

GT.M

Release Notes

V7.1-003

Contact Information

GT.M Group
Fidelity National Information Services, Inc.
347 Riverside Drive
Jacksonville, FL 13220
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

Legal Notice

Copyright ©2023 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	23 November 2023	V7.1-003

Table of Contents

V7.1-003	1
Overview	1
Conventions	1
Platforms	2
Platform support lifecycle	5
Additional Installation Instructions	5
Recompile	6
Rebuild Shared Libraries or Images	6
Compiling the Reference Implementation Plugin	6
Re-evaluate TLS configuration options	7
Upgrading to V7.1-003	7
Stage 1: Global Directory Upgrade	8
Stage 2: Database Files Upgrade	9
Stage 3: Replication Instance File Upgrade	13
Stage 4: Journal Files Upgrade	14
Stage 5: Trigger Definitions Upgrade	14
Managing M mode and UTF-8 mode	14
Setting the environment variable TERM	15
Installing Compression Libraries	16
Change History	17
V7.1-003	17
Database	19
Language	21
System Administration	23
Other	25
Error and Other Messages	27
EXCEEDRCTLNRNDWN 	27
FDSIZELMT 	27
FORCEDHALT2 	27
GVSUBOFLOW 	27
JNLPVTINFO 	28
RLNKCTLOPENDEL 	28
SOCKCLOSE 	28
SOCKHANGUP 	28
SPCFCBUFDELAY 	29
TLSRENEGOTIATE 	29
WRITERSTUCK 	29

This page is intentionally left blank.

V7.1-003

Overview

V7.1-003 provides a significantly shorter critical section for managing actions associated with Instance Replication, enhancement to support contemporary TLS capabilities and an assortment of other enhancements and fixes.

Items marked with the 🟢 symbol document new or different capabilities.

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



Note

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

Conventions

This document uses the following conventions:

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



Note

The term UNIX refers to the general sense of all platforms on which GT.M uses a POSIX API. As of this date, this includes: AIX and GNU/Linux x86_64.

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, we refer to the combination of operating system and hardware architecture as a platform. We deem this set of specific versions: 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. We deem this larger set of versions: 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 and therefore deem them: 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 reference plugin has its own additional requirements.

Platform	Supported Versions	Notes
IBM Power Systems AIX	7.1 TL 5, 7.2 TL 5, 7.3 TL 1	<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>

Platform	Supported Versions	Notes
		<p>Running GT.M on AIX 7.1 requires APAR IZ87564, a fix for the POW() function, to be applied. To verify that this fix has been installed, execute instfix -ik IZ87564.</p> <p>Only the AIX jfs2 filesystem is Supported. Other filesystems, such as jfs1 are Supportable, but not Supported. FIS strongly recommends use of the jfs2 filesystem on AIX; use jfs1 only for existing databases not yet migrated to a jfs2 filesystem.</p>
<p>x86_64 GNU/Linux</p>	<p>Red Hat Enterprise Linux 7.9, 8.8, 9.3; Ubuntu 20.04 LTS, and 22.04 LTS; Amazon Linux 2</p>	<p>To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware. As of V7.1-001, GT.M on x86-64 requires hardware/virtualized support for AVX instructions.</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 or later.</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 must ensure that your kernel includes commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3).</p>

Platform	Supported Versions	Notes
		<p>The Red Hat Bugzilla identifier for the bug is 1213487. With NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs.</p> <p>Our testing has shown an interaction between glibc 2.36 and all versions of GT.M on Linux/x86_64 systems without AVX2 support. This can cause segmentation violations (SIG-11) in processes performing concurrent updates to the same database block, which terminate the process, but do not damage the database. The issue is due to the way glibc performs certain memory operations when using SSE2 instructions. The glibc behavior was subsequently modified to avoid this issue, and the change was included in glibc 2.37, however, we have not yet confirmed the change resolved issue. Linux/x86_64 systems with support for AVX2 instructions are not vulnerable, as glibc chooses its AVX2 implementation, when available, over its SSE2 implementation, and the problematic behavior is specific to SSE2. Note, depending on how CPU virtualization is configured, that virtual environments may not support AVX2 even if the underlying hardware does.</p> <p>Ubuntu 18.04 LTS and 23.10 are Supportable.</p> <div data-bbox="776 995 857 1079"> </div> <p>Note</p> <div data-bbox="894 1037 1403 1747" style="border: 1px solid black; padding: 5px;"> <p>FIS recommends recompiling the reference encryption plugins to match the target platform. See Compiling the Reference Implementation Plugin section for instructions.</p> <p>OpenSSL 3.0 by default does not allow client-side initiated TLSv1.2 renegotiation requests due to potential DoS attacks. Because of this, the reference TLS implementation in GT.M versions before V7.0-004 do not use the appropriate OpenSSL 3.0 API to enable support for client-side initiated TLSv1.2 renegotiation. Customers needing to replicate to/from GT.M versions before V7.0-004 with OpenSSL 3.0 must use -RENEGOTIATE_INTERVAL=0 in the Source Server startup. This limitation only affects database replication and not SOCKET devices.</p> </div>



Important

Effective V7.0-003, GT.M is no longer Supportable on the 32 bit x86 platform. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.

Platform support lifecycle

FIS usually supports new operating system versions six months or so after stable releases are available, and we usually support each version for a two-year window.

We support GT.M releases in a rolling support model of eight major/minor certified releases, the current release and seven prior quarterly releases. A release becomes no longer officially supported once a given release is beyond the nine release threshold; not to exceed a two year duration. Historically we have produced GT.M releases on a quarterly basis, but that might be subject to change. Note: customers always get the best support by staying current with releases as they are made available.

FIS will continue to attempt to support any release of GT.M in use by a Profile customer under that client's maintenance agreement, while that agreement is still in effect. FIS's ability to provide that level of support may become increasingly costly to the client. In other words, FIS may need to enact a special maintenance agreement to continue to provide support. The additional costs required would be maintain client release level specific servers, operating systems and other ancillary software for a given and reasonable time frame beyond the normal window.

FIS policy is only to provide remediation, in the current release, for identified issues in generally available and supported releases. It is not FIS policy to provide ongoing support of client specific release levels of unsupported software.

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


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 V7.1-003 in a Filesystem Hierarchy Standard compliant location such as `/usr/lib/fis-gtm/V7.1-003_arch` (for example, `/usr/lib/fis-gtm/V7.1-003_x86_64` on Linux systems). A location such as `/opt/fis-gtm/V7.1-003_arch` would also be appropriate.
- 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 `gtmsechr` is not running. If `gtmsechr` is running, first stop all GT.M processes including the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP *pid_of_gtmsechr***.
- 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 `gtmconfig` 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 the example / reference implementation plugin in support of database encryption, TLS replication, or TLS sockets, you must compile the reference plugin in order to match the shared library dependencies specific to your platform. The instructions for compiling the Reference Implementation plugin are as follows:

1. Install the development headers and libraries for `libcrypt`, `libpgme`, `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 libpgme11-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



Note

The encryption plugin currently uses functionality that is deprecated in OpenSSL 3.0. This will be fixed in a future release.

4. When reinstalling or upgrading GT.M, stop existing gpg-agents. The agents may be working with information about the prior GT.M installation, such as `GNUPGHOME`, that will not work with the new version. Additionally, if the process deletes the GPG agent's socket, proper operation requires a new agent.
5. It is a good idea to read the A&O Guide section entitled "Special note - GNU Privacy Guard and Agents" and re-evaluate the GPG configuration options in use.

Re-evaluate TLS configuration options

The GT.M TLS reference encryption plugin implements a subset of options as documented in the OpenSSL man page for `SSL_set_options` which modify the default behavior of OpenSSL. Future versions of the plugin will enable new options as and when the OpenSSL library adds them. To enable options not supported by the GT.M TLS reference plugin, it is possible to create an OpenSSL configuration for GT.M processes. See the OpenSSL man page for "config".

Upgrading to V7.1-003



Before you begin

GT.M supports upgrade from V5*, V6.* and V7.* versions to V7.1-003.

GT.M does not support upgrading from V4* versions. Please upgrade V4 databases using instruction in the release notes of an appropriate GT.M V6.* version.

The GT.M database consists of four types of components- database files, journal files, global directories, and replication instance files.

GT.M upgrade procedure for V7.1-003 consists of 5 stages:

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

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V7.1-003 depends on your GT.M upgrade history and your current version.

Stage 1: Global Directory Upgrade

FIS strongly recommends you back up your Global Directory 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 V7.1-003.
- Execute the EXIT command. This command automatically upgrades the Global Directory.
- 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 V7.1-003.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

An analogous procedure applies in the reverse direction.

Stage 2: Database Files Upgrade

Before starting the database file upgrade, use the prior GT.M version to perform an appropriate MUPIP action (i.e. ROLLBACK, RECOVER, RUNDOWN) to removes abandoned GT.M database semaphores and releases any IPC resources.

There are three upgrade paths available when you upgrade to V7.1-003.

V7 Upgrade Path 1: In-place Upgrade

To upgrade from GT.M V7*:

There is no explicit procedure to upgrade a V7 database file when upgrading to a newer V7 version. After upgrading the Global Directory, opening a V7 database with a newer V7 GT.M process automatically upgrades the fields in the database file header.

To upgrade from GT.M V6* (or V5*):

There are two phases to upgrade from V6 to V7:

- Phase 1: MUPIP UPGRADE phase
- Phase 2: MUPIP REORG -UPGRADE (GVT Index Block Upgrade)

Both phases operate once per region and require standalone access. Phase 1 is not restartable. Phase 2 is restartable.

While these are the basic steps, customers must integrate them with appropriate operational practice and risk mitigating procedures, such as comprehensive testing, backup, integrity checks, journal and replication management, and so on based on their environments and risk tolerance. FIS strongly recommends performing a MUPIP INTEG (-FAST), of the database and creating a backup prior to upgrade. Customers must test these utilities against copies of their own production files, using their planned procedures, before undertaking the conversion of current production files.

While FIS has done considerable testing of MUPIP UPGRADE and MUPIP REORG -UPGRADE, the duration of that testing has not reached the level FIS typically performs for work of this complexity and impact. While our goal is to allow MUPIP REORG -UPGRADE to run with concurrent activity, our testing has not reached a level to allow it to run without standalone access. Using MUPIP UPGRADE and MUPIP REORG -UPGRADE should be a significantly faster alternative to using MUPIP EXTRACT and LOAD. FIS favors using a "rolling" upgrade using a replicated instance. Whatever the approach you choose, FIS requests capturing all logs in case there are issues or questions leading to support requests.

Phase 1: Standalone MUPIP UPGRADE

MUPIP UPGRADE performs Phase 1 actions of upgrading a database to V7. The format of the UPGRADE command is:

```
MUPIP UPGRADE {-FILE <file name>; | [-REGION] <region list>}
```

As the GT.M version upgrade changes the journal format to support 64-bit block pointers, MUPIP UPGRADE does not maintain journal files or replication; configured journaling and replication resumes for activity after MUPIP UPGRADE.

UPGRADE:

- Requires standalone access
- Turns off journaling and replication for the duration of UPGRADE
- When encountering an error where the command specifies multiple regions, UPGRADE moves on to the next region, while for a single file/region, it terminates; avoid any unnecessary <CTRL_C> or MUPIP STOP (or kill) of an active MUPIP UPGRADE process, as such an action leaves the database region effectively unusable
- Estimates and reports the space required for its work
 - UPGRADE estimates are intended to be generous, and, particularly for small databases, they may seem unnecessarily large
 - If MUPIP is not authorized to perform a required file extension, that is, the extension amount is defined as zero (0), it produces an error before it does anything that would damage the selected database file
- Moves blocks from immediately after the existing master map to make room for a V7 master map
 - Depending on the block size and the GT.M version with which it was created, the new starting Virtual Block Number (VBN), the location of block zero for the database file, may exceed the starting VBN for a database created with V7, which causes a minor waste of space
 - UPGRADE relocates blocks in multiples of 512 to align blocks with their local bitmaps
- Eliminates any globals that previously existed, but have been KILL'd at the name level; these global variable trees (GVTs) contain only a level one (1) root block and an empty data (level zero) block and are "invisible" to the GT.M process run-time
- Stores the offset GT.M must apply to the original block pointers as a consequence of the relocation of the starting VBN
- Upgrades the directory tree (DT) block pointers from 32- to 64-bits; this requires splitting any blocks that do not have sufficient space to accommodate the larger block pointers
- Ensures that all its work is flushed to secondary storage
- Reports completion of its activity on a database file with a "MUPIP MASTERMAP UPGRADE completed" message

At this point, after a successful MUPIP UPGRADE:

- All DT blocks are in V7m format and all GVT index blocks remain in V6/V6p format
- Subsequent activity that updates index blocks for existing GVTs implicitly converts any V6 index blocks to V6p format after applying the offset
- No process other than MUPIP REORG -UPGRADE converts GVT index blocks from V6p format to V7m format; in other words, adding new nodes does not create GVT index blocks with V7 format - adding new nodes splits existing index blocks and such block splits retain the pre-existing block format
- Newly created GVTs, storing new global names, have V7m format
- Data blocks, at level zero (0), and local bit map blocks have the same format in V6 and V7, so, for consistency, normal updates also give those blocks a V7m format designation

These database changes are physical rather than logical, and thus do not require replication beyond noting the increase in transaction numbers.

Phase 2: MUPIP REORG -UPGRADE (GVT Index Block Upgrade)

MUPIP REORG -UPGRADE performs Phase 2 actions of upgrading a database to V7 format. The format of MUPIP REORG -UPGRADE is:

```
MUPIP REORG -UPGRADE {-FILE <file_name> | [-REGION] <region_list>}
```

Before image journaling with MUPIP REORG upgrade provides maximum resiliency. MUPIP REORG -UPGRADE reports it has completed its actions for a region with a MUPGRDSUCC message, at which point all index blocks have V7m format with 64-bit block pointers. You can resume and complete a MUPIP REORG -UPGRADE stopped with a MUPIP STOP (or <Ctrl-C>); avoid a kill -9, which carries a high risk of database damage.

MUPIP REORG -UPGRADE:

- Requires standalone access
- Runs on an entire region; as a result, MUPIP REORG -UPGRADE prevents multiple concurrent REORG -UPGRADE runs per region
- Stops execution when a concurrent Online ROLLBACK is detected because that operation changes the block content of the database
- Can be subject to stopping and restarting at any point
- Processes the GVTs within a database file
 - Splitting any index blocks that do not have sufficient space to accommodate the block pointer upgrade from 32 to 64 bits
 - Updating the block pointers from 32 to 64 bits, also changing the version of the block to V7m

- Journaling its work as before images (if so configured) and INCTN records

Phase 3: Optional GVT Data and Local Bit Map Block Upgrade

While it makes no operational or processing difference, GT.M does not consider the database "fully upgraded" until the block version format of all data blocks becomes V7m. Any of the following operations upgrade some or all of the remaining data blocks:

- MUPIP REORG

Because this operation may not visit every block in the database it may fail to upgrade static/unchanging blocks

- MUPIP REORG -ENCRYPT

- MUPIP INTEG -TN_RESET

This operation requires standalone access and resets the transaction number on all blocks in the database.

Failure to perform Phase 3 has **NO** implications for V7.1-003 but might be an issue for any as-yet unplanned further enhancement.

V7 Upgrade Path 2: EXTRACT and LOAD

Two commonly used mechanisms are as follows. We recommend you use replication to stage the conversion and minimize down time.

- MUPIP EXTRACT -FREEZE followed by a MUPIP LOAD

Using MUPIP EXTRACT with -FREEZE ensures that the V6 database files are frozen at the point of the extract, preventing updates without administrative action to unfreeze the database. MUPIP LOAD the extracts into newly created V7 database files

Use this operation when there is insufficient space to make a database extract

- MERGE command with two global directories and Extended References

Using this approach to transfer data from a V6 database file to a V7 database, administrators must take some action to prevent updates during the transfer

This operation consumes less disk space and disk I/O. As a result the operation is faster than an EXTRACT and LOAD.



If you are using triggers, extract the triggers from the V6 database and load them in the new V7 database.

V7 Upgrade Path 3: No change

Continue using your V6 databases with GT.M V7.1-003. In case you do not wish to operate with files of differing format, specify the -V6 qualifier when invoking MUPIP CREATE.

Choosing the right upgrade path

Choose V7 Upgrade Path 1 or 2 if you anticipate a database file to grow to over 994Mi blocks or require trees of over 7 levels as V7.1-003 supports 16Gi blocks and 11 levels. Note that the maximum size of a V7 database file having 8KiB block size is 114TiB (8KiB*16Gi).

Choose the V7 Upgrade Path 3 if you do not anticipate a database file to grow beyond the V6 database limit of 994Mi blocks or a tree depth limit of 7 levels. Note that the maximum size of a V6 database file having 8KiB block size is 7TiB (8KiB*992Mi).

Other than the new maximum database file size and greater tree depth that comes with V7 Upgrade Path 1 and 2, there is no difference between V7 Upgrade Path 1 and 2 and V7 Upgrade Path 3. You can choose V7 Upgrade Path 3 first and then later choose V7 Upgrade Path 1 or 2 if a need arises.

For additional details on differences in factors involved in the V6 to V7 upgrade refer to Appendix G in the GT.M Administration and Operations Guide.

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 the V5 version of 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 through V6.3-014 have a maximum size of 1,040,187,392 (992Mi) blocks.
- Database created with V7.0-000 and up have a maximum size of 17,179,869,184 (16Gi) blocks.

Stage 3: Replication Instance File Upgrade

GT.M V7.1-003 does not require new replication instance files when upgrading from any version after V6.0-000.

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 e.g. RECOVER, ROLLBACK, or EXTRACT.

MUPIP UPGRADE temporarily disables journaling and replication settings for the duration of its activity. Once complete, MUPIP UPGRADE restores prior settings.

Stage 5: Trigger Definitions Upgrade

GT.M V7.1-003 does not require trigger definition upgrade when upgrading GT.M from any version after V6.3-000. If upgrading from a prior GT.M release, please see the instructions in the release notes for V6.3-014.

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. During installation, GT.M provides an option that allows placing the object code in shared libraries in addition to 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_V7.1-003_i686, then gtmprofile sets \$gtm_dist to /usr/lib/fis-gtm/gtm_V7.1-003_i686/utf8).
 - On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of \$gtmroutines is \$gtm_dist(\$gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

For more information on gtmprofile, 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.

Change History

V7.1-003

Fixes and enhancements specific to V7.1-003:

Id	Prior Id	Category	Summary
GTM-DE201175	GTM-9008	Admin	Source and Receivers Servers start with inptu redirected to / dev/null
GTM-DE376113	-	Language	\$ZTIMEOUT deferred during \$ZINTERRUPT
GTM-DE540134	-	Language	\$STORAGE attempts to be more useful 🟢
GTM-DE551455	-	Admin	🟡 Statsdb excluded from Instance Freeze 🟢
GTM-DE557990	-	Other	GT.M protects certain library calls against asynchronous signal handling
GTM-DE559768	-	Language	GT.M frees memory associated with a pattern alternation when encountering an error in compiling it
GTM-DE563049	-	Other	Prevent inappropriate modification of TLS protocol cipher options
GTM-DE563207	-	Admin	MUPIP REPLICATE -SOURCE -CHECKHEALTH works appropriately with Online Rollback
GTM-DE565976	-	Admin	REORG -MIN_LEVEL=1 correctly traverses the global variable tree
GTM-DE567906	-	Admin	Receiver Server continues to operate after a TLSHANDSHAKE or REPLNOTLS error
GTM-DE567908	-	DB	More appropriate TLS messages
GTM-DE568030	-	Admin	REORG no longer attempts an out-of-bounds read under rare concurrency conditions
GTM-DE568333	-	Admin	Add process ID to certain messages identifying resource contention
GTM-DE568389	-	Admin	🟡 Receiver Server TLS configuration default change 🟢
GTM-F134571	GTM-7883	Language	Improve context in GVSUBOFLOW error message
GTM-F135133	GTM-8912	Language	SOCKET \$PRINCIPAL devices recognize and report SIGHUP signals 🟢
GTM-F228991	-	DB	GT.M can perform more work in parallel when updating a replicated database

Id	Prior Id	Category	Summary
GTM-F235505	GTM-8716	Language	Allow \$ZPEEK / %PEEKBYNAME access to multiple replication instances ✔
GTM-F235586	-	Other	GTMPCAT installation in UTF-8 mode ✔
GTM-F248691	-	Other	✔ TLS configuration and behavior for both replication and SOCKET devices ✔
GTM-F249422	-	Admin	Change EXCEEDRCTLRNDWN in relinkctl_open() to a warning message

Database

- Because TLSv1.3 protocol connections do not implement renegotiation, the Receiver and Source Servers messaging report "renegotiation" and "key-update" for TLSv1.2 and TLSv1.3 connections respectively. Previously the Servers issued messages stating that renegotiation occurred for TLSv1.3 even though TLSv1.3 does not support renegotiation. (GTM-DE567908)
- GT.M performs less work while holding an instance-wide lock during replication. This instance-wide lock controls any increment of the sequence number and prevents any replicated update instance-wide when held. In particular, GT.M processes now calculate the size of any journaling activity and perform the first phase of a database update in BG or the entire update in MM without needing to hold the instance lock. Previously, these operations were done only while holding that lock. A source server process now performs certain replication instance file flushes without holding the same lock where it previously was required. Further, in certain situations where there is no consumer of the updates in the replication journal pool, such as when there are only passive or -JNLFILEONLY source servers, mumps processes do not write each commit into the replication journal pool (they continue to update instance-wide metadata like the sequence number). Previously, processes wrote every commit into the replication journal pool even if there was no consumer for that data. Finally, GT.M gracefully handles certain kill -9 situations that previously led to REPLBRKNTRANS errors. (GTM-F228991)

This page is intentionally left blank.

Language

- GT.M correctly defers executing the \$ZTIMEOUT vector when its timeout coincides with an invocation of \$ZINTERRUPT occurring during a long running command. Previously, due a regression introduced in V7.0-002, \$ZTIMEOUT did not execute its vector in this rare case. The workaround was to minimize the chance of using long running commands when \$ZTIMEOUT expiry and \$ZINTERRUPT processing overlap. This was only seen in development and not reported by a customer. (GTM-DE376113)
- \$STORAGE reports the difference between the current \$ZREALSTOR as a subtrahend and, in order of precedence, \$ZMALLOCLIM if it has a non-zero value, the process R_LIMIT if it is not unlimited or less than the current the end of the process's data segment address, and otherwise the maximum of a 32-bit address space. Previously, \$STORAGE reported a large value that had little objective usefulness. (GTM-DE540134) 🟢
- GT.M frees memory associated with processing a pattern alternation, such as the '2U' in '.3(1N,2U)', when the processing encounters an error. Previously, GT.M failed to free this memory along certain error paths, resulting in a memory leak. (GTM-DE559768)
- GT.M issues a GVSUBOFLOW error message with the length of the exceeded subscript, in certain cases it appends the string "..." indicating the subscript is incomplete. Previously, GT.M issued a less informative GVSUBOFLOW error message. (GTM-F134571)
- USE for a socket device recognizes NOHUPENABLE and HUPENABLE deviceparameters to determine whether to recognize a SIGHUP signal on \$PRINCIPAL as an error. The gtm_hupenable environment variable determines the initial setting for the process. Previously, this feature was only available for terminal devices and a process with a socket \$PRINCIPAL device ignored all SIGHUP signals. (GTM-F135133) 🟢
- When \$ZGBLDIR is set to a global directory specifying a replication instance and replication has started, the Replication Journal Pool for that instance becomes the source of data reported by \$ZPEEK() and thus %PEEKBYNAME(). Those tools report on replication related blocks including GSL (gtmsource_local_struct,) GLF (gtmsrc_lcl,) JPC (jnlpool_ctl_struct,) and RIH (repl_inst_hdr.) If a global directory does not specify a Replication Instance, the gtm_repl_instance environment variable determines the Instance for its replicated regions. Previously, \$ZPEEK() and %PEEKBYNAME() used the most recently accessed Replication Journal Pool which made reliance on its results problematic. (GTM-F235505) 🟢

This page is intentionally left blank.

System Administration

- The Source and Receiver Servers start appropriately when input is redirected from /dev/null. Previously, FIS recommended starting the Source Server with input redirected from /dev/null to workaround GTM-8576. GT.M V6.3-004 included a fix for GTM-8576 that resulted in the workaround causing the Source Server to fail, so the workaround was to not use the obsolete workaround. (GTM-DE201175)
- GT.M creates STATSDB regions with FREEZE_ON_ERROR disabled, preventing errors associated with such regions from causing an instance freeze when the INST_FREEZE_ON_ERROR policy is enabled. Previously, GT.M used the value of FREEZE_ON_ERROR from the associated base region for the STATSDB region. (GTM-DE551455) 🟡🟢
- MUPIP REPLICATE -SOURCE -CHECKHEALTH works appropriately with Online Rollback. Previously -CHECKHEALTH could deadlock with activity involving Online Rollback. This was only seen in development and not reported by a customer. (GTM-DE563207)
- MUPIP REORG -MIN_LEVEL=1 correctly traverses the global variable tree. Previously under rare circumstances REORG could skip processing certain blocks. The workaround is to restart the REORG with the same parameters as the conditions involved are transient. (GTM-DE565976)
- The Receiver Server continues to operate after a TLSHANDSHAKE or REPLNOTLS error. The Source Server operation is unchanged, without plaintext fallback enabled, it terminates for all TLSHANDSHAKE or REPLNOTLS errors. Previously, the Receiver Server terminated when issuing such an error. (GTM-DE567906)
- MUPIP REORG appropriately protects an unusual type of block split from attempting to read out-of-bounds memory due to relying on concurrently changing state. A regression caused by GTM-DE549072 in V7.1-002 introduced this issue, which can cause the REORG process to terminate after a segmentation violation (SIG-11) with no damage to the database. The workaround is to restart the reorg after any such failure. This was only seen in development and not reported by a customer. (GTM-DE568030)
- GT.M messages SPCFCBUFDELAY and WRITERSTUCK include the process ID (PID) of the block resource holder. Note: GT.M WRITERSTUCK messages repeat for each block resource held. Previously, BUFSPCDELAY and WRITERSTUCK did not include the PID of the block resource holder. (GTM-DE568333)
- The Receiver Server defaults SSL_VERIFY_PEER and SSL_VERIFY_FAIL_IF_NO_PEER_CERT if the ID in the TLS configuration file does not specify a verify-mode. Previously, the Receiver server defaulted to SSL_VERIFY_PEER. Customers desiring prior behavior should set verify-mode in the configuration file to just SSL_VERIFY_PEER. (GTM-DE568389) 🟡🟢
- GT.M issues a RLNKCTLOPENDEL warning message to syslog to notify of a stuck process, when a competing process rundowns the relinkctl and another process tries to connect. Previously, GT.M issued an error condition EXCEEDRCTLNDWN message, after 16 failed attempts to connect to the relinkctl. (GTM-F249422)

This page is intentionally left blank.

Other



- GT.M protects certain non-reentrant library calls involved in acquiring and releasing process memory from interrupt processing driven by signals, typically those generated by internal GT.M timers. Previously, GT.M recognized interrupts during these calls, which could in rare cases lead to memory corruption when competing operations coincided. (GTM-DE557990)
- The `$getciphers()` function restores the TLS minimum and maximum protocol versions. Previously, once invoked, all future TLS operations forced TLSv1.3 because the function did not appropriately restore the settings. (GTM-DE563049)
- When run with `gtm_dist` pointing to UTF-8 installation, `install_gtmcat.sh` obtains the necessary UTF-8 settings automatically; the ownership and permissions of the installed files matches the `mumps` executable. Previously, if the appropriate UTF-8 setup had not been done for the installing process, such an installation failed. When the installation succeeded, the ownership and permissions were picked up from the softlink pointing to the `mumps` executable rather than the executable itself. The installed `gtmcat` executable runs in the character set of the underlying environment. Previously, `gtmcat` executable forced itself to run only in M mode. (GTM-F235586) 🟢
- GT.M Database Replication implements Post Handshake Authentication (PHA) when using TLSv1.3. Previously, GT.M Replication ignored the `SSL_VERIFY_POST_HANDSHAKE` verification mode, rejected, or in V7.0-004 through V7.1-001, incorrectly applied it. To negotiate a TLS connection using PHA, the configured TLS ID must specify `SSL_VERIFY_PEER` and `SSL_VERIFY_POST_HANDSHAKE` in the `verify-mode`. Using other `SSL_VERIFY_*` options without `SSL_VERIFY_PEER` results in an error, either `TLSINIT` at server startup or `TLSCONVSOCK` when the Server reads the TLS configuration for its `TLSID`. While GT.M versions V7.0-004 through V7.1-001 do not support `SSL_VERIFY_POST_HANDSHAKE`, they do not reject it. Receiver Servers from V7.0-004 through V7.1-001, when configured with `SSL_VERIFY_POST_HANDSHAKE`, issue `TLSCONNINFO` messages because the Receiver Server fails to request the peer certificate. This happens whether or not the configuration for the Source Server specifies PHA. Receiver Servers configured with `SSL_VERIFY_FAIL_IF_NO_PEER_CERT` fail to connect in this situation.

This change adds two new configuration options to the TLS configuration file: "post-handshake-fallback" and "plaintext-fallback" which each take an integer value. Any positive integer value enables the configuration. GT.M applies these configuration options only to GT.M replication Servers and ignores them for SOCKET devices because these options are part of the replication protocol. Applications using SOCKET devices can imitate either fallback option by checking `$DEVICE` for the TLS connection failure reason and reacting appropriately. Please see the GT.M Programmer's Guide section about ``WRITE /TLS`` for more information.

- "plaintext-fallback" option sets a default action, which can be overridden by the MUIP command line option `-[no]plaintextfallback`, to disallow or allow the servers to fallback to plaintext when they fail to negotiate a TLS connection
- "post-handshake-fallback" enables Receiver Servers configured with `SSL_VERIFY_POST_HANDSHAKE` to temporarily drop `SSL_VERIFY_POST_HANDSHAKE` when

Other

a TLS handshake (TLSHANDSHAKE) with the Source Server fails due to missing support for `SSL_VERIFY_POST_HANDSHAKE`. After the next successful connection, the Receiver Server restores `SSL_VERIFY_POST_HANDSHAKE` for subsequent connections. This allows misconfigured peers, including prior versions, to establish a connection while the Receiver Server periodically issues a warning message.

The OpenSSL project documentation states that client mode connections should only use `SSL_VERIFY_PEER` and no other option. The GT.M reference TLS plugin ignores all verify-mode options except `SSL_VERIFY_PEER` to avoid any unintended operation for client mode connections. GT.M does this to allow sharing of configuration between both client mode TLS processes and server mode TLS processes. (GTM-F248691)  

Error and Other Messages

EXCEEDRCTLNRDWN

Last used version: V7.1-003

EXCEEDRCTLNRDWN, Maximum relinkctl rundown retries limit of nnnn exceeded

MUPIP Error: This indicates competing processes tried to rundown relinkctl more than nnnn times while another process was trying to connect to the relinkctl

Action: Consider a mupip stop to one (or more) of the competing processes

FDSIZELMT

FDSIZELMT, Too many nnnn descriptors needed by GT.CM server

GT.CM Error: A large number (nnnn) of regions accessed on behalf of GT.CM clients forced the file descriptor numerical value to its FD_SETSIZE limit. Under Linux as of this writing, this limit is fixed to 1024.

Action: Review the application, the database layout and the number of concurrent clients and adjust conditions to reduce the number of concurrent database files managed by a GT.CM server.

FORCEDHALT2

FORCEDHALT2, Receipt of 3 MUPIP STOP signals within xxxx seconds, process: yyyy shutting down with no exit handling like a kill -9

Run Time Fatal: This indicates that a GT.M process recognized the receipt of three MUPIP STOP commands within approximately 30 seconds and is shutting down without normal clean up - very similar to a kill -9 signal. This event doesn't stop GT.M processes in an orderly fashion, and might cause database damage if the target process is concurrently actively updating. Therefore use it only on processes that are deadlocked or otherwise stuck, say due to some type of FREEZE.

Action: Determine who initiated the MUPIP STOP and why they did so. Run MUPIP INTEG as appropriate.

GVSUBOFLOW

GVSUBOFLOW, The combined length of subscripts (xxxx) is greater than maximum allowed limit (yyyy) for region: zzzz

Run Time/MUPIP Error: This indicates that a global variable reference specified a total subscript length xxxx that exceeds the maximum length yyyy permitted for region zzzz. The region name zzzz may be

unavailable in some cases. In such instance, use the global name mapping in the Global Directory to find the region. In some instances xxxx may exceed the space available for the error message, in which case it ends with "...".

Action: Ensure that the subscript is correct. If appropriate, adjust the key size for the region with MUPIP SET -REGION -KEY=.

JNLPVTINFO

JNLPVTINFO, Pid aaaa cycle mmmm fd_mismatch nnnn channel rrrr sync_io ssss pini_addr xxxx
qio_active yyyy

Run Time Information: This message always accompanies some other GT.M journaling error message. This gives detailed information on the state of the journal buffers at the time of the accompanying error.

Action: For information purposes only. Review the accompanying message(s) for additional information.

RLNKCTLOPENDEL

RLNKCTLOPENDEL, The relinkctl file rrrr for \$ZROUTINES directory dddd has been deleted by another process, will try to reconnect. (retry = nnnn)

Run Time Warning: Indicates a process failed to connect to the relinkctl file zzzz for \$ZROUTINES directory dddd because it has been deleted by a competing processes.

Action: Ignore when sporadic. If it persists (large nnnn), consider a MUPIP STOP to the process issuing the message and the relinkctl file.

SOCKCLOSE

SOCKCLOSE, Error closing socket: (errno = aaaa) xxxx

Operator log/All GT.M Components Error: aaaa contains the OS error code and xxx indicates information about the error that occurred while closing a socket connection for the process attempting to log a command to the audit logging facility.

Action: Review the message to determine whether the logger programs are running or whether a change is required in the configuration of your audit logging facility.

SOCKHANGUP

Last used version: undefined

SOCKHANGUP, Socket has disconnected

Run Time Error: The process has a \$PRINCIPAL that is a SOCKET device and the process received a SIGHUP signal.

Action: Ensure the application error handling deals with this error by closing down the application gracefully without additional READ or WRITES to \$PRINCIPAL.

SPFCBUFDELAY +

Last used version: undefined

SPFCBUFDELAY, Request for block bbbb in database file dddd delayed by PID iiiii

Run Time Warning: Indicates a process with PID iiiii has control of block bbbb in database file dddd blocking other processes from appropriate access to that block.

Action: Investigate the state and activity of process iiiii to determine why the process is not releasing the block, for instance check whether there is evidence of a deadlock cycle that GT.M was not successful in resolving. If process iiiii is ""stuck"" consider the implications of terminating it, including with a triple MUPIP STOP. Report the results of your investigation to those responsible for ensuring database integrity.

TLSRENEGOTIATE ⚠

TLSRENEGOTIATE, Failed to XXXX TLS/SSL connection

Run Time/MUPIP Error: The XXXX indicates that an attempt to renegotiate or update the keys for the SSL/TLS connection failed.

Action: Review the following TEXT message from the plug-in for additional diagnostic information, and adjust the environment accordingly.

WRITERSTUCK ⚠

WRITERSTUCK, Buffer flush stuck waiting for concurrent writer PID pppp (mmmm of xxxx) to finish writing to database file dddd

Run Time Error: This indicates that GT.M timed out after waiting nearly a minute for process pppp to complete flushing modified global buffers to the disk in database file dddd. This message repeats for each mmmm of xxxx writers.

Action: This is usually symptomatic of a stressed I/O subsystem, where disk writes take a long time. System Administration might be warranted to improve the performance.

This page is intentionally left blank.