

# GT.M

## Release Notes

V7.0-001

Empowering  
the Financial World

FIS

## Contact Information

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

GT.M Support for customers: [gtmsupport@fisglobal.com](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 ©2021 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 1.0	22 November 2021	V7.0-001

# Table of Contents

V7.0-001 .....	1
Overview .....	1
Conventions .....	1
Platforms .....	2
Platform support lifecycle .....	4
32- vs. 64-bit platforms .....	4
Call-ins and External Calls .....	4
Internationalization (Collation) .....	5
Environment Translation .....	5
Additional Installation Instructions .....	5
Recompile .....	6
Rebuild Shared Libraries or Images .....	6
Compiling the Reference Implementation Plugin .....	6
Upgrading to GT.M V7.0-001 .....	7
Managing M mode and UTF-8 mode .....	8
Setting the environment variable TERM .....	8
Installing Compression Libraries .....	9
Change History .....	10
V7.0-001 .....	10
Database .....	12
Language .....	13
System Administration .....	14
Other .....	16
Additional Information .....	17
Additional Information about GTM-8843 .....	17
Error and Other Messages .....	19
BACKUPDBFILE  .....	19
BACKUPFAIL  .....	19
BACKUPREPL  .....	19
BACKUPSEQNO  .....	19
BACKUPSUCCESS  .....	19
BACKUPTN  .....	19
BKUPFILEPERM  .....	20
BKUPPROGRESS  .....	20
BKUPRETRY  .....	20
CMDERR  .....	20
CRYPTNOV4  .....	20
DBMISALIGN  .....	20
DIRACCESS  .....	21
LASTTRANS  .....	21
NORTN  .....	21
REPLAHEAD .....	21
REPLCOMM  .....	21
RESTORESUCCESS  .....	22
SOCKBLOCKERR  .....	22
SOCKWAITARG  .....	22
SRCBACKLOGSTATUS  .....	22


# V7.0-001

## Overview

V7.0-001 supports database files with up to 11 levels (10 index and one data). V7.0-001 includes support for operation with V6 database file format used by recent GT.M versions.

V7.0-001 also includes other fixes and enhancements including changes aimed at making management of backups and replication more straightforward. For more information, refer to the Change History section.

Items marked with  document new or different capabilities.

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



### Note

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

## Conventions

This document uses the following conventions:

<b>Flag/Qualifiers</b>	-
<b>Program Names or Functions</b>	upper case. For example, MUPIP BACKUP
<b>Examples</b>	lower case. For example: mupip backup -database ACN,HIST /backup
<b>Reference Number</b>	A reference number is used to track software enhancements and support requests. 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.	-updok (recommended)

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
		Within the context of a replication connection between two instances, an originating instance is referred to as source instance or source side. For example, in an B<-A->C replication configuration, A is the source instance for B and C.	-rootprimary (still accepted)
replicating instance (or secondary instance) and propagating instance	N/A for replicating instance or secondary instance.  -propagateprimary for propagating instance	replicating instance.  Within the context of a replication connection between two instances, a replicating instance that receives updates from a source instance is referred to as receiving instance or receiver side. For example, in an B<-A->C replication configuration, both B and C can be referred to as a receiving instance.	-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). Over time, we'll update all GT.M documentation to this standard.

-  denotes a new feature that requires updating the manuals.
-  denotes a new feature or an enhancement that may not be upward compatible and may affect an existing application.
-  denotes deprecated messages.
-  denotes revised messages.
-  denotes added messages.

## Platforms

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

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

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

Platform	Supported Versions	Notes
IBM Power Systems AIX	7.1 TL 5, 7.2 TL 3	<p>Only 64-bit versions of AIX with POWER7 as the minimum required CPU architecture level are Supported.</p> <p>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.</p> <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>
x86_64 GNU/Linux	Red Hat Enterprise Linux 7.9, 8.5; Ubuntu 18.04 LTS, and 20.04 LTS	<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>Due to build optimization and library incompatibilities, GT.M versions older than V6.2-000 are incompatible with glibc 2.24 and up. This incompatibility has not been reported by a customer, but was observed on internal test systems that use the latest Linux software distributions from Fedora (26), Debian (unstable), and Ubuntu (17.10). In internal testing, processes either hung or encountered a segmentation violation (SIG-11) during operation. Customers upgrading to Linux distributions that utilize glibc 2.24+ must upgrade their GT.M version at the same time as or before the OS upgrade.</p> <p>GT.M requires a compatible version of the libtinfo library. On Red Hat, the ncurses-libs and ncurses-compat-libs packages contain the libtinfo library. On Debian/Ubuntu, libtinfo5 and libncurses5 packages contain the libtinfo library. If any of these packages is not already installed on your system, please install using an appropriate package manager.</p> <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.</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 <b>must</b> 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>). The Red Hat Bugzilla identifier for</p>

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



## Note

Debian 11 (bullseye) x86 GNU/Linux is Supportable but not Supported. The 32-bit version of GT.M runs on either 32- or 64-bit x86 platforms; we expect the x86\_64 GNU/Linux version of GT.M to be preferable on 64-bit hardware. Running a 32-bit GT.M on a 64-bit GNU/Linux requires 32-bit libraries to be installed. The CPU must have an instruction set equivalent to 586 (Pentium) or better. If you are using the 32-bit GT.M, please also refer to the notes above on x86\_64 GNU/Linux.

## Platform support lifecycle

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

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

## 32- vs. 64-bit platforms

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

- \* You must compile the application code separately for each platform. Even though the M source code is the same, the generated object modules are different - the object code differs between x86 and x86\_64.
- \* Parameter-types that interface GT.M with non-M code using C calling conventions must match the data-types on their target platforms. Mostly, these parameters are for call-ins, external calls, internationalization (collation) and environment translation, and are listed in the tables below. Note that most addresses on 64-bit platforms are 8 bytes long and require 8 byte alignment in structures whereas all addresses on 32-bit platforms are 4 bytes long and require 4-byte alignment in structures.

## Call-ins and External Calls

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

## Additional Installation Instructions

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




## Caution

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

\* FIS strongly recommends installing each version of GT.M in a separate (new) directory, rather than overwriting a previously installed version. If you have a legitimate need to overwrite an existing GT.M installation with a new version, you must first shut down all processes using the old version. FIS suggests installing GT.M V7.0-001 in a Filesystem Hierarchy Standard compliant location such as `/usr/lib/fis-gtm/V7.0-001_arch` (for example, `/usr/lib/fis-gtm/V7.0-001_x86` on 32-bit Linux systems). A location such as `/opt/fis-gtm/`



V7.0-001\_arch would also be appropriate. Note that the *arch* suffix is especially important if you plan to install 32- and 64-bit versions of the same release of GT.M on the same system.

- \* Use the appropriate MUPIP command (e.g. ROLLBACK, RECOVER, RUNDOWN) of the old GT.M version to ensure all database files are cleanly closed.
- \* Make sure gtmsecshr is not running. If gtmsecshr is running, first stop all GT.M processes including the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP *pid\_of\_gtmsecshr***.
- \* Starting with V6.2-000, GT.M no longer supports the use of the deprecated \$gtm\_dbkeys and the master key file it points to for database encryption. To convert master files to the libconfig format, please click  to download the CONVDBKEYS.m program and follow instructions in the comments near the top of the program file. You can also download CONVDBKEYS.m from <http://tinco.pair.com/bhaskar/gtm/doc/articles/downloadables/CONVDBKEYS.m>. If you are using \$gtm\_dbkeys for database encryption, please convert master key files to libconfig format immediately after upgrading to V6.2-000 or later. Also, modify your environment scripts to include the use of gtmcrypt\_config environment variable.

## Recompile

- \* Recompile all M and C source files.

## Rebuild Shared Libraries or Images

- \* Rebuild all Shared Libraries after recompiling all M and C source files.
- \* If your application is not using object code shared using GT.M's auto-relink functionality, please consider using it.

## Compiling the Reference Implementation Plugin

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

1. Install the development headers and libraries for libgcrypt, libpgpme, 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 libpgpme11-dev libconfig-dev libssl-dev
```

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

2. Unpack \$gtm\_dist/plugin/gtmcrypt/source.tar to a temporary directory.

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

3. Follow the instructions in the README.

- \* Open Makefile with your editor; review and edit the common header (IFLAGS) and library paths (LIBFLAGS) in the Makefile to reflect those on your system.
- \* Define the gtm\_dist environment variable to point to the absolute path for the directory where you have GT.M installed
- \* Copy and paste the commands from the README to compile and install the encryption plugin with the permissions defined at install time



## Caution

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

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

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

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

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

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

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

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

## Upgrading to GT.M V7.0-001

There are two upgrade paths available when you upgrade to GT.M V7.0-000:

- \* **V7 Upgrade Path 1:** Perform a MUPIP EXTRACT -FREEZE on an existing V6 database and then perform MUPIP LOAD on an empty V7 database using replication to minimize downtime. You can also use the MERGE command to move data from V6 to the new V7 database; as with MUPIP EXTRACT, you need to keep the source data in a stable state. If you are using triggers, extract the triggers from the V6 database and load them in the new V7 database.
- \* **V7 Upgrade Path 2:** Continue using your V6 databases with GT.M V7.0-000.

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

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

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



## Note

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



## Important

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

## Managing M mode and UTF-8 mode

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

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

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

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

- \* Actual files for GT.M executable programs (mumps, mupip, dse, lke, and so on) are in the parent directory, that is, the location specified for installation.
- \* Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for UTF-8 mode in the utf8 subdirectory, and one compiled without support for UTF-8 mode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install UTF-8 mode support. Note that on 64-bit versions of GT.M, the object code is in shared libraries, rather than individual files in the directory.
- \* The utf8 subdirectory has files called mumps, mupip, dse, lke, and so on, which are relative symbolic links to the executables in the parent directory (for example, mumps is the symbolic link ../mumps).
- \* When a shell process sources the file gtmprofile, the behavior is as follows:
  - \* If \$gtm\_chset is "m", "M" or undefined, there is no change from the previous GT.M versions to the value of the environment variable \$gtmroutines.
  - \* If \$gtm\_chset is "UTF-8" (the check is case-insensitive),
    - \* \$gtm\_dist is set to the utf8 subdirectory (that is, if GT.M is installed in /usr/lib/fis-gtm/gtm\_V7.0-001\_i686, then gtmprofile sets \$gtm\_dist to /usr/lib/fis-gtm/gtm\_V7.0-001\_i686/utf8).
    - \* On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of \$gtmroutines is \$gtm\_dist(\$gtm\_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

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

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

## Setting the environment variable TERM

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

- \* Some terminfo entries may seem to work properly but fail to recognize function key sequences or fail to position the cursor properly in response to escape sequences from GT.M. GT.M itself does not have any knowledge of specific terminal control characteristics. Therefore,

it is important to specify the right terminfo entry to let GT.M communicate correctly with the terminal. You may need to add new terminfo entries depending on your specific platform and implementation. The terminal (emulator) vendor may also be able to help.

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

```
auto_right_margin(am), clr_eos(ed), clr_eol(el), columns(cols), cursor_address(cup), cursor_down(cud1),
cursor_left(cub1), cursor_right(cuf1), cursor_up(cuu1), eat_newline_glitch(xen1), 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 as that provided by zlib.

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

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

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

# Change History

## V7.0-001

Fixes and enhancements specific to V7.0-001:

Id	Prior Id	Category	Summary
GTM-4272	C9C04-001975	Admin	MUPIP BACKUP displays information in standard GT.M messages format
GTM-4814	C9D01-002231	Language	M-profiling (VIEW "TRACE") restored after ZSTEP
GTM-5148	C9D10-002419	Admin	REPLAHEAD message for ROLLBACK
GTM-8010	-	Language	Appropriate handling of deviceparameters on OPEN "dev/null" and EXCEPTION values in general
GTM-8681	-	Admin	MUPIP BACKUP -RECORD stores the time of its start when it completes successfully ✔
GTM-8843	-	Language	GT.M SOCKET devices support non blocking WRITES ✔
GTM-9057	-	Admin	MUPIP JOURNAL -EXTRACT to a FIFO device ✔
GTM-9131	-	Other	LOGTPRESTART appropriately identifies restarts due to less frequent reduced statsdb file extensions
GTM-9213	-	Language	A process can SET the trailing portion of \$SYSTEM ✔
GTM-9324	-	Language	ZSTEP restored after MUPIP INTRPT or \$ZTIMEOUT; ZSTEP subject to ZBREAK restriction ✔
GTM-9333	-	Other	Handle one asynchronous event of any type at a time
GTM-9363	-	Admin	🔴 Source Server polling and TLS renegotiation improvements ✔
GTM-9373	-	Admin	Additional information from MUPIP REPLICATE - SOURCE -SHOWBACKLOG ✔
GTM-9378	-	Language	Appropriate handling of pattern match in large compilations
GTM-9382	-	Other	Fix V7.0-000 issue with symbolic links and relative addresses for \$gtm_dist
GTM-9388	-	Language	ZSHOW "B" shows the action ✔
GTM-9392	-	DB	NOISOLATION characteristics maintained correctly when databases and global directories are not aligned
GTM-9400	-	Admin	MUPIP REORG prevents MUPIP STOP from causing KILLABANDONED

### Change History

Id	Prior Id	Category	Summary
GTM-9405	-	Other	Better diagnostic information from MUPIP JOURNAL for errors reading and writing flat files
GTM-9408	-	Language	Prevent an occasional indefinite HANG
GTM-9409	-	Other	More context with certain JOBFAIL errors
GTM-9410	-	Admin	GT.M components try to ensure a missing global directory does not mean a GDE is writing an updated version
GTM-9416	-	Admin	Receiver server started with -REUSE continues to operate across a connection reset
GTM-9422	-	Other	🔴 Counter statistics replace waiting state for critical code section waits 🟢
GTM-9423	-	Admin	MUPIP DUMPFHEAD recognizes the -FLUSH qualifier 🟢
GTM-9424	-	Admin	🔴 Automatically select the best copy mechanism for MUPIP BACKUP 🟢
GTM-9425	-	Admin	\$GTCM_<node-name> accepts "host-name:port-number"
GTM-9429	-	Language	🔴 Features such as \$QLENGTH() and \$QSUBSCRIPT() do tighter checking for canonic references
GTM-9432	-	Other	ztriggers with GT.CM errors out with REMOTEDBNOTRIG
GTM-9434	-	DB	Maximum tree depth of 11 🟢
GTM-9437	-	Language	🔴 USE for SOCKET improvements
GTM-9441	-	Admin	Cleaner build for the encryption plug-in on AIX
GTM-9442	-	Admin	Scripts no longer rely on /usr/bin/which
GTM-9443	-	Admin	MUPIP SET -JOURNAL more cautious with the journal file chain
GTM-9444	-	Language	🔴 Change in behavior when an indirect argument to a potentially argumentless command evaluates to an empty string
GTM-9451	-	DB	TPNOACID releases database critical sections for LOCK operations that hang for lack of LOCKSPACE
GTM-9452	-	Language	CLOSE deviceparameter REPLACE overwrites an existing file, which RENAME does not

---

## Database

- \* GT.M correctly maintains NOISOLATION characteristics for globals. Due to a regression in V6.3-003, the application of NOISOLATION to globals may not have worked when there was a configuration difference between a region's maximum key size in the Global Directory and the database file header. The workaround was to ensure that the maximum key size settings are the same in the Global Directory and the database file header. (GTM-9392)
- \* GT.M databases support a maximum tree depth of 11 levels; previously the limit was 7. The workarounds were to use a large block size, or to map the data into multiple files. (GTM-9434) 🟢
- \* GT.M issues a TPNOTACID message and releases all database critical sections it owns during any LOCK operation in the final retry that could result in an indefinite hang, e.g. LOCKSPACE full. Previously, LOCK operations with a timeout less than \$gtm\_tpnotacidtime (or the default of 2 seconds), would not generate such an action. As a result, a process could hang in the final transaction processing retry. (GTM-9451)

---

## Language

- \* GT.M restores TRACE operation (M-Profiling) after ZSTEP operations. However, issuing a VIEW "[NO]TRACE" may interfere with ZSTEP operations. Note that using ZSTEP materially impacts M-Profiling times, so using these two facilities together may be problematic. Previously, ZSTEP usage usually turned off M-Profiling. (GTM-4814)
- \* OPEN "/dev/null" with deviceparameters works appropriately. The only deviceparameters GT.M actually attempts to implement for /dev/null are [NO]WRAP and EXCEPTION= and, at least in theory, the device should never give an exception. Previously, such an OPEN handled deviceparameters inappropriately, which could cause unintended WRAP behavior or an attempt to instantiate an exception handler constructed out of garbage, which, in turn, could cause a segmentation violation (SIG-11). The workaround was to specify no deviceparameters on an OPEN of /dev/null. In addition, GT.M appropriately handles EXCEPTION= values settings whose lengths are between 128 and 255 bytes long. Previously, GT.M mishandled such settings potentially resulting in a segmentation violation (SIG-11). The second issue has not been reported by a customer, but was observed in development. (GTM-8010)
- \* GT.M SOCKET devices support non blocking WRITES. For more information, refer to the Additional Information section. (GTM-8843) 🟢
- \* GT.M accepts SET \$SYSTEM=expr, where expr appends to the initial value up to the length permitted for an initial value; an empty string removes any current added value. Initial values are determined by the \$gtm\_sysid environment variable always preceded by "47,". Previously, an attempt to SET \$SYSTEM produced an SVNOSET error. (GTM-9213) 🟢
- \* GT.M restores ZSTEP operations after an asynchronous event such as MUPIP INTRPT or \$ZTIMEOUT; previously asynchronous events implicitly removed any pending ZSTEP operations. In addition, a restriction configured for ZBREAK also applies to ZSTEP; previously it did not. If you prefer we separate these restrictions, please let FIS know. (GTM-9324) 🟢
- \* The GT.M compiler appropriately manages the heap during compilation of pattern match (?) operations; previously it could mishandle an address that produced a segmentation violation at compile time or an ASSERTPRO at runtime, typically with large patterns. (GTM-9378)
- \* ZSHOW "B" output consists of an entryref followed by any associated code with a right angle-bracket delimiter after the entryref. Previously the output only contained the entryref. (GTM-9388) 🟢
- \* The HANG command avoids a race condition where a non-zero duration could occasionally hang indefinitely. The change makes things, including \$HOLOROG and \$ZUT, that rely on the system clock more sensitive to changes which adjust that resource. The workaround for this was to wake the affected process with a SIGALRM, and change any HANG that exhibited the symptom to use a timed READ of some non-responding device in place of the HANG. (GTM-9408)
- \* \$QLENGTH() and \$QSUBSCRIPT() report errors when a literal portion of the namevalue argument contains a leading decimal point (.) or minus-sign (-) not followed by one or more numeric digits, or text matching the appearance of a \${Z}CHAR() function. Previously these cases were not appropriately detected. (GTM-9429) 🟡
- \* Making a socket current in a SOCKET device with "USE dev:SOCKET=handle" is now significantly faster when no other device parameters are specified. Previously, the operation was slowed down by preparations needed by other device parameters. In addition, USE ATTACH and USE DETACH issue an error if additional device parameters are specified. Previously, they silently ignored the extra device parameters. (GTM-9437) 🟡
- \* When a command has an argument in the form @indir where indir evaluates to the empty string, GT.M treats it as equivalent to the argument being an empty string, which typically generates an error; the exceptions being IF, QUIT, and TROLLBACK where the indirect-derived argumentless form acts the same as an empty string argument, and also cases where the empty string acts as a NOOP, namely: HANG, WRITE, XECUTE and ZSHOW. For example: DO @x where ""=x now produces a LABELEXPECTED error. Previously with this form, some potentially argumentless commands (DO, KILL, LOCK, NEW, and ZDEALLOCATE) behaved as if argumentless, and certain Z\* commands behaved as NOOPs. We believe this is more in conformance with the language definition and less likely to violate the principal of "least surprise." (GTM-9444) 🟡
- \* CLOSE accepts REPLACE=<file-name> as a deviceparameter, to overwrite an existing file. RENAME or REPLACE that specifies the original name simply closes the file. Previously GT.M only provided RENAME on CLOSE which intentionally protected against accidental replacement, including of itself. (GTM-9452)



---

## System Administration

- \* MUPIP BACKUP and RESTORE display information in standard GT.M messages format. The messages display the full path when they include a file context. Please refer to the Error and Other Messages section of these release notes for details on the standard messages. Previously, certain MUPIP BACKUP and RESTORE messages did not follow the standard GT.M message format and/or did not display the full path when there was a file context. (GTM-4272)
- \* The Receiver Server reports a REPLAHEAD warning when it detects the replicating instance is ahead of the originating instance. The message also includes the backlog count, and information on whether the Receiver Server requires a manual rollback. Previously, the Receiver Server recorded the REPL\_ROLLBACK\_FIRST message which lacked clarity on the Receiver Server actions. (GTM-5148)
- \* MUPIP BACKUP -RECORD provides a timestamp marking the start of backup when a backup a backup completes with no errors. The timestamp provides a backup date which is meaningful to operators without regard to when a particular transaction might have occurred. MUPIP DUMPFHEAD characterizes the timestamp as "sgmnt\_data.last\_start\_backup", while DSE DUMP -FILEHEADER, labels it as "Last Record Backup Start". If a database has never been backed up with MUPIP BACKUP -RECORD, the utilities display "Never" for the field. Previously the -RECORD option stored only a starting transaction number. (GTM-8681) 🟢
- \* MUPIP JOURNAL -EXTRACT accepts a named pipe (FIFO) as its output device. A process needs to open one end of the FIFO (in read mode) and the device can then be passed as an extract output device. Previously, such extracts could not be written into a FIFO. (GTM-9057) 🟢
- \* The Source Server performs its polling activities in a more CPU-efficient manner. Also, the Source Server schedules a TLS renegotiation after an appropriate multiple of heartbeat events. This prevents a TLS renegotiation from interfering/overlapping with the normal replication message interchange mechanism and eliminates the need of a separate timer event for running periodic TLS renegotiation. Previously, the Source Server used a separate timer for TLS renegotiation which could cause a race condition. MUPIP produces the MUPCLIERR error when the specified heartbeat interval (the fifth parameter of -CONNECTPARAMS) is larger than the TLS renegotiate interval. Previously, it did not report this condition as an error. The default value of the heartbeat max period (the sixth parameter) of -CONNECTPARAMS is 300 seconds. Previously, the value was 60 seconds. The Receiver Server logs an REPLCOMM information message when the connection breaks during a message exchange. Previously, the Receiver Server reported this event in its log file but not as an REPLCOMM informational message.. (GTM-9363) 🟡🟢
- \* MUPIP REPLICATE -SOURCE -SHOWBACKLOG considers a transaction as backlogged until it is acknowledged from the Receiver Server. The SRCBACKLOGSTATUS message reports whether a Receiver Server is behind, ahead, or has not yet acknowledged the transactions. The LASTTRANS message reports the state (posted, sent, or acknowledged) of the Source Server transactions under replication. Previously, the MUPIP REPLICATE -SOURCE -SHOWBACKLOG did not display the SRCBACKLOGSTATUS and LASTTRANS messages, did not consider in-flight transactions as a backlog and did not report when the replicating instance was ahead during conditions such as online rollback. (GTM-9373) 🟢
- \* MUPIP REORG defers MUPIP STOP recognition while performing bit map adjustments after an action that frees a block, which prevents such events from possibly causing KILLABANDONED and associated errors. Previously REORG did not use such a deferral. The workaround was to stop the REORG with a CTRL-C. (GTM-9400)
- \* GDE attempts to avoid inappropriately creating a global directory by retrying its opening of an existing file a number of times; other components that read a global directory use the same technique to ensure a missing global directory is not a transient condition. These additional attempts take a fraction of a second, but one may perceive the additional time. Writing a revised global directory has a short gap between the removal of the prior file and the replacement by the new/revised file, during which another process might find the global directory missing; previously this was unlikely but has been encountered. (GTM-9410)
- \* A Receiver Server of an SI replication started with -REUSE qualifier continues to run normally on a connection reset. Previously on a connection reset, the receiver server exited with GTM-E-REUSEINSTNAME. (GTM-9416)
- \* MUPIP DUMPFHEAD -FLUSH -REGION performs a database file header flush for the specified region(s) before displaying the database file header fields. If the database file header flush fails, MUPIP DUMPFHEAD -FLUSH produces the BUFFLUFAILED warning. The qualifier makes the command considerably more heavy weight, and, in most cases, does not provide material benefit, but there may be



## System Administration

cases where it addresses a need. Previously, MUPIP DUMPFHEAD provided no option to flush the database file header fields. (GTM-9423)



- \* MUPIP BACKUP -DATABASE uses what seems to be the best copy mechanism available on the kernel to create a backup copy of the database. If the copy mechanism supports monitoring, MUPIP BACKUP -SHOWPROGRESS periodically displays the transfer progress, estimated time left, speed, and the number of transaction applied to the database during backup. Pressing CTRL\_C performs the appropriate cleanup and exits the BACKUP command. Previously, MUPIP BACKUP used cp or pax for copy phase and did not have a progress monitoring mechanism. On kernels where the copy mechanism supports robust error handling, MUPIP BACKUP handles error conditions such as ENOMEM, ENOSPC, EIO, and so on with a more descriptive action message.

MUPIP BACKUP displays the BKUPFILPERM message when it finds that there is no write permission for the backup file. MUPIP BACKUP performs this check before starting BACKUP. Previously, MUPIP BACKUP reported this condition after performing the copy.

The -RETRY=n qualifier of MUPIP BACKUP -DATABASE makes n number of attempts to retry performing BACKUP if the backup fails. If -RETRY is not specified, GT.M defaults -RETRY to zero (0). In case of an error, retry attempts are always based on cp or pax. Previously, the -RETRY qualifier was not available. (GTM-9424)  

- \* The environment variable \$GTCM\_<node-name> accepts host-name and port-number in the form "host-name:port-number". Previously, host names were accepted only in the form [host-name]:port-number. (GTM-9425)
- \* The GT.M encryption plugin build looks for libraries in /opt/freeware. Previously, the build could generate plugins based on AIX RPM/YUM encryption packages which obtain incompatible library dependencies at run time from elsewhere (e.g. /usr/local/lib). [AIX] (GTM-9441)
- \* The gtminstall script wrapper (gtminstall.sh) and the reference encryption plugin scripts no longer rely on the presence of the utility /usr/bin/which. (GTM-9442)
- \* When MUPIP SET -JOURNAL encounters a temporary journal file that is an artifact of the renaming process GT.M uses to create a chain of journal files, it only deletes it after concluding that has been abandoned by seeing that it persists for an interval longer than the renaming process should take. Previously when MUPIP encountered a temporary journal file, it assumed the file was an abandoned artifact and immediately deleted it potentially inappropriately breaking the chain. (GTM-9443)

---

## Other

- \* LOGTPRESTART appropriately identifies restarts associated with extensions of a statsdb database; Previously, it inappropriately identified these as caused by a BITMAP conflict. GT.M doubles the block count with each statsdb size increase; Previously, GT.M used a fixed extension size of 2050 blocks. GT.M saves the database block count after each extension and uses it as the initial size for any subsequent recreation of the statsdb database during the life of the associated database file; Previously, GT.M always created a statsdb database with 2050 blocks, which is still the initial size for such a database file when the corresponding database is first created. (GTM-9131)
- \* At any point GT.M recognizes only one request for any type of asynchronous processing, including CTRL-C, CTRAP, MUPIP INTRPT, \$ZMAXTPTIME, and \$ZTIMEOUT. Note that MUPIP INTRPT (SIGUSR1) and untrapped CTRL-C can interrupt other asynchronous events, and an untrapped CTRL-C cancels any other pending asynchronous processing. Previously, GT.M could inappropriately attempt to handle multiple requests for a single type of asynchronous operation, which caused unintended behavior, most likely a stack overflow. The workaround was to avoid rapid interrupting. (GTM-9333)
- \* GT.M appropriately handles symbolic links and relative paths in the \$gtm\_dist path. In V7.0-000, GT.M issued a GTMSECshrPERM or SYSCALL error when the first process attempting access to a previously quiescent database file had read-only permissions to the file. Other actions that require gtmsecshr (e.g., when dealing with processes or resources created or owned by a different user, such as waking or checking for the existence, and removing or instantiating a resource, such as semaphores, shared memory or socket files) could result in such error messages as well. The workaround was to avoid symbolic links and relative paths for V7.0-000 (GTM-9382)
- \* MUPIP JOURNAL reports error conditions related to file reads and file writes both for sequential devices and non-database product-specific files more accurately. Previously, MUPIP always displayed "ENO1 : operation not permitted" for such error conditions. (GTM-9405)
- \* GT.M appropriately reports the underlying cause of a JOBFAIL error due to an issue setting up a socketpair to transfer context to the new JOB process. Previously, it did not supply this information. (GTM-9409)
- \* GT.M provides counter statistics designated DEXA, GLB, JNL, MLK, PRC, TRX, ZAD, JOPA, AFRA, BREA, MLBA & TRGA, which accumulate counts when a process waits for a critical code section in various cases. These counters are documented in the "ZSHOW" section of the GT.M Programmers Guide. Previously, GT.M provided toggle statistics to indicate when a process was waiting for a critical code section. (GTM-9422) 🍏 🍏
- \* A ZTRIGGER command involving a remote GT.CM database errors out with GTM-E-REMOTEDBNOTRIG. Previously, it resulted in a segmentation violation (SIG-11). (GTM-9432)

---

## Additional Information

---

### Additional Information about GTM-8843

Sockets default to blocking WRITES. WRITE /BLOCK("OFF") enables non blocking WRITES for the current socket of the current SOCKET device.

A socket must be enabled for non blocking WRITES before enabling it for TLS if both features are desired.

For non blocking sockets, GT.M retries a WRITE that blocks up to the number of times specified by the \$gtm\_non\_blocked\_write\_retries environment variable with a 100 millisecond delay between each retry. The default retries value is 10 times if \$gtm\_non\_blocked\_write\_retries is not defined.

If WRITE remains blocked after the specified retries, the WRITE sets \$DEVICE to "1,Resource temporarily unavailable" and issues an error if IOERROR is "TRAP".

If IOERROR is not "TRAP", \$DEVICE must be checked after each WRITE. An attempt to WRITE to a socket after it has been blocked is an error which sets \$DEVICE to "1,Non blocking WRITE blocked - no further WRITES allowed". Thus the only operation permitted on a blocked socket is to CLOSE it.



#### Note

Note that multi-argument WRITES are equivalent to a series of one argument WRITES, and that GT.M turns unparenthesized concatenation within a write argument into multi-argument WRITES. Format control characters such as "!" and "#" are each considered as an argument.

Note that a significant delay between bytes for any reason, including blocking, especially within a multibyte character when CHSET is UTF-8, may be considered an error by the receiving end of a connection. If the application is unable to handle such a delay, it may result in an application error.

A WRITE to a non blocking socket, which is not enabled for TLS, may terminate early on the following events:

<CTRL-C>, exceeding \$ZMAXTPTIME, or \$ZTIMEOUT expiring. These events result in a transfer to the interrupt vector or error handler at the next execution boundary as described in the "Interrupt Handling" section of the GT.M Programmer's Guide.

When non blocking WRITES are enabled for a socket, WRITE /WAIT may check if that socket would not block on WRITE in addition to READ. The optional second argument may contain a string containing "READ" and/or "WRITE".

If the second argument is omitted or specifies both "READ" and "WRITE" and the socket selected by WRITE /WAIT is ready for both READ and WRITE, \$KEY contains:

```
READWRITE|<socket handle>|<address>.
```

If the second argument is omitted or contains "WRITE", WRITE /WAIT checks readiness for WRITE on non blocking sockets, but never checks readiness to WRITE on blocking sockets, even if explicitly requested.

If the socket selected by a WRITE /WAIT which implicitly or explicitly requests the write state would block on a READ but not block on WRITE, \$KEY contains:

```
WRITE|<socket handle>|<address>
```

Note that a WRITE may still not be able to complete if it tries to write more bytes than the system is ready to accept.



## Note

In most circumstances, WRITE /WAIT or WRITE /WAIT(timeout,"WRITE") for SOCKET devices which contain a non blocking socket returns immediately because non blocking sockets are usually ready for writing.

If the socket selected by WRITE /WAIT which implicitly or explicitly requests the read state would not block on a READ but would block on a WRITE, \$KEY contains:

```
READ|<socket handle>|<address>
```

If the second argument only specifies "WRITE", WRITE /WAIT does not check incoming connections for listening sockets.

The optional third argument to WRITE /WAIT can be used to check only a single socket instead of all sockets in the current SOCKET device by specifying the handle name of a socket:

```
WRITE /WAIT[(timeout[, [what][, handle]])]
```

\$ZKEY after a prior WRITE /WAIT will contain a piece of the format "WRITE|sockethandle|ipaddress" if a non blocking socket was considered writable which will be the case most of the time. If a socket was also readable, there will be two pieces in \$ZKEY for the socket, one for WRITE and the other for READ.

Whether a socket is enabled for non blocking WRITES may be determined by:

```
$ZSOCKET(device, "BLOCKING", index)
```

which returns either 1 (TRUE) for blocking, or 0 (FALSE) for non blocking.

---

## Error and Other Messages

---

### **BACKUPDBFILE**

**BACKUPDBFILE**, DB file dddd backed up in file bbbb

MUPIP Information: This message indicates MUPIP BACKUP successfully backed up database file dddd to file bbbb.

Action: None required.

---

### **BACKUPFAIL**

**BACKUPFAIL**, MUPIP cannot start backup with the above errors

MUPIP Error: This message indicates MUPIP BACKUP was unable to complete the backup.

Action: Review accompanying messages for action guidance.

---

### **BACKUPREPL**

**BACKUPREPL**, Replication Instance file iiii backed up in file rrrr

MUPIP Information: This message indicates that MUPIP BACKUP was successful in backing up replication instance file iiii to file rrrr.

Action: None required.

---

### **BACKUPSEQNO**

**BACKUPSEQNO**, Journal Seqnos up to 0xhhhh are backed up

MUPIP Information: This message indicates MUPIP BACKUP -REPLINSTANCE backed up journal sequence numbers up to 0xhhhh to the specified replication instance file.

Action: None required.

---

### **BACKUPSUCCESS**

**BACKUPSUCCESS**, Backup completed successfully

MUPIP Information: This message indicates the backup actions specified with MUPIP BACKUP command were successful. MUPIP does not display this message until all actions are complete.

Action: None required.

---

### **BACKUPTN**

**BACKUPTN**, Transactions from 0xbbbb to 0xeeee are backed up

MUPIP Information: This information message indicates MUPIP BACKUP backed up transactions from 0xbbbb to 0xeeee.

Action: None required.

## BKUPFILEPERM

**BKUPFILEPERM**, Backup file dddd does not have write permission

MUPIP Information: MUPIP BACKUP encountered an authorization issue with the target location while preparing to perform the BACKUP.

Action: Ensure the target location has appropriate authorization and the appropriate user is properly configured.

---

## BKUPPROGRESS

**BKUPPROGRESS**, Transfer : cccc / tttt (pppp%) ; Speed : zzzz MiB/sec ; Transactions : nnnn ; Estimated left : tt minutes

MUPIP Information: MUPIP BACKUP -SHOWPROGRESS displays this message when the kernel supports copy progress monitoring. cccc is the size (MiB or GiB) of the copied database file and tttt is the total size of the database file. pppp is the progress percentage. Speed is always in MiB/sec and can vary based on the resources available for copy. Transactions includes the number of transaction (increments of current tn), applied to the region during MUPIP BACKUP. If the kernel does not support progress monitoring, MUPIP BACKUP -SHOWPROGRESS does not report this message. This message is expected to appear after about 25% completion of copy. Note that GT.M instructs the kernel to copy as much data as possible in one go. If the kernel has available resources and the database file size is relatively small, you may only see one BKUPPROGRESS message followed by the BACKUP COMPLETED message.

Action: None required.

---

## BKUPRETRY

**BKUPRETRY**, Retrying MUPIP BACKUP for region: rrrr (database file: dddd). Attempt: #nnnn of mmmm

MUPIP Information: This message appears when MUPIP BACKUP initiates a retry attempt because a prior backup attempt failed. #nnnn is the current retry attempt count and mmmm is the maximum number of retry attempts.

Action: None required.

---

## CMDERR

**CMDERR**, Error running command : cccc

MUPIP Error: This message indicates MUPIP BACKUP received an error trying to execute the shell command cccc.

Action: Look at the error message and preceding messages. Check for errors in paths, authorizations and/or other possibilities related to the specified MUPIP BACKUP actions.

---

## CRYPTNOV4

**CRYPTNOV4**, ffff is an encrypted database. Cannot downgrade(to V4) with Encryption option enabled

MUPIP Error: An attempt to downgrade ffff which is an encrypted database to the V4 (GT.M version 4) format failed because the V4 format does not support encrypted database files.

Action: Use the database in the current format. If a V4 format is required, extract the data in unencrypted ZWRite format with MUPIP EXTRACT and load it into a newly created V4 database.

---

## DBMISALIGN

**DBMISALIGN**, Database file xxxx has yyyy blocks which does not match alignment rules. Reconstruct the database from a backup or extend it by at least zzzz blocks.

---

## Error and Other Messages

**MUPIP Error:** This error appears when GT.M detects a mismatch between the total block count in the file header and the expected block count based on the database file size reported by the file system. This error may appear when you perform a MUPIP INTEG -FILE after a GT.M upgrade on a database file which has not yet been opened by a process using a normal database access which performs an automatic database file header upgrade.

**Action:** If there are prior messages, address them first. Extend the database by at least one block, perform at least one \$GET() operation or run MUPIP INTEG -REGION. If the error persists, reconstruct the database from a backup.

---

## DIRACCESS

**DIRACCESS,** Do not have full access to directory for temporary files: pppp

**MUPIP Error:** The message indicates that MUPIP BACKUP does not have appropriate access to the temporary directory pppp.

**Action:** Check the path and the directory permissions for the temporary directory. You can also set the gtm\_baktmpdir environment variable to specify the location of the temporary directory.

---

## LASTTRANS

**LASTTRANS,** Last transaction sequence number SSSS : NNNN

**MUPIP Information:** This message appears with the output of MUPIP REPLICATE -SOURCE -SHOWBACKLOG. SSSS denotes the three states the latest transaction sequence number - posted, sent, and acknowledged. NNNN denotes the associated count for each state. A transaction is first "posted" on the Journal Pool, "sent" to the Receiver Server, and finally "acknowledged" once the Source Server receives confirmation that it has reached the Receiver Server.

**Action:** None

---

## NORTN

**NORTN,** Routine name missing

**Run Time Error:** This indicates the specification used to locate a routine for compilation and / or zlinking was missing the name.

**Action:** Correct the routine specification.

---

## REPLAHEAD

**REPLAHEAD,** Replicating instance is ahead of the originating instance. aaaa

**MUPIP Error:** The message appears on the Source and Receiver Server log files when the Receiver Server is ahead due to a possible rollback on the Source Server side. aaaa contains additional information or action that the user may have to perform.

**Action:** Action: Acknowledge or perform the appropriate action suggested with aaaa.

---

## REPLCOMM

**REPLCOMM,** Replication subsystem communication failure

**MUPIP Error:** This is a generic error or message indicating that there has been a communication error between the two systems performing replication.

**Action:** Review the accompanying message(s) for more information about the cause of this error. When REPLCOMM has an error severity, it accompanies a shut down of the replicating server. When REPLCOMM has an information severity, it indicates a temporary pause in replication due to a situation described by accompanying messages.



## RESTORESUCCESS

**RESTORESUCCESS**, Restore completed successfully

MUPIP Information: This message indicates MUPIP RESTORE successfully completed all specified with command actions.

Action: None required.

---

## SOCKBLOCKERR

**SOCKBLOCKERR**, WRITE /BLOCK error: dddd

Run Time Error: This indicates a format or usage error in a WRITE /BLOCK command. Specific details are provided by dddd.

Action: Correct the format or usage of the WRITE /BLOCK command.

---

## SOCKWAITARG

**SOCKWAITARG**, nnnn argument to WRITE /WAIT xxxx

Run Time Error: This indicates an error with argument number nnnn of a WRITE /WAIT as described by xxxx.

Action: Correct the specified argument.

---

## SRCBACKLOGSTATUS

**SRCBACKLOGSTATUS**, Instance RRRR SSSS NNNN transaction(s)

MUPIP Information: This message appears with the output of MUPIP REPLICATE -SOURCE -SHOWBACKLOG. RRRR specifies the name of the replicating instance. SSSS denotes three possible stages of the replicating instance in relation to the originating instance - "is behind by", "has not acknowledged" and "is ahead by". A replicating instance is behind by the originating instance when there is a backlog of unacknowledged transactions. A replicating instance is ahead by the originating instance when the Receiver Server is performing an online rollback. An instance has not yet acknowledged transaction when the originating instance has not received a response from the replicating instance. NNNN is the number of transactions. There are no in-flight updates when SRCBACKLOGSTAUS reports that the replicating instance is behind by 0 transactions and the LASTTRANS messages for "posted", "sent", and "acknowledged" have the same number of transaction count.

Action: Use this message as an operational aid to determine the status of the replicating instance in relation to the originating instance.

---