to be played as if they were complete and result in a database file potentially out-of-sync with its journal files.

The -SINCE time qualifier applies to MUPIP JOURNAL -ROLLBACK -BACKWARD. As in MUPIP JOURNAL -RECOVER, the -SINCE qualifier specifies how far back in time MUPIP JOURNAL -ROLLBACK -BACKWARD should at least process (from the end of the journal file), before starting the forward processing. The actual turn-around point for -RECOVER and -ROLLBACK in each database region is an epoch in the journal files before or at the -SINCE time, but not after it.

-NOVERIFY is the default for MUPIP JOURNAL -RECOVER -FORWARD as well as MUPIP JOURNAL -ROLLBACK -FORWARD, with the exception of MUPIP JOURNAL -RECOVER -FORWARD -NOCHECKTN for which -VERIFY remains the default. Previously, -VERIFY was the default for MUPIP JOURNAL -RECOVER -FORWARD. -VERIFY remains the default for all other MUPIP JOURNAL commands (including MUPIP JOURNAL -RECOVER -BACKWARD and MUPIP JOURNAL -ROLLBACK -BACKWARD).

The -FETCHRESYNC, -ONLINE, and -RSYNC\_STRM qualifiers are not supported for MUPIP JOURNAL -ROLLBACK -FORWARD.

---

## Additional Information for GTM-8296 - %PEEKBYNAME()

The format of the %PEEKBYNAME() function is %PEEKBYNAME(field[,regindex][,format]) where field specifies the type of information to be returned, in the format: control\_block[.field].\* Some control\_blocks are:

- `gd_region` - fields from the global directory typically accessed via GDE using a SHOW command; remember that these values are only used when MUPIP CREATE creates new database files.
- `gtmsource_local_struct` - fields from the replication instance file, and typically accessed using MUPIP REPLICATE.
- `jnl_buffer` - fields from journaling control structures typically accessed using DSE DUMP FILEHEADER
- `jnlpool_ctl_struct` - journal Pool fields typically accessed using MUPIP REPLICATE.
- `node_local` - fields from database shared memory that are not part of the fileheader, typically accessed using DSE DUMP FILEHEADER.
- `recvpool_ctl_struct` - receive Pool fields (on an instance receiving a replication stream) typically accessed using MUPIP REPLICATE.
- `repl_inst_hdr.inst_info` - replication fields that change infrequently, if ever.
- `sgmnt_data` - fields from database shared memory also part of the database fileheader, typically accessed using DSE DUMP FILEHEADER.

The optional second expression specifies a region name, structure index or a base address associated with the first (field name) argument. The choice is governed by the following rules applied in the following order:

- If the value is a hexadecimal number in the form of `0xhhhhhhh[hhhhhhh]`, then PEEKBYNAME uses it as the base address of the data to fetch. Also in this case, the offset, length, and type are taken from the field specified in the first expression (field). For more information, see the description of the "PEEK" mnemonic in \$ZPEEK(). FIS recommends that you not use a hexadecimal number except under the direction of your GT.M support channel.
- If the first expression refers to one of the region-related structures supported by the \$ZPEEK() function, PEEKBYNAME treats this second expression as a region name.
- If the first expression refers to one of the replication related structures supported by the \$ZPEEK() function that are indexed, PEEKBYNAME treats this second expression as an integer index value.
- For those structures supported by the \$ZPEEK() function that do not accept an argument, this second expression must be NULL or left unspecified.

The optional third expression specifies the output format in one character as defined in the "format" argument in the \$ZPEEK() documentation. This argument overrides the automatic format detection



by the %PEEKBYNAME utility. FIS recommends that you not use the third argument except under the direction of your GT.M support channel.

## Examples:

```
GTM>write $$^%PEEKBYNAME("gd_region.max_key_size","DEFAULT") ; Max key size for region
DEFAULT
255
GTM>
```

## LISTALL ^%PEEKBYNAME

Prints all the field mnemonics acceptable as the first argument to %PEEKBYNAME() on the current output device.

## LIST ^%PEEKBYNAME(.output)

Populates output variable with the type and size information indexed by the field mnemonics for all fields accepted by %PEEKBYNAME(). FIS recommends that you not use the results of this entryref except under the direction of your GT.M support channel.

## Labels for Selected Fields

Below are selected fields for which you may find %PEEKBYNAME to be a useful alternative to running a DSE or MUIP command in a PIPE device, and parsing the output. If there is a field that you wish to access using %PEEKBYNAME, please send questions to your GT.M support channel. We will get you an answer, and if it seems to us to be of general interest, we will add it to the %PEEKBYNAME user documentation.

## Region Parameters

Calls to %PEEKBYNAME with the listed string as value of the first parameter, and the region name as the value of the second parameter, return the value. For example:

```
GTM>write $$^%PEEKBYNAME("sgmnt_data.n_bts","DEFAULT") ; How many global buffers there are
1000
GTM>write $$^%PEEKBYNAME("node_local.wcs_active_lvl","DEFAULT") ; How many of them are dirty
0
GTM>for i=1:1:10000 set ^x($$^%RANDSTR(8))=$$^%RANDSTR(64)

GTM>write $$^%PEEKBYNAME("node_local.wcs_active_lvl","DEFAULT") ; And now, how many of them
are dirty
377
GTM>
```

When using the following, remember to write code that allows for values other than those listed, e.g., if writing code to check whether before image journaling is in use, make sure it can deal with values

other than 0 and 1, because a future release of GT.M can potentially introduce a new return value for a field.

Parameter	^%PEEKBYNAME() Parameter	Value
Block size	"sgmnt_data.blk_size"	Integer number of bytes
Commit wait spin count	"sgmnt_data.wcs_phase2_commit_wait_spincnt"	Integer count
Current transaction	"sgmnt_data.trans_hist.curr_tn"	Integer count
Defer allocate	"sgmnt_data.defer_allocate"	Integer - 1 means DEFER_ALLOCATE, 0 means NODEFER_ALLOCATE
Encryption current key hash	"sgmnt_data.encryption_hash"	String of binary bytes
Encryption - IVs in use	"sgmnt_data.non_null_iv"	Integer - 1 means unencrypted or encrypted with IVs, 0 means encrypted with zero IVs
Encryption new key hash (while MUPIP REORG -ENCRYPT is underway)	"sgmnt_data.encryption_hash2"	String of binary bytes
Extension size	"sgmnt_data.extension_size"	Integer number of blocks
Epoch taper set	"sgmnt_data.epoch_taper"	Integer - 1 means epoch taper is set, 0 means it is not
Flush trigger	"sgmnt_data.flush_trigger"	Integer number of blocks (not meaningful for MM)
Journal align size	"sgmnt_data.alignsize"	Integer number of bytes
Journal autoswitch limit	"sgmnt_data.autoswitchlimit"	Integer number of bytes for maximum size of each journal file
Journal before imaging	"sgmnt_data.jnl_before_image"	Integer - 1 means BEFORE image journaling, 0 means NOBEFORE (meaningful only if journaling is on)
Journal buffer size	"sgmnt_data.jnl_buffer_size"	Integer number of journal buffers

Parameter	^%PEEKBYNAME() Parameter	Value
Journal epoch interval	"sgmnt_data.epoch_interval"	Integer number of seconds
Journal next write offset	"jnl_buffer.dskaddr"	Integer number of bytes from beginning of journal file
Journal next epoch time	"jnl_buffer.next_epoch_time"	Integer number of seconds since January1, 1970 00:00:00 UTC
Journal state	"sgmnt_data.jnl_state"	Integer 0 means disabled, 1 means enabled but off, 2 means on
Journal SYNCIO	"sgmnt_data.jnl_sync_io"	Integer - 1 means SYNC_IO, 0 means NOSYNC_IO
Journal yield limit	"sgmnt_data.yield_lmt"	Integer count
Lock space	"sgmnt_data.lock_space_size"	Integer number of bytes
Maximum key size	"sgmnt_data.max_key_size"	Integer number of bytes
Machine name	"node_local.machine_name"	String
Maximum record size	"sgmnt_data.max_rec_size"	Integer number of bytes
Mutex hard spin count	"sgmnt_data.mutex_spin_parms.mutex_hard_spin_count"	Integer count
Mutex sleep spin count	"sgmnt_data.mutex_spin_parms.mutex_sleep_spin_count"	Integer count
Number of global buffers (dirty)	"node_local.wcs_active_lvl"	Integer count
Number of global buffers (total)	"sgmnt_data.n_bts"	Integer count
Number of processes accessing the database	"node_local.ref_cnt"	Integer count (always greater than zero, because the process running %PEEKBYNAME has the database open)
QDBRUNDOWN setting	"sgmnt_data.mumps_can_bypass"	Integer - 1 means QDBRUNDOWN set, 0

Parameter	^%PEEKBYNAME() Parameter	Value
		means QDBRUNDOWN not set
Region replication sequence number	"sgmnt_data.reg_seqno"	Integer count
Spanning nodes absent	"sgmnt_data.span_node_absent"	Integer - 1 means that no global variable nodes span multiple database blocks, 0 means GT.M does not know (in the past, at least one global variable node spanned multiple blocks, but it may since have been overwritten or KILL'd)

## Replication Parameters

Calls to %PEEKBYNAME with the listed parameter as the first or only parameter return replication fields as described. For example:

```
GTM>write $$^%PEEKBYNAME("repl_inst_hdr.inst_info.this_instname") ; Name of this instance
Collegeville
GTM>write $$^%PEEKBYNAME("gtmsource_local_struct.secondary_instname",0) ; Name of instance in
slot 0 of replication instance file
Malvern
GTM>set x=$$^%PEEKBYNAME("jnlpool_ctl_struct.jnl_seqno") ; Sequence number in Journal Pool of
Collegeville

GTM>set y=$$^%PEEKBYNAME("gtmsource_local_struct.read_jnl_seqno",0) ; Next sequence number to
send to Malvern
GTM>write x-y ; Current replication backlog from Collegeville to Malvern
2
GTM>
```

Replication Parameter	^%PEEKBYNAME() Parameter	Value
Journal sequence number	"jnlpool_ctl_struct.jnl_seqno"	Integer
Journal sequence number to send to receiving instance in replication file slot	"gtmsource_local_struct.read_jnl_seqno", <i>i</i> where <i>i</i> is the slot number in the replication instance file	Integer

Replication Parameter	^%PEEKBYNAME() Parameter	Value
Name of receiving instance in replication instance file slot	"gtmsource_local_struct.secondary_instname", <i>i</i> where <i>i</i> is the slot number in the replication instance file	String of text
Name of this instance	"repl_inst_hdr.inst_info.this_instname"	String of text
QDBRUNDOWN setting	"repl_inst_hdr.qdbrundown"	Integer - 1 means QDBRUNDOWN set, 0 means QDBRUNDOWN not set
Supplementary Replication	"repl_inst_hdr.is_supplementary"	Integer - 1 means supplementary instance; 0 means not supplementary instance
Updates disabled	"jnlpool_ctl_struct.upd_disabled"	Integer - 1 means updates disabled; 0 means updates permitted



---

## Error and Other Messages

---

### **CRYPTJNLMISMATCH**

**CRYPTJNLMISMATCH**, Encryption settings mismatch between journal file jjjj and corresponding database file dddd

All GT.M Components Error: Encryption settings in the header of database file dddd do not match those stored in the header of journal file jjjj.\nThis is most likely caused by inappropriate operator action such as replacing the current journal file with an older journal file.\n

Action: Correct the error that caused the incorrect journal file to be pointed to by the database file. If the correct journal file has been inadvertently deleted, create new journal files with the -noprevjnl switch. Take a backup as soon as possible thereafter. Depending on your situation, you may need to refresh secondary instances.

---

### **CRYPTKEYRELEASEFAILED**

**CRYPTKEYRELEASEFAILED**, Could not safely release encryption key corresponding to file ffff. eeee

All GT.M Components Error: gtmcrypt plug-in reports that it is unable to release the memory pertaining to the encryption key associated with file ffff due to error eeee

Action: Examine message eeee from the plug-in and take the needed action: for example, ensure that the memory is accessible, process has correct permissions, and so on.

---

### **CRYPTNOKEY**

**CRYPTNOKEY**, No encryption key specified

MUPIP Error: MUPIP REORG -ENCRYPT prints this message if no encryption key is specified.

Action: Provide the requisite encryption key to the command as instructed in GT.M documentation.

---

### **ENCRYPTCONFLT**

**ENCRYPTCONFLT**, MUPIP REORG -ENCRYPT and MUPIP EXTRACT -FORMAT=BIN cannot run concurrently - skipping oooo on region: rrrr, file: ffff

MUPIP Error: MUPIP cannot perform REORG -ENCRYPT and EXTRACT -FORMAT=BIN on file ffff at the same time; rrrr is the region that mapped the file; oooo is the just started operation.

Action: Reschedule the just started operation or terminate the conflicting operation to allow the just started operation to run immediately.

---

## EXTRINTEGRITY

**EXTRINTEGRITY**, Database ffff potentially contains spanning nodes or data encrypted with two different keys

MUPIP Error: MUPIP EXTRACT cannot run because the database file ffff might contain spanning nodes or be partially encrypted with a particular key. Proceeding on a live database in such situation could result in data corruption.

Action: Reformat the data to contain no spanning nodes, let MUPIP REORG -ENCRYPT complete (re)encryption of the database, or reissue the MUPIP EXTRACT command with a -FREEZE qualifier to acquire stand-alone access for the duration of the procedure. As a final resort, use an -OVERRIDE qualifier to proceed on a live database that either contains spanning nodes or is undergoing (re)encryption. FIS highly discourages using the -OVERRIDE qualifier unless the database is quiescent.

---

## GDINVALID

**GDINVALID**, Unrecognized Global Directory file format: ffff, expected label: eeee, found: bbbb

Run Time Error: This indicates that a version of the global directory file xxx does not match with the version expected by GT.M. The file might have been created by an incompatible GT.M version. If the text of eeee or bbbb contain non-graphic characters, GT.M replaces each of them with a period (.).

Action: Compare the labels eeee and bbbb. If the global directory was created by an earlier GT.M version, upgrade the file by loading and then saving the file using the GDE of the new GT.M version.

---

## INVLINKTMPDIR

**INVLINKTMPDIR**, Value for \$gtm\_linktmpdir is either not found or not a directory: dddd

Run Time Error: Indicates the process cannot access directory dddd, which it needs in order to do auto-relink as specified by its \$ZROUTINES; the directory may not exist as a directory or the process lacks authorization to the directory.

Action: The directory specification comes from \$gtm\_linktmpdir if it is defined, otherwise from \$gtm\_tmp if that is defined; otherwise it defaults to the system temporary directory, typically /tmp. Either correct the environment variable definition or ensure directory dddd is appropriately set up. Note that all users of auto-relink for a directory normally need to use the same temporary directory for their relink control files.

---

## INVLOCALE

**INVLOCALE**, Attempt to reset locale to supplied value of \$gtm\_locale xxxx failed

All GT.M Components Error: GT.M found the value of \$gtm\_locale xxxx did not specify a valid currently supported local

Action: Correct the locale setup and restart the process.



---

**INVZWRITECHAR** 

**INVZWRITECHAR**, Invalid characters for a ZWRITE format

Run Time/Compile Time Error: When transforming an expression from ZWRITE format to full text format with \$ZWRITE(expr,1), the expression must be in a format that GT.M would have produced when transforming a text string to ZWRITE format.

Action: Examine the expression and ensure that it is in proper ZWRITE format

---

**IOEOF** 

**IOEOF**, Attempt to read past an end-of-file

Run Time/MUPIP Error: This indicates that a READ command for a run-time system or a MUPIP command attempted to move past an end-of-file.

Action: Verify that the \$ZEOF special variable is tested by the function between READs or that an EXCEPTION code string is assigned to handle EOFs. Alternatively, have your \$ETRAP (or \$ZTRAP) error handling deal with this error. The USE command has a REWIND deviceparameter that allows you to read from the beginning of the file without having to CLOSE and OPEN again, which may facilitate recovery from this error. Attempting to READ from a non-existent file not opened READONLY also causes this error. In the event of a MUPIP error, make sure the file being read is not corrupted.

---

**JNLDBSEQNOMATCH** 

**JNLDBSEQNOMATCH**, Journal file ffff has beginning region sequence number jjjj but database dddd has region sequence number ssss

MUPIP Error: MUPIP JOURNAL ROLLBACK FORWARD has found that journal file ffff has beginning region sequence number jjjj, but the corresponding database file dddd has region sequence number ssss. This condition may arise due to missing or incorrect journal files, for example due to a -NOCHAIN specification.

Action: Use "\*" and / or do not use -NOCHAIN to specify the list of journal files. If specifying explicit list of journal file names verify you are specifying the exact set of needed journal file names.

---

**JNLPOOLRECOVERY** 

**JNLPOOLRECOVERY**, The size of the data written to the journal pool (xxxx) does not match the size of the data in the journal record (yyyy) for the replication instance file zzzz. The journal pool has been recovered.

Run Time Error: An internal error was detected while writing to the journal pool associated with instance file zzzz, and the journal file has been recovered. Subsequent transactions will be written to the journal pool, but the source server will switch to reading from files until it reaches them. A core file may have been produced.

Action: Report the entire incident context to your GT.M support channel.

---

## JOBLVN2LONG

**JOBLVN2LONG**, The zwrite representation of a local variable transferred to a JOB'd process is too long. The zwrite representation cannot exceed MMMM. Encountered size: LLLL

Run Time Error: This error indicates that the total length LLLL (in bytes) of the ZWRITE representation of the variable name, subscripts, and value exceeds the maximum MMMM supported by the PASSCURLVN facility. Note that the ZWRITE representation contains the appropriate punctuation for any subscripts, the equal-sign and replaces any non-graphic characters with their \${Z}CHAR() representations.

Action: Consider whether the JOB'd process needs the variable(s) that exceed the maximum for PASSCURLVN - if not they can be taken out of scope before the JOB command. Alternatively, pass them using global variables or a local SOCKET device.

---

## JOBLVNDETAIL

Last used version: V6.2-003

**JOBLVNDETAIL**, The zwrite representation of a local variable transferred to a JOB'd process is too long. The zwrite representation cannot exceed XXXX. Encountered size: YYYY

Run Time Error: The length of the zwrite representation of a local, (including the quotes, the '=', concatenate operator "\_", and "\${Z}C()") has the length of YYYY which exceeds the maximum limit of XXXX.

Action: Please check the sizes of locals that needs to be sent and make sure their lengths are less than XXXX. For those big locals, consider using another mechanism such as sockets.

---

## MULTIPROCLATCH

**MULTIPROCLATCH**, Failed to get multi-process latch at xxxx

MUPIP Error: A process was unable to acquire a multi-process latch (the resource that ensures correctness of execution amongst multiple processes) in a timely manner; xxxx is the address of the failing request.

Action: Report the entire incident context to your GT.M support channel.

---

## MUPIPSET2BIG

**MUPIPSET2BIG**, vvvv too large, maximum tttt allowed is mmmm

MUPIP Error: The value vvvv for tttt specified in a MUPIP SET command is above the maximum mmmm for tttt

Action: Decrease the specified value to not exceed the maximum.

---

## MUPIPSET2SML

**MUPIPSET2SML**, vvvv too small, minimum tttt allowed is mmmm

MUPIP Error: The value vvvv for tttt specified in a MUPIP SET command is below the minimum mmmm for tttt

Action: Increase the specified value to meet or exceed the minimum.

---

## MUPJNLINTERRUPT

**MUPJNLINTERRUPT**, Database file xxxx indicates interrupted MUPIP JOURNAL command. Restore from backup for forward recover/rollback.

MUPIP Error: This indicates that a MUPIP JOURNAL -ROLLBACK -FORWARD or a MUPIP JOURNAL -RECOVER -FORWARD did not proceed because a previous MUPIP JOURNAL command attempted on the database was terminated abnormally.

Action: Restore the database and journal files from a backup to proceed with the MUPIP JOURNAL -ROLLBACK -FORWARD or MUPIP JOURNAL -RECOVER -FORWARD.

---

## MURENCRYPTEND

**MURENCRYPTEND**, Database ffff : MUPIP REORG ENCRYPT finished by pid pppp at transaction number 0xtttt

MUPIP Information: The MUPIP REORG -ENCRYPT initiated by process pppp completed an encryption change for database file ffff at transaction number 0xtttt

Action: None required.

---

## MURENCRYPTSTART

**MURENCRYPTSTART**, Database ffff : MUPIP REORG ENCRYPT started by pid pppp at transaction number 0xtttt

MUPIP Information: Process pppp used MUPIP REORG -ENCRYPT to start or restart an encryption change at transaction number 0xtttt for database file ffff

Action: None required.

---

## MURENCRYPTV4NOALLOW

**MURENCRYPTV4NOALLOW**, Database (re)encryption supported only on fully upgraded V5 databases. ffff has V4 format blocks

MUPIP Error: MUPIP cannot enable or perform encryption on database file ffff while it contains GDS V4 format blocks.

Action: Upgrade the database to V5 and re-run the action.

---

## NLRESTORE

NLRESTORE, DB file header field FFFF: VVVV does not match the value used in original mapping - restoring to: OOOO

DSE Warning: When DSE encounters a internal header field named FFFF whose value VVVV conflicts with the original value OOOO, DSE issues a warning message and uses the original value in order to successfully access shared memory.

Action: Please restore the header fields to their correct values. As a low level database repair tool of last resort, DSE assumes a knowledgeable user, and does no edit checking of input values. Do not use DSE to make routine changes, and do not use DSE to change a parameter if you can accomplish the same goal with MUPIP. As the normal system administration and operations tool, MUPIP has the ability to change parameters you might normally need to change, and it does check that input values are reasonable. Changing fileheader parameters with DSE should normally be performed with stand-alone access. Change fileheader parameters on an open database only under the guidance of an expert GT.M support channel.

---

## NOMORESEMCNT

NOMORESEMCNT, SSSS counter semaphore has reached its maximum and stopped counting for database DDDD. Run MUPIP JOURNAL -ROLLBACK -BACKWARD, MUPIP JOURNAL -RECOVER -BACKWARD or MUPIP RUNDOWN to restore the database files and shared resources to a clean state

All GT.M Components Information: The counter semaphore reached its system-imposed limit so GT.M no longer maintains the count. SSSS is either "access" or "ftok" signifying the particular counter type that stopped. DDDD is the database of the corresponding counter.

Action: GT.M will not automatically shutdown the database. To clean the database file header and shared resources after the last process has exited the database file, do an explicit MUPIP -ROLLBACK -BACKWARD (for replicated database files), MUPIP JOURNAL -RECOVER -BACKWARD (for database files that are journaled but not replicated), or MUPIP RUNDOWN (for database files that are neither replicated nor journaled), on the database file DDDD.

---

## NONTPRESTART

NONTPRESTART, Database dddd; code: cccc; blk: bbbb in glbl: ^gggg; blklvl: llll, type: tttt, zpos: PPPP

Run Time Information: This is an informational message for non-TP transaction messages. The frequency of this message can be set by \$gtm\_nontprestart\_log\_delta and \$gtm\_nontprestart\_log\_first

environment variables. dddd is the database the restart occurred; cccc is the code described in the Maintaining Database Integrity chapter of the Administration and Operations Guide; bbbb is the block where GT.M detected a concurrency conflict that caused the restart; gggg shows the global reference within that block; llll is the level of that block; tttt indicates the type of activity that detected the conflict; pppp is the source line where restart occurred on.

Action: None required in most cases. If the messages are too frequent either investigate the processes that reference to that particular global and its block, or reduce the number of messages by tweaking \$gtm\_nontprestart\_log\_delta and \$gtm\_nontprestart\_log\_first environment variables.

---

## NOPRINCIO

**NOPRINCIO**, NOPRINCIO Unable to write to principal device

Run Time Fatal: This indicates that GT.M attempted to but could not read from or write to the principal device.

Action: The NOPRINCIO error message works differently from other messages. The first occurrence results in an error that can be caught by device and trap handlers. The second occurrence is FATAL which does not drive device or trap handlers and terminates the process. This termination does not allow any application level orderly shutdown and, depending on the application may lead to out-of-design application state. Therefore FIS recommends appropriate application level error handling that recognizes this error and performs an orderly shutdown without performing any additional READ or WRITE to the principal device. The most common causes for the principal device to cease to exist involve terminal sessions or socket connections (including those from processes started by inetd/xinetd). When the remote client terminates the connection, the underlying principal device is closed and becomes inaccessible when the process attempts to READ from, or WRITE to, it. In the case of terminals, a typical cause is users closing the window of a terminal session without cleanly exiting from the GT.M process.

---

## NOTALLJNLEN

**NOTALLJNLEN**, Journaling disabled/off for dddd regions

MUPIP Warning: This indicates that some or all regions do not have journal state ON.

Action: Ensure you have journaling enabled for all regions that require it; use MUPIP SET to enable journaling.

---

## NOTALLREPLON

**NOTALLREPLON**, Replication off for dddd regions

MUPIP Warning: This indicates that some or all regions have replication state OFF.

Action: Ensure you have replication on for all regions that require it; use MUPIP SET to enable replication.

---

**PBNINVALID** 

**PBNINVALID**, ssss does not have a field named ffff

Utility Error: This message comes from %PEEKBYNAME() when a valid struct but an invalid field name is given as the first argument. A struct, ssss, does not have a field named ffff.

Action: Check the field name. Verify the field exists and its specification has no typo.

---

**PBNNOFIELD** 

**PBNNOFIELD**, %ZPEEKBYNAME() requires a field.item as its first parameter

Utility Error: The first argument of %ZPEEKBYNAME() may be missing, empty, contain an unsupported field or be missing an item.

Action: Verify the first parameter to %ZPEEKBYNAME() is not NULL.

---

**PBNNOPARM** 

**PBNNOPARM**, First parameter pppp does not support a second parameter

Utility Error: pppp does not take a region name or index number as the second parameter to %PEEKBYNAME().

Action: Omit the second parameter of %PEEKBYNAME() or make it NULL.

---

**PBNPARAMREQ** 

**PBNPARAMREQ**, A first parameter value pppp requires a second parameter specified containing rrrr

Utility Error: pppp requires a second parameter but the second parameter of %PEEKBYNAME() is NULL or undefined. rrrr indicates whether the required parameter is an index number or region name.

Action: Depending on rrrr, choose a valid index number or region name and make sure the second parameter is not NULL.

---

**PBNUNSUPSTRUCT** 

**PBNUNSUPSTRUCT**, \$ZPEEK() does not support structure ssss

Utility Error: The first argument of %PEEKBYNAME() is a value that is not known to \$ZPEEK().

Action: Make sure the first argument of %PEEKBYNAME() is a valid struct name that is accessible to \$ZPEEK.

---

**RECLOAD** 

**RECLOAD**, Error loading record number: nnnn

MUPIP Error: This message identifies a record that MUPIP could not LOAD and follows a message about the cause. If this message is Fatal, which it can be for BIN format, it produces a core file for diagnostic analysis.

Action: Address the cause or, for GO and ZWR format input files, examine the record with a text editor for possible correction or alternate action and for BIN format if fixing the cause does not resolve the error switch to ZWR format EXTRACT.

---

**REPLLOGOPN** 

**REPLLOGOPN**, Replication subsystem could not open log file xxxx : yyyy. Logging done to zzzz

MUPIP Error: This indicates that MUPIP could not find the log file or did not have access permission to open the log file. If there is another log file available (a previously opened file), MUPIP writes to the other log file. If there is no other log file available, MUPIP sends any remaining messages to /dev/null and terminates the replication server process.

Action: Check the log file permissions, and if permissions are correct, move the log file and specify that MUPIP should log to a log file which has appropriate access permissions.

---

**REPLSTATEOFF** 

**REPLSTATEOFF**, MUPIP JOURNAL -ROLLBACK -BACKWARD cannot proceed as database xxxx does not have replication ON

MUPIP Error: This indicates that a MUPIP JOURNAL -ROLLBACK -BACKWARD command cannot proceed because the specified database xxxx does not have replication state ON. In most situations, this error occurs when the journal file storage runs out of disk space.

Action: Ensure replication is turned ON for a database, before executing the MUPIP JOURNAL -ROLLBACK -BACKWARD command. If the database is in the WAS\_ON state, refer to the "Recovering from the WAS\_ON state" section in the Database Replication chapter of the Administration and Operations Guide. Alternatively, if replication was not in use on the database, use MUPIP JOURNAL -RECOVER.

---

**RESRCINTRLCKBYPAS** 

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

Run Time Information: GT.M issues the RESRCINTRLCKBYPAS message to the system log as an indication it may not detect when the last process detaches from the shared resource and therefore

may not rundown the database shared resources as it normally would. GT.M protects the actions of setting up and tearing down the shared resources associated with a database with a pair of semaphores. Because DSE, and LKE are tools for diagnosing issues, when they start and find they cannot acquire the semaphores after a reasonable number of tries, they proceed to open the database anyway because it is highly probable the database is already set up. When DSE and LKE bypass the semaphore acquisition, they leave the count of attached processes incorrect. When many processes terminate at the same time, typically because of a system shutdown, there can be significant contention for the semaphores that can cause their terminations to take an unusually long time. When this happens, and the count of remaining attached processes is significant, a process may skip the semaphore acquisition, again leaving the count of attached process incorrect. If either of these events occurs, GT.M issues the RESRCINTRLCKBYPAS message where tttt identifies the process type: "LKE", "DSE" or "GT.M"; qqqq is the bypassing process's PID; ssss identifies the semaphore type: "FTOK" or "access control"; rrrr is the region bypassed; ffff is the file corresponding to region rrrr; pppp is the PID of the process holding the semaphore.

Action: These messages when shutting down GT.M activity may indicate a need to complete the process by invoking a MUPIP JOURNAL -ROLLBACK -BACKWARD for replicated databases, a MUPIP JOURNAL -RECOVER -BACKWARD for unreplicated journaled databases and a MUPIP RUNDOWN for journal-free databases to get the database to a safe state; doing so as part of every shutdown is good practice.

---

## SETQUALPROB

**SETQUALPROB**, Error getting qqqq qualifier value

MUPIP Error: The utility was unable to parse the command input to successfully determine the value supplied for the qqqq qualifier

Action: Examine the command and correct the value

---

## TPRESTART

**TPRESTART**, Database mmmm; code: xxxx; blk: yyyy in glbl: zzzz; pvtmods: aaaa, blkmods: bbbb, blklvl: cccc, type: dddd, readset: eeee, writeset: ffff, local\_tn: gggg, zpos: hhhh

Run Time Information: The UNIX environment variables or OpenVMS logical names GTM\_TPRESTART\_LOG\_FIRST and GTM\_TPRESTART\_LOG\_DELTA control the logging of TPRESTART messages. GTM\_TPRESTART\_LOG\_FIRST indicates the number of TP restarts to log from GT.M invocation. Once that many have been logged, every GTM\_TPRESTART\_LOG\_DELTA TP restarts, GT.M logs a restart message. If GTM\_TPRESTART\_LOG\_DELTA is undefined, GT.M performs no operator logging. The default value for GTM\_TPRESTART\_LOG\_FIRST is 0 (zero), which leaves the control completely with GTM\_TPRESTART\_LOG\_DELTA. The facility that produces this message can serve as a diagnostic tool in developmental environments for investigating contention due to global updates. A zzzz of "\*\*BITMAP" indicates contention in block allocation which might involve multiple globals. hhhh is the \$ZPOSITION of the line of M code that caused the restart of the transaction; utilities leave this field blank.



Action: Disable, or adjust the frequency of, these messages with the mechanism described above. To reduce the number of restarts, consider changes to the global structure, varying the time when work is scheduled. Consider whether the business and program logic permits the use of NOISOLATION.

---

**TRIGINVCHSET** 

**TRIGINVCHSET**, Trigger tttt for global gggg was created with CHSET=cccc which is different from the current \$ZCHSET of this process

Trigger/Run Time Error: TRIGINVCHSET occurs when a process invokes a trigger on a global using a \$ZCHSET that is different from the \$ZCHSET used at the time of loading the first trigger on that global. GT.M implicitly uses the \$ZCHSET of the first trigger on a global to invoke all triggers on that global. Note that tttt is the name of the first trigger on the global gggg-not necessarily the name of the trigger being invoked. cccc is the \$ZCHSET of the process at the time of loading tttt on global gggg.

Action: Ensure that the process invoking a trigger on a global uses the same \$ZCHSET that was used to load the first trigger on that global. If your application requires triggers in both M and UTF-8 modes, use different globals to load M mode and UTF-8 mode triggers.

