

GT.M

Release Notes

V6.3-007

Contact Information

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

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

Legal Notice

Copyright ©2019, 2022 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	04 July 2022	Added GTM-8188.
Revision 1.0	01 February 2019	V6.3-007

Table of Contents

V6.3-007	1
Overview	1
Conventions	1
Platforms	3
Platform support lifecycle	5
32- vs. 64-bit platforms	5
Call-ins and External Calls	5
Internationalization (Collation)	6
Environment Translation	6
Additional Installation Instructions	7
.....	7
Upgrading to GT.M V6.3-007	9
Stage 1: Global Directory Upgrade	9
Stage 2: Database Files Upgrade	10
Stage 3: Replication Instance File Upgrade	12
Stage 4: Journal Files Upgrade	13
Stage 5: Trigger Definitions Upgrade	13
Downgrading to V5 or V4	14
Managing M mode and UTF-8 mode	15
Setting the environment variable TERM	17
Installing Compression Libraries	17
Change History	19
V6.3-007	19
Database	21
Language	23
System Administration	25
Other	27
More Information	29
Additional information for GTM-7318 - Audit Principal Device	29
Enabling Audit Principal Device (APD)	29
Logging	30
Error and Other Messages	33
APDCONNFAIL 	33
APDINITFAIL 	33
APDLOGFAIL 	33
DEVNAMERESERVED	33
GBLOFLOW 	34
ILLEGALUSE 	34
INVALIDGBL 	34
ORLBKREL 	34
ORLBKRESTART 	34
STATSDBMEMERR 	35
TOTALBLKMAX 	35

TRANS2BIG 	35
UNIQNAME 	36

V6.3-007

Overview

V6.3-007 introduces the the ability to log actions initiated from a principal device including MUMPS commands typed interactively, or piped in by a script or redirect, from the principal device (\$PRINCIPAL) and / or any information entered in response to a READ from \$PRINCIPAL (GTM-7318). Much of the work for this release was to analyze, and in a few cases address, issues identified by a new static scanning approach; any changes associated with this effort should have no user impact. The release brings numerous smaller enhancements, and fixes. See the Change History below. Items marked with 🟢 as those 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.



Note

Messages are not part of the GT.M API whose stability we strive to maintain. Make sure that you review any automated scripting that parses GT.M messages.

Conventions

This document uses the following conventions:

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



Note

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

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

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
originating instance or primary instance	-rootprimary	originating instance or originating primary instance. 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). 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 4, 7.2	<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 instfix -ik IZ87564.</p> <p>AIX 7.1 TL 5 is Supportable.</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 7.5; Ubuntu 16.04 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</p>

Platform	Supported Versions	Notes
		<p>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 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:</p> <ul style="list-style-type: none"> ● Find the directory where libncurses.so is installed on your system. ● Change to that directory and make a symbolic link to libncurses.so.<ver> from libtinfo.so.<ver>. Note that some of the libncurses.so entries may themselves be symbolic links, for example, libncurses.so.5 may itself be a symbolic link to libncurses.so.5.9. <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>Although GT.M itself does not require libelf, the geteuid program used by the GT.M installation script requires libelf (packaged as libelf1 on current Debian/Ubuntu distributions and elfutils-libelf on RHEL 6 & 7).</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 XXXX ensure that your kernel includes commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3). The Red Hat Bugzilla identifier for the bug is 1213487. With</p>

Platform	Supported Versions	Notes
		NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs.
x86 GNU/Linux	Debian 9 (Stretch)	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 CPU must have an instruction set equivalent to 586 (Pentium) or better. 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.

Parameter type	32-Bit	64-bit	Remarks
gtm_ulong_t	4-byte	8-byte	gtm_ulong_t is much the same as the C language unsigned long type.
gtm_int_t	4-byte	4-byte	gtm_int_t has 32-bit length on all platforms.
gtm_uint_t	4-byte	4-byte	gtm_uint_t has 32-bit length on all platforms



Caution

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

Internationalization (Collation)

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



Important

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

Environment Translation

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



Important

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


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 V6.3-007 in a Filesystem Hierarchy Standard compliant location such as `/usr/lib/fis-gtm/V6.3-007_arch` (for example, `/usr/lib/fis-gtm/V6.3-007_x86` on 32-bit Linux systems). A location such as `/opt/fis-gtm/V6.3-007_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 `gtmencrypt_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 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 libcrypt, 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 libcrypt11-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

There 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 existing directions above
- Remove /usr/local/ssl to avoid future complications

Upgrading to GT.M V6.3-007

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 differs for 32-bit and 64-bit GT.M releases for the x86 GNU/Linux platform.

GT.M upgrade procedure for V6.3-007 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 V6.3-007 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 one-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 V6.3-007.
- 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 the specified file.
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 V6.3-007.
- 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 V6*:

There is no explicit procedure to upgrade a V6 database file when upgrading to a newer V6 version. After upgrading the Global Directory, opening a V6 database with a newer V6 GT.M process automatically upgrades fields in the database fileheader.

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 992Mi 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 V6.3-007 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. A database file created with V6.0-000 (or above) can grow up to a maximum of 992Mi blocks. This means that, for example, the maximum size of a V6 database file having 8KiB block size is 7936GiB (8KiB*992Mi).



Important

In order to perform a database downgrade you must perform a MUPIP INTEG -NOONLINE. If the duration of the MUPIP INTEG exceeds 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 and does not itself span regions; 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

V6.3-007 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.3-007 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 V6.3-007 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 V6.3-007 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 V6.3-007 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/V6.3-007.

To extract and reapply the trigger definitions on V6.3-007 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 V6.3-007, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.3-007 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 V6.3-007.
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 in 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 V6.3-007 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

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_V6.3-007_i686`, then `gtmprofile` sets `$gtm_dist` to `/usr/lib/fis-gtm/gtm_V6.3-007_i686/utf8`).
 - On platforms where the object files have not been placed in a `libgtmutil.so` shared library, the last element of `$gtmroutines` is `$gtm_dist($gtm_dist/..)` so that the source files in the parent directory for utility programs are matched with object files in the `utf8` subdirectory. On platforms where the object files are in `libgtmutil.so`, that shared library is the one with the object files compiled in the mode for the process.

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

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

Setting the environment variable TERM

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

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

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

This page is intentionally left blank.

Change History

V6.3-007

Fixes and enhancements specific to V6.3-007:

Id	Prior Id	Category	Summary
GTM-4796	C9D01-002213	Language	🔴 Reserved name for socket pool 🟢
GTM-7318	-	Admin	Audit Direct Mode facility 🟢
GTM-8130	-	Other	Modify %GSEL to deal appropriately with names and wildcards 🟢
GTM-8188	-	Admin	Library loading restriction improvements; Trigger piece parsing handles integers close to MAXPOSINT; 🟢
GTM-8626	-	Other	MUPIP JOURNAL requires different names for different output files 🟢
GTM-8653	-	Admin	Prevent potential deadlock between MUPIP JOURNAL ROLLBACK and MUPIP REPLICATE -SOURCE -FREEZE=OFF
GTM-8665	-	Admin	Improve reporting on an interrupted MUPIP JOURNAL -RECOVER/-ROLLBACK 🟢
GTM-8729	-	Other	MUPIP JOURNAL support for -NOLOSTTRANS and -NOBROKENTRANS 🟢
GTM-8871	-	DB	Processes survive a StatDB out-of-space, and maintain statistics when switching between private and shared
GTM-8872	-	Other	Please see GTM-8871
GTM-8904	-	Other	Revise ^%RCE to not lose files when changing routines on multiple file systems
GTM-9003	-	DB	🔴 Critical Resource Management Changes 🟢
GTM-9042	-	Language	Resolve some issues with \$ZTIMEOUT 🟢
GTM-9043	-	Language	The compiler detects too much concatenation in an expression in a way

Id	Prior Id	Category	Summary
			that does not prevent production of an object file
GTM-9047	-	Language	🔴 Improved \$ZCSTATUS presentation of compilation results 🟢
GTM-9049	-	Language	Adjust WRITE of a concatenation expression in non side effect mode for consistency
GTM-9053	-	Admin	Resource management fixes for a couple of unusual cases
GTM-9056	-	Admin	MUPIP SET for -WRITES_PER_FLUSH and -TRIGGER_FLUSH, both of which persist 🟢
GTM-9058	-	Language	🔴 JOB error handling changes
GTM-9061	-	Other	^%YGBLSTAT returns an empty string when directed to report on a nonexistent process
GTM-9065	-	Other	🔴 GDE treats canonic numeric subscripts as numerics rather than strings 🟢
GTM-9071	-	Language	Fix ZMESSAGE to allow Boolean expressions in its argument
GTM-9072	-	Admin	GBLOFLOW message identifies the database file rather than a global
GTM-9073	-	Other	Relationship between Maximum Key Size and Maximum Reserved Bytes. 🟢
GTM-9074	-	Other	GDE accepts values between 2048 and 8388607 for JOURNAL ALLOCATION 🟢
GTM-9075	-	Other	GDE accepts values within quotes and exits on first out of bounds error

Database

- If a GT.M process receives a SIGBUS (SIG-7) signal when attempting to register itself in a statistics database, it prints a STATSDBMEMERR message in the system log, indicating the need to provide sufficient space for the StatsDB file to expand. The process then reverts to process-private statistics collection. After the cause of the SIGBUS has been addressed, the process may turn statistics database logging back on by doing a VIEW "STATSHARE":"REGION", at which point the process returns to doing updates to the corresponding statistics database. Previously, a GT.M process that received a SIGBUS while trying to do statistics database logging terminated, producing a core file. Additionally, when a GT.M process changes between private and shared statistics collection, it copies the current statistics into the appropriate statistics location so collected statistics persist. Previously, GT.M did not properly copy over process-private statistics, meaning that switching over to a statistics database lost any previously collected statistics. (GTM-8871)
- GT.M uses facilities provided by the operating system to protect critical database and journal pool resources on Linux. Previously, GT.M used a combination of multiple operating system facilities and its own logic to provide this protection. Several ZSHOW "G" mnemonics behave differently with this change. CQS, CQT, CYS, and CYT are not maintained and contain zeros. CFT and CFE are maintained, but are only incremented a single time for each observed instance of contention, whereas previously it counted the number of low-level synchronization operations performed, which would typically have been significantly larger. CFS is incremented a single time along with CFT (as the square of one is one.) CAT is maintained as before. \$VIEW("PROBECRIT") returns valid CPT and CAT fields, but zero for all other fields. [Linux] (GTM-9003) 🍏 🍏

This page is intentionally left blank.

Language

- SOCKET devices use "YGTMSOCKETPOOL" to identify the socket pool; an attempt OPEN a device of that name produces a DEVNAMERESERVED error. Note this change requires adjustment of any code explicitly referencing the socket pool. Previously, SOCKET devices used the name "socketpool" to designate the socket pool and an OPEN of a device with that name prevented the use of the pool or access to any devices sockets in it. (GTM-4796) 🚫🟢
- GT.M defers acting on a \$ZTIMEOUT interrupt while it controls any critical database resource and processes the interrupt after their release. In V6.3-006, GT.M did not take this precaution against a \$ZTIMEOUT interfering with a database commit. GT.M handles \$ZTIMEOUT values correctly when the timeout is followed by a colon (:). In V6.3-006, setting \$ZTIMEOUT to any value followed by a colon caused in an immediate timeout. GT.M validates the \$ztimeout XECUTE string before installing it as thetimeout handler. Previously poorly formed XECUTE strings could error out resulting in process termination. ZSHOW "I" and ZWRITE display \$ZTIMEOUT; the initial release omitted those functionalities. (GTM-9042) 🟢
- GT.M detects the case of more concatenation operands in a row than it can handle when parsing the source code; previously it detected this at code generation, which meant it always failed to create an object file in this case. (GTM-9043)
- \$ZCSTATUS holds an indication of the result of the last ZCOMPILE, ZLINK, \$ZTRIGGER() or auto-zlink compilation. One (1) indicates a clean compilation, a positive number greater than one is an error code you can turn into text with \$ZMESSAGE(), and a negative number is a negated error code that indicates GT.M was not able to produce an object file. The error details appear in the compilation output, so \$ZCSTATUS typically contains the code for ERRORSUMMARY. Previously, \$ZSTATUS almost always indicated a one (1) except when object file creation failed. \$ZTRIGGER() and MUPIP TRIGGER don't install trigger definitions with XECUTE strings that do not compile without error; previously they did. In addition, the value for \$ZCSTATUS provided by ZSHOW "I" matches that provided by WRITE \$ZCSTATUS; previously ZSHOW provided a zero (0) when it should have provided a one (1). (GTM-9047) 🚫🟢
- WRITE does not turn an expression starting with a concatenation operation into separate arguments if the expression is within parentheses. WRITE compilation turns a concatenated sequence into separate arguments which, when not processing in side effect mode, can affect the evaluation of side effects. Therefore the documentation contains the following: "The GT.M compiler breaks a concatenated WRITE argument into a series of WRITE arguments to eliminate the overhead of the concatenation. If circumstances provide a reason for a single WRITE, perform the concatenation prior to the WRITE." Previously protecting the concatenation with parentheses in non-side effect mode did not suppress this optimization, which made the result inconsistent with separate evaluation. (GTM-9049)
- GT.M handles JOB errors differently. Errors associated with the specified routine, label, or offset appear in the error file of the JOBbed process in detail in addition to the JOBFAIL error received by the original process. Previously GT.M did not report the more specific errors and did not start the JOBbed process. GT.M detects and reports JOBLVN2LONG errors in the original process;

Language

GT.M does not start the JOBbed process in this case. Previously the JOBbed process would report JOBLVN2LONG to its error file. (GTM-9058) 🍷

- The ZMESSAGE command appropriately handles a Boolean expression within an argument; starting with V6.3-000 such an argument tended to cause a segmentation violation (SIG-11). (GTM-9071)

System Administration

- GT.M supports the ability to log actions initiated from a principal device including MUMPS commands typed interactively, or piped in by a script or redirect, from the principal device (\$PRINCIPAL) and / or any information entered in response to a READ from \$PRINCIPAL. An action initiated from \$PRINCIPAL executes as usual when Audit Principal Device is disabled, which it is by default. However, when Audit Principal Device is enabled, GT.M attempts to send the action out for logging before acting on it. Additionally, the \$ZAUDIT Intrinsic Special Variable (ISV) provides a Boolean value that indicates whether Audit Principal Device is enabled. Please see the Additional information for GTM-7318 - Audit Principal Device in this document for details. (GTM-7318) ✓
- The GT.M restrictions facility recognizes LIBRARY[:<group-name>] to enforce loading of libraries from \$gtm_dist/plugins. The libraries may be actual shared libraries or symlinks to permitted shared libraries. This restriction allows administrators to override the default library search which includes \$LD_LIBRARY_PATH. When present and restriction conditions are met, the restricted library loads produce a RESTRICTEDOP error. GT.M allows PIECE specifications of up to 8192 pieces in trigger specifications. Previously, GT.M incorrectly parsed some large piece values that should have been rejected. (GTM-8188) ✓
- MUPIP JOURNAL -ROLLBACK -ONLINE -BACKWARD, on encountering a frozen region when Instance Freeze is ON, releases all its resources and retries the rollback from the start. Previously, this could cause a potential deadlock with MUPIP REPLICATE -SOURCE -FREEZE=OFF. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-8653)
- MUPIP INTEG reports an interrupted MUPIP JOURNAL -RECOVER/-ROLLBACK operation on the database. Previously, a MUPIP INTEG on such a database did not report an interrupted recovery. Note: The "MUPIP dumphead" command already provided this information. GT.M reports the "Recover interrupted" field with DSE DUMP -FILEHEADER even when journal is turned off. Previously, GT.M reported the "Recovery interrupted" field only with DSE DUMP -FILEHEADER - ALL and only when journaling was turned ON. (GTM-8665) ✓
- GT.M does additional error checking when managing shared resources associated with relinkctl and replication update operations. Previously rare error conditions could leave the resources in an unintended status, resulting in core dumps and interfering with future relinkctl or replication update operations. This issue was only observed in the GT.M development environment, and was never reported by a user. (GTM-9053)
- MUPIP SET {-FILE|-REGION} accepts -TRIGGER_FLUSH=n and -WRITES_PER_FLUSH=n qualifiers; previously only DSE supported these qualifiers. Also, the trigger_flush value appears in MUPIP DUMPFHEAD as trigger_flush_top and acts as a stable limit; previously GT.M tended to lose any user supplied value as it made adjustments intended to improve performance. (GTM-9056) ✓
- GBLOFLOW messages identify the database file that is full; previously they identified the global node the process was updating when it found no room, and in the case of a TP transaction the report was for the last update in the transaction, which was not necessarily even in the full database file. (GTM-9072)

This page is intentionally left blank.

Other

- The %GSEL utility now silently ignores any subscript in the search string and throws a non-fatal error if the input contains an invalid character. Previously, %GSEL would remove invalid characters and then perform the search. This could cause problems if a subscript was present because the utility removed "(" and ")" from the search pattern but not what was between them. This change also applies to %GCE, %GD, %GO, and %GSE which use GD^%GSEL to search for globals. In addition, when used interactively, it attempts to preserve the original I/O state of the caller; previously it tended to leave that state disrupted. (GTM-8130) ✓
- MUPIP JOURNAL does not allow any two of -EXTRACT, -LOSTTRANS or -BROKENTRANS to specify the same file name unless they are special files (-stdout or /dev/null). Previously, MUPIP JOURNAL allowed overlapping file specifications, which lead to unexpected behavior including missing files. specified file was not created. (GTM-8626) ✓
- MUPIP JOURNAL recognizes -NOLOSTTRANS and -NOBROKENTRANS as optional qualifiers which cause it to discard any lost or broken transactions rather than record them in files. Previously, MUPIP JOURNAL always produced file containing any lost and/or broken transactions as there was no such option to discard them. (GTM-8729) ✓
- Please see GTM-8871. (GTM-8872)
- The ^%RCE utility deals appropriately with the case where \$ZROUTINES lists source directories where the target directory is on a different file system than the default (first) source directory. Previously the utility would delete the original, but then fail to move the modified copy to the correct directory. In addition, when used interactively, it attempts to preserve the original I/O state of the caller; previously it tended to leave that state disrupted. (GTM-8904)
- ^%YGBLSTAT skips non-existent processes and returns an empty string when it's sent after a nonexistent PID; Previously it could inappropriately return information on the reporting process or defunct processes. In addition, when used interactively, it attempts to preserve the original I/O state of the caller; previously it tended to leave that state disrupted. Also, the utility versions its local working storage where previously it did not. (GTM-9061)
- GDE treats canonic numeric string subscripts as numeric subscripts, in line with other GT.M utilities. Previously GDE treated them as string subscripts. In addition, GDE SHOW -NAME <NAME> prints the name-region mapping of all subscripts of the specified name. Previously GDE printed the name-region mapping of only the unsubscripted name. (GTM-9065) ✓ ✓
- GDE and MUPIP maintain a consistent relationship between Maximum Key Size and Maximum Reserved Bytes. It adheres to the equation, Maximum Reserved Bytes = Block Size - Key Size - (Size of Block Header + Size of Record Header + Size of Block id + B-star Record Size), which is equivalent to Block Size - Key Size - 32. Previously, they used inconsistent calculations and allowed inconsistent Maximum Key Size. (GTM-9073) ✓

Other

- GDE accepts values between 2048 and 8388607 for JOURNAL ALLOCATION, with 2048 as the default value. Previously, GDE incorrectly accepted values between 200 and 16777216 with 200 as the default value. (GTM-9074) 🟢
- For -ACCESS_METHOD and -NULL_SUBSCRIPTS, GDE accepts valid values enclosed within double quotes. Previously, GDE rejected valid values when enclosed within double quotes. If a value of any qualifier is out of bounds, GDE prints VALTOOSMALL/VALTOOBIG and exits immediately without further processing. Previously GDE continued to process that value and the values of other qualifiers and printed potentially confusing messages. (GTM-9075)

More Information

Additional information for GTM-7318 - Audit Principal Device

Direct Mode receives a command line, GT.M first checks if direct mode auditing or Audit Principal Device (APD) is enabled. If it is disabled, which it is by default, the command executes as usual. If it is enabled, GT.M establishes a connection (via a UNIX/TCP/TLS socket) to a logger/listener process, and sends the to-be-logged command through the socket to the listener for logging. If sending is successful, GT.M executes the logged command. If the connection is not successful or sending of the command fails, then GT.M produces an error and does NOT execute the command.

Enabling Audit Principal Device (APD)

The "APD_ENABLE" entry in a restrictions definition file turns on APD and enables the logging of all code entered from Direct Mode and optionally any input entered on the principal device (\$PRINCIPAL). To enable APD, add a line with the following format to the restriction file:

```
APD_ENABLE:[comma-separated-list-of-options]:{path-to-sock-file|host:port}[:tls-id]
```

- The optional "comma-separated-list-of-options" can consist of zero or more of these options:
 - TLS - Enables TLS connectivity between GT.M and the logger; this option requires the host information (e.g. IP/port or hostname/port)
 - RD - Enables logging of all responses READ from \$PRINCIPAL in addition to that entered at the Direct Mode prompt. This option is more comprehensive and captures input that might be EXECUTEd, but depending on your application architecture may significantly increase the amount of logged information.
- The "path-to-sock-file" is the absolute path of the UNIX domain socket file for connecting to the logger.
- The "host" is the hostname or numeric IPv4/IPv6 address of the logger; numeric IP addresses must be enclosed in square brackets (i.e. '[' and ']').
- The "port" is the port number the logger listens on.
- The optional "tls-id" is the label of the section within the GT.M configuration file that contains TLS options and/or certificates for GT.M to use; APD ignores any "tls-id" if the "TLS" option is not specified.

If parsing the "APD_ENABLE" line in restriction file or initializing logger information fails, GT.M enforces all restrictions (default restriction file behavior).

Examples

```
APD_ENABLE::/path/to/sock/file/audit.sock
```

Adding this line to the restriction file enables APD. GT.M connects with the logger via UNIX domain socket using the domain socket file "/path/to/sock/file/audit.sock" and sends all Direct Mode activity from \$PRINCIPAL to logger.

```
APD_ENABLE:RD:[123.456.789.100]:12345
```

Adding this line to the restriction file enables APD. GT.M connects with the logger (listening on port 12345 at the IPv4 address 1enable23.456.789.100) via TCP socket and sends all Direct Mode and READ activities from \$PRINCIPAL to logger.

```
APD_ENABLE::loggerhost:56789
```

Adding this line to the restriction file enables APD. GT.M connects with the logger (listening on port 56789 at the hostname "loggerhost") using a TCP socket and sends all Direct Mode activities from \$PRINCIPAL to logger.

```
APD_ENABLE:TLS,RD:[1234:5678:910a:bcde::f]:12345:clicert
```

Adding this line to the restriction file enables APD. GT.M connects with the logger (listening on port 12345 at the IPv6 address 1234:5678:910a:bcde::f) via TLS socket. GT.M configures its TLS options for APD based on the contents within the section of the configuration file labeled "clicert". GT.M sends all Direct Mode and READ activities from \$PRINCIPAL to logger.

Logging

The "logger" is a separate server-like program responsible for receiving the to-be-logged information from GT.M and logging it. This separate program must be introduced by the user, either running in foreground or background, in order for logging to actually work. GT.M distributions include basic example logger programs.

The six fields in the message, separated by semicolons (;), contain information on the to-be-logged activity. Each to-be-logged message sent to the logger from GT.M has the following format:

```
dist=<path>; src={0|1|2}; uid=<uid>; euid=<euid>; pid=<pid>; command=<text>
```

- The "dist" field, shows the path to location of the sender/user's \$gtm_dist (GT.M executables).
- The "src" field shows zero (0) for input from unknown source, one (1) for Direct Mode input, or two (2) for READ input from \$PRINCIPAL.
- The next three fields ("uid", "euid", and "pid") show (respectively) decimal representations of the user ID, effective user ID, and process ID of the process that sent the message.
- The "command" field is the input provided on the GT.M side.

Examples

```
dist=/path/to/gtm_dist; src=1; uid=112233445; euid=112233445; pid=987654; command=write  
"Hello world",!
```

```
dist=/usr/library/V123/dbg; src=2; uid=998877665; euid=998877665; pid=123456; command=set  
a=789
```

This page is intentionally left blank.

Error and Other Messages

APDCONNFAIL

APDCONNFAIL, Audit Principal Device failed to connect to audit logger

Run Time Error: The facility for logging activity on principal devices is enabled, but is unable to form a connection with its configured logging program. This prevents a process from taking actions configured for logging initiated on its principal device (\$PRINCIPAL).

Action: Check to make sure logger program is running and listening/accepting connections. If using a TCP or TLS-enabled logger, make sure the port number the logger is listening/accepting on matches the port number provided in the restriction file. Ensure the provided information (logger's connection info) in the restriction file is correct. Also make sure the line in restriction file is in correct format. If running a TLS-enabled logger, make sure the logger's TLS certificate is signed by a root CA that GT.M is aware of through the GT.M TLS configuration file. Check syslog for more information on the error. After addressing identified issues, restart all processes subject to APD.

APDINITFAIL

APDINITFAIL, Audit Principal Device failed to initialize audit information

Run Time Error: GT.M was unable to process or initialize the provided information (e.g. IP, hostname, port number, UNIX domain socket file path, or TLS ID) from the restriction file. This prevents a process from taking actions configured for logging initiated on its principal device (\$PRINCIPAL).

Action: Check the restriction file to make sure information is in proper format. After addressing identified issues, restart all processes subject to APD.

APDLOGFAIL

APDLOGFAIL, Audit Principal Device failed to log activity

Run Time Error: GT.M was unable to send the to-be-logged activity to logger. This prevents a process from taking the action initiated on its principal device (\$PRINCIPAL).

Action: Check to make sure that GT.M is able to successfully connect to the logger program. Check syslog for more information on error.

DEVNAMERESERVED

DEVNAMERESERVED, Cannot use NNNN as device name. Reserved for GTM internal usage.

Run Time Error: This error appears when there is an attempt to OPEN a device with the name YGTMSOCKETPOOL. GT.M internally reserves the name YGTMSOCKETPOOL to identify the socket pool and prevents any other device from using it.

Action: Use a different name for the SOCKET device.

GBLOFLOW

GBLOFLOW, Database file FFFF is full

Run Time/MUPIP Error: This indicates that an error was encountered while extending database file FFFF.

Action: Examine the accompanying message(s) for the cause of the error. If the error is due to insufficient authorization, address that. If the error is due to TOTALBLKMAX (refer to the explanation of that message) or a lack of enough free space on the disk to fit the size of a database file, try performing a KILL of some nodes in the database to get free blocks in the existing allocated space (you may need to KILL several subscripted nodes before you can KILL a name node).

ILLEGALUSE

ILLEGALUSE, Illegal use of the special character "?" in %GSEL

Utility Error: This is an illegal use of the special character "?" in %GSEL. The special character "?" is not valid as the first character of a global name search pattern. "?" only valid as the first character of a search pattern when invoking the commands "?D" or "?d".

Action: Review and re-enter a valid search pattern.

INVALIDGBL

INVALIDGBL, Search pattern is invalid

Utility Error: The search pattern used is invalid due to either using invalid characters or improper formatting.

Action: Review and re-enter a valid search pattern

ORLBKREL

ORLBKREL, ONLINE ROLLBACK releasing all locking resources to allow a freeze OFF to proceed

MUPIP Information: MUPIP ROLLBACK -ONLINE encountered an Instance Freeze and must release its resources and restart to prevent a possible deadlock.

Action: None Required as this is an informational message

ORLBKRESTART

ORLBKRESTART, ONLINE ROLLBACK restarted on instance iiiii corresponding to rrrr

MUPIP Information: MUPIP ROLLBACK -ONLINE is restarting on the instance iiiii with replication journal pool rrrr

Action: None required for this informational message

STATSDBMEMERR

STATSDBMEMERR, Process attempted to create stats block in statistics database SSSS and received SIGBUS--invalid physical address. Check file system space.

Run Time Error: A process attempted to enable shared statistics collection for the region associated with SSSS, but was unable to find room to add its records, so it cannot contribute to sharing. This message goes to the operator log facility rather than the process as an error, but the process continues without shared statistics.

Action: Adjust the environment so that SSSS can expand, and then, if possible, have the process again attempt to enable sharing.

TOTALBLKMAX

TOTALBLKMAX, Extension exceeds maximum total blocks, not extending

Run Time Error: This indicates that the database file extension specified implicitly or explicitly (using MUPIP EXTEND) would cause the GDS file to exceed its maximum size. Please see the most recent GT.M Release Notes, Upgrading to GT.M [LatestVersion], Stage 2:Database File Upgrades section, Important bullet for maximum database sizes.

Action: Modify the extension to use a smaller size. This may indicate that you should move some contents of the database file to another file.

TRANS2BIG

TRANS2BIG, Transaction exceeded available buffer space for region rrrr

Run Time Error: This indicates that a transaction updated more blocks than the global buffer could hold for a particular region rrrr or accessed more than the single transaction limit of 64K blocks.

Action: Look for missing TCOMMIT commands; modify the code to reduce the total content or change content of the transaction. If the transaction is as intended and the issue is the number of updates, increase the GLOBAL_BUFFERS for the region using MUPIP SET, or modify the Global Directory to redistribute the relevant globals to more regions. If this occurs on a replicating instance it may indicate either a difference in configuration between the originating and replicating instances, which probably should be addressed, or a transaction that was borderline on the originating instance, but failed on the replicating instance because of difference in the database layout. In the later case, consider examining the application code to see if it's possible to reduce the size of the transaction, or alternatively increase the global buffers on both the instances.

UNIQNAME +

UNIQNAME, Cannot provide same file name (nnnn) for ffff and FFFF

MUPIP Error: The command species the same name, nnnn for both output ffff and output FFFF.

Action: Revise the command to use unique names for different outputs.