



GT.M

Release Notes

V7.1-007

Empowering
the Financial World

FIS

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 ©2025 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	27 March 2025	V7.1-007

Table of Contents

V7.1-007	1
Overview	1
Conventions	1
Platforms	2
Platform support lifecycle	5
GT.M as Open Source Software (OSS)	5
Additional Installation Instructions	6
Recompile	7
Rebuild Shared Libraries or Images	7
Compiling the Reference Implementation Plugin	7
Re-evaluate TLS configuration options	8
Upgrading to V7.1-007	8
Stage 1: Global Directory Upgrade	8
Stage 2: Database Files Upgrade	9
Stage 3: Replication Instance File Upgrade	14
Stage 4: Journal Files Upgrade	14
Stage 5: Trigger Definitions Upgrade	15
Managing M mode and UTF-8 mode	15
Setting the environment variable TERM	16
Installing Compression Libraries	17
Change History	19
V7.1-007	19
Database	21
System Administration	23
Other	25
More Information	27
Additional information for GTM-10955 - GT.M Database Encryption Reference Plugin supports the use of RSA keys to encrypt database keys	27
Configuration File Version 2	27
Plugins Configuration	27
File Encryption	28
TLS	28
Adapting Existing Configurations	28
Command to Convert from GnuPG to OpenSSL	28
Annotated Example Configuration	29
Error and Other Messages	35
JNLNOTALLOWED 	35

This page is intentionally left blank.

V7.1-007

Overview

V7.1-007 Addresses some potentially impactful database issues as well as numerous changes aimed at improving ease of use and a number of more minor fixes. This release is intended for use on more contemporary versions of Linux - please see the Platform section for specifics. The Encryption Reference Plugin demonstrates support for key management using OpenSSL, which seems to perform better when starting larger numbers of processes.

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, message 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	- (dash)
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 enclosed between parentheses () used to track software enhancements and support requests.
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 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>

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>Note: this is the last GT.M release that will support AIX 7.1.</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 9.4; Ubuntu 22.04 LTS and 24.04 LTS; Amazon Linux 2023	<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 V7.1-001 and up are compatible with executable stack restrictions introduced in glibc 2.41</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.</p>

Platform	Supported Versions	Notes
		<p>If you must use NODEFER_ALLOCATE with ext4, you must ensure that your kernel includes commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3). The Red Hat Bugzilla identifier for 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. 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> <div data-bbox="776 1066 857 1150">  </div> <div data-bbox="914 1066 995 1098"> <h3>Note</h3> </div> <div data-bbox="914 1136 1385 1289"> <p>FIS recommends recompiling the reference encryption plugins to match the target platform. See Compiling the Reference Implementation Plugin section for instructions.</p> </div> <div data-bbox="914 1325 1369 1728"> <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</p> </div>

Platform	Supported Versions	Notes
		only affects database replication and not SOCKET devices.



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 based on two years of certified releases. A release becomes no longer officially supported once a given release is more than one release beyond the two year window. Historically we have produced GT.M releases on a quarterly basis, 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 an appropriate 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.

GT.M as Open Source Software (OSS)

FIS maintains and releases GT.M on Linux as OSS. GT.M does not include any OSS libraries.

However, using some GT.M capabilities activates APIs that require the user make some OSS software available:

- Compression: zlib
- Encryption: libconfig and openssl (or equivalent as determined by the encryption plugin); key management is the user's responsibility

- UTF-8 mode: libicuio

while those are what FIS tests with, as long as the API is compatible, substitutions should work.



Note

Linux distributions include various OSS components some of which GT.M relies on.


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 V7.1-007 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V7.1-007_arch (for example, /usr/lib/fis-gtm/V7.1-007_x86_64 on Linux systems). A location such as /opt/fis-gtm/V7.1-007_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 gtmsecshr is not running. If gtmsecshr is running, first stop all GT.M processes including the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP *pid_of_gtmsecshr***.
- Starting with V6.2-000, GT.M no longer supports the use of the deprecated \$gtm_dbkeys and the master key file it points to for database encryption. To convert master files to the libconfig format, please click  to download the CONVDBKEYS.m program and follow instructions in the comments near the top of the program file. You can also download CONVDBKEYS.m from <http://tinco.pair.com/bhaskar/gtm/doc/articles/downloadables/CONVDBKEYS.m>. If you are using \$gtm_dbkeys for database encryption, please convert master key files to libconfig format immediately after upgrading to V6.2-000 or later. Also, modify your environment scripts to include the use of gtmcrypt_config environment variable.

Recompile

- Recompile all M and C source files.

Rebuild Shared Libraries or Images

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

Compiling the Reference Implementation Plugin

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

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

Note that the package names may vary by distribution / version. For example, on RHEL 9 the libraries required to recompile the reference implementation encryption plugin are libgcrypt-devel, gpgme-devel, libconfig-devel, and openssl-devel.

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
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 Administration and Operations 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".

<para/>

Upgrading to V7.1-007



Before you begin

GT.M supports upgrade from V5*, V6.* and V7.* versions to V7.1-007.

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 V7.1-007 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

Before starting, read the upgrade instructions of all stages carefully. Your upgrade procedure for GT.M V7.1-007 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 V7.1-007.
- 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 V7.1-007.
- 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.

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 remove abandoned GT.M database semaphores and release any IPC resources.

There are three upgrade paths available when you upgrade to V7.1-007.

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; requires standalone access
- Phase 2: MUPIP REORG -UPGRADE (GVT Index Block Upgrade); may optionally run with concurrent access if performance is acceptable

Both phases operate once per region. 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.

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 is 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 V7.1-007 but might be an issue for any as-yet unplanned further enhancement.



Important

Taking the steps in the following list that use MUPIP REORG -MIN_LEVEL=1 significantly reduce upgrade time.

The following lists the recommended ordered steps for the full upgrade process:

1. Offline Upgrade instance to use new GT.M V7.1-002+ version - at this point, customers can use the upgraded the GT.M version without any DB changes

2. Online MUPIP SET -INDEX_RESERVED_BYTES=n - where n is 1/3 the block size
3. Online MUPIP REORG -MIN_LEVEL=1 -NOSWAP - free up space in all index blocks to ease the block reference change from 32bits (4bytes) to 64bits (8bytes); this operation alters only index blocks (-MIN_LEVEL=1), and so generates a much lower volume of before image journal records.
4. Offline MUPIP UPGRADE -move blocks around to make space for the expanded master bitmap and upgrade the index blocks in the directory tree (tree of Global names).
5. Online MUPIP REORG -UPGRADE - upgrade the remaining index blocks
6. Online MUPIP SET -INDEX_RESERVED_BYTES=0 - remove the previously applied reservation as it is no longer needed; some application may find it produces a continuing performance benefit.
7. (optional) Online REORG -MIN_LEVEL=1 -NOSWAP -NOSPLIT - coalesce the index blocks to leave index blocks in a less fragmented state

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 V7.1-007. 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 V7.1-007 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 17,112,825,856 (~16Gi) blocks.

Stage 3: Replication Instance File Upgrade

GT.M V7.1-007 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), typically by turning journaling OFF and then back ON



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 V7.1-007 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_V7.1-007_i686, then gtmprofile sets \$gtm_dist to /usr/lib/fis-gtm/gtm_V7.1-007_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





V7.1-007

Fixes and enhancements specific to V7.1-007:

Id	Prior Id	Category	Summary
GTM-9041		Admin	MUPIP JOURNAL -EXTRACT -GLOBAL command accurately retrieves and displays TCOM journal records for a specified global
GTM-9676		DB	Update helpers do not create statsdb files when statshare is disabled.
GTM-9792		Admin	For passive source, MUPIP REPLIC -SOURCE -CHANGELOG works, and replication log indicates -STATSLOG={ON OFF} status.
GTM-10039		DB	GT.M issues a NOGTCMDB error when \$ZPEEK() or ^ %PEEKBYNAME references a GT.CM database.
GTM-10651		Admin	MUPIP SET [-file -region -jnlfile] command with no qualifiers issues GTM-E-QUALEXP
GTM-10700		Other	GT.M Replication waits for TLS handshake to complete
GTM-10950		Admin	MUPIP INTEG no longer exits unexpectedly on certain error paths
GTM-10955		DB	GT.M Database Encryption Reference Plugin supports the use of RSA keys to encrypt database keys 🟢
GTM-10969		Admin	LKE SHOW and LKE CLEAR commands do not accept arguments
GTM-11004		DB	GT.M maintains more reliable and accurate GVSTATS
GTM-11011		Admin	MUPIP SET -nodefer_time works
GTM-11015		Admin	GT.M utilities appropriately handle autodeleteable and uncreated autodbs
GTM-11017		Admin	Additional filter status messages in the replication log files and on the operator consol. 🟢
GTM-11033		DB	🔴 Triggers on NOISOLATION globals which perform no updates can no longer cause corruption
GTM-11040		DB	🔴 GT.M correctly handles \$ORDER() operations on globals with subscript-level-mapping

Id	Prior Id	Category	Summary
GTM-11051		Admin	GT.M utilities appropriately reject negative hexadecimal integers.
GTM-11054		Admin	More flexible interpretation of input by DSE EVALUATE 🟢
GTM-11078		DB	🔴 Users can no longer inappropriately modify a ^%YGS stats node directly

Database

- Update helpers do not create statsdb files when statshare is not enabled; previously, update helpers unconditionally created statsdb files regardless of the statshare configuration. (GTM-9676)
- GT.M throws NOGTCMDB error message when the \$ZPEEK() function or the ^%PEEKBYNAME utility references a GT.CM database. Previously, GT.M threw a BADZPEEKRANGE error, which lacked clarity. (GTM-10039)
- The GT.M Reference Encryption Plugin can use RSA keys to encrypt database symmetric keys. The reference plugin leverages OpenSSL(tm) for this functionality. Please see the additional details section for instructions on how to configure RSA keys and how to convert GnuPG encrypted database symmetric keys to RSA encrypted. Customers have reported contemporaneously starting a material number of processes using GnuPG could produce errors and temporary sluggish response. Previously, the Reference Encryption Plugin only supported GnuPG. Also, the GT.M Reference Encryption Plugin now includes a version number in its initialization. This prevents the unintended use of incompatible plugins. Previously the initialization did not include this capability. (GTM-10955) 
- GT.M processes maintain private copies of their gvstats when they explicitly (NOSTATSHARE) or implicitly (failure to open a statsDB) don't maintain stats in a statsDB. Previously, a process that did not maintain shared statsDB statistics did not maintain any GVSTATS. Also, processes maintain shared database statistics accurately; previously some values could be lower than accurate as their maintenance was not protected against concurrency conflicts. (GTM-11004)
- GT.M correctly handles updates to NOISOLATION global variables with triggers installed. Previously, due to a longstanding issue affecting the interaction between NOISOLATION and triggers, triggers which performed no update, or whose last global operation was something other than a SET or KILL, caused improper commit validation for the enclosing SET or KILL operation on the noisolation global. This could lead to corruption in the database and journal before images if there was concurrent activity. (GTM-11033) 
- \$ORDER() correctly handles subscript-level mapping of global variables into different regions in a certain pathological case. Previously, an \$ORDER(1) could skip a map and return an incorrect value when the correct return value of the \$ORDER() would have been the first possible node of a subsequent map, that node had a \$DATA() value of 10, and certain other database layout conditions existed. This issue affected versions of GT.M starting with V7.0-004. Users can work around the issue by ensuring the first possible node of any subscript-level mapping has a \$DATA() value other 10 by ensuring that the top-level subscript has either data or no children. \$ORDER(-1) and \$ZPREVIOUS() are exempt from this issue and may serve as an alternative workaround depending on the application. (GTM-11040) 
- GT.M reliably prevents users from writing ^%YGS nodes into a stats database. Previously, under certain circumstances users could perform a SET or KILL of a ^%YGS node without encountering a PCTYRESERVED error message, potentially modifying the stats nodes of other processes and corrupting stats databases. (GTM-11078) 

This page is intentionally left blank.

System Administration

- MUPIP JOURNAL -EXTRACT -GLOBAL command accurately returns TCOM journal records for the specified global. It correctly prints the transaction block, including the corresponding TSTART records for TCOM entries and the TSTART records for UUPD records. Previously, the command included TCOM records from other regions, which was unintended. Additionally, TCOM records were printed without their corresponding TSTART records, and there were no TSTART records for UUPD records. (GTM-9041)
- MUPIP REPLIC -SOURCE -CHANGELOG functions correctly, while MUPIP REPLIC -SOURCE -STATSLOG={ON|OFF} reports "Begin statistics logging" and "End statistics logging" respectively in the replication log file for passive source server. Previously, attempts to change the log file had no effect, and turning stats on or off did not put any message in the replication log file for passive source server. (GTM-9792)
- MUPIP SET [-file|-region|-jnlfile] command issues GTM-E-QUALEXP error if a command to modify database or journal attribute sdoes not specify qualifiers; previously such a command without any attributes to modify quietly returned with no action. (GTM-10651)
- MUPIP INTEG -REGION no longer exits unexpectedly on certain error paths due to an segmentation violation (SIG-11). The issue, introduced in GTM-10671, causes early process termination and prevents standalone INTEG from cleaning up shared resources it would otherwise remove; this can be remedied with a rundown, and there are no other known consequences. (GTM-10950)
- LKE SHOW and LKE CLEAR commands do not accept arguments; previously LKE SHOW and LKE CLEAR commands accepted but ignored one and two arguments, respectively. (GTM-10969)
- MUPIP set command with -nodefer_time option turns off timed journal writing in MM access mode; previously MUPIP quietly ignored the -nodefer_time option. (GTM-11011)
- MUPIP rollback/recover operations ignore uncreated autodb. Previously, uncreated autodb caused rollback/recover to issue a FILENOTFND error during startup. In addition, MUPIP, DSE, and LKE operations do not delete autodeletable databases, excepting MUPIP daemon processes such as the Source server, Update Process, and helper processes. Previously, due to a defect in GTM-10671, certain MUPIP operations could automatically delete AUTODELETE files during the rundown prior to performing the standalone work. Finally, GT.M treats the file header, not the global directory, as the authoritative source of truth for whether a file is autodeletable. Previously, certain MUPIP commands incorrectly drew this information from the global directory. (GTM-11015)
- GT.M now prints a "Starting filter" message in the replication log file for both the active source and receiver, while the passive source server prints "Enabling filter" in the replication log file when the filter is started. When stopping the filter on the passive source server, GT.M prints "Disabling filters on passive source server" as standard output and "Disabling filter" in the replication log file. Previously, starting the filter did not print any message in the replication log file for the source, receiver, and passive server, while stopping the filter on the passive source server printed "Stop filter initiated" as standard output and did not print any message in the replication log file. (GTM-11017) 🟢

- GT.M utilities reject negative hexadecimal numbers in commands with exception of MUPIP SIZE and DSE EVALUATE; previously, they inappropriately accepted them when processing the command line and treated them as TRUE or only later determined them to be invalid. (GTM-11051)
- When the command provides neither -DECIMAL nor -HEXADECIMAL. DSE EVALUATE treats the number as hexadecimal and performs the conversion and if the number is a valid decimal, DSE performs a second conversion, treating it as decimal. Previously If neither -decimal nor -hexadecimal was provided, DSE treated the number as hexadecimal by default. (GTM-11054) 🟢

Other

- GT.M replication waits for the TLS handshake to complete before proceeding. Previously, GT.M did not wait. This could cause TLS related data to be misinterpreted by replication processing. This was only seen in our development environment and not reported by any clients. (GTM-10700)

This page is intentionally left blank.

More Information

Additional information for GTM-10955 - GT.M Database Encryption Reference Plugin supports the use of RSA keys to encrypt database keys

GT.M ships with a reference implementation of an encryption plugin which uses OpenSSL and GnuPG to perform various operations for Database Encryption, File Encryption, TLS replication and/or TLS-enabled sockets. The environment variable `$gtmencrypt_config` points to the configuration file that GT.M uses to set certain OpenSSL options. This configuration file must be in the structure of a libconfig configuration file. For more information on the structure of a libconfig configuration file, refer to http://hyperrealm.github.io/libconfig/libconfig_manual.html#Configuration-Files.

GT.M Database Encryption, File Encryption, TLS replication, and TLS sockets are not prerequisites for each other. You can use these GT.M features together or selectively in your environment depending on the `$gtmencrypt_config` file for a given process.

The configuration file, pointed to by `$gtmencrypt_config`, is divided into five major sections:

- Configuration Version
- Plugins
- Database Encryption
- File Encryption
- TLS

The `$gtmencrypt_config` file does not need to include all sections. For example, if a GT.M application only uses TLS replication and/or TLS sockets, it does not need to include sections for Database, Files or Plugin.

Configuration File Version 2

The configuration file version does not need to be the first item in the entire configuration. However, placing it at the top of the file makes for simplicity and understanding. Prior to version 2, configuration files were not explicitly versioned. When no version is present, the plugin treats the file version as 1. Configuration files using the "Plugins" section must be version 2 or higher.

A version 2 configuration file is backwards compatible provided keys protected by GnuPG are configured and present on disk. If this section is missing, the plugin treats the configuration as version 1.

Plugins Configuration

Version 2 of the reference plugin configuration supports the "openssl-pkcs8" plugin configuration using OpenSSL. This section, if enabled, defines an RSA private and public key/certificate that encrypt

database encryption keys and file encryption keys. The format of these sections follows the format described for Database Encryption and File Encryption below. Those configurations are nested inside the plugins definition to signify those keys are encrypted with the RSA private key.

The plugin section uses "openssl" in the name because it uses functionality from OpenSSL for encryption.

FIS recommends following your corporate procedures for generating and/or acquiring the RSA key.

File Encryption

The files section defines keys that MUMPS applications use to perform file encryption. It can exist as a global configuration used by GnuPG or as a subsection of the "openssl-pkcs8" plugin configuration.

MUMPS applications reference these keys by their name. These names must be unique among encryption keys inside the configuration file. This section serves to define GnuPG operation in a way compatible with version 1.

TLS

The TLS section contains information such as the location of the root certificate authority, leaf-level certificates with the corresponding private key files, and other TLS configuration options. Under the TLS section, you can use the same tlsid or separate tlsids (with a different set of certificates and options) for TLS replication and TLS sockets depending on what establishes appropriate security for your application.

Adapting Existing Configurations

One can adapt an existing configuration that uses GnuPG protected database and/or file encryption keys to one that uses both an OpenSSL RSA key and GnuPG. This can be useful when sites use more than one version of GT.M. To do so, convert all GnuPG protected symmetric keys to OpenSSL RSA protected symmetric keys. Then add entries for each into the "openssl-pkcs8" section as needed.

Command to Convert from GnuPG to OpenSSL

Each GnuPG protected key can be decrypted and piped to OpenSSL to encrypt with the RSA public key. In the example below, one must input the password for the GnuPG key ring.

```
gpg --homedir=$GNUPGHOME --pinentry-mode=ask --decrypt /path/to/database/key/dbONE.key | \
openssl pkeyutl -certin -inkey /path/to/key/one.crt -out \
/path/to/database/key/pkcs8dbONE.key -pkeyopt rsa_padding_mode:pkcs1 -encrypt
```

Each RSA protected database key file generated like this will need an entry in the "database" subsection of the "openssl-pkcs8" plugin configuration.

Annotated Example Configuration



Note

Each user or role will need a role specific configuration file and keys.

```
/* Default is 1 which disables "plugins" section parsing */
version: 2;

/* Database Encryption Section
 * GT.M processes lookup database symmetric key file paths using both database file paths
 * and the symmetric key hash value in the database file header.
 * To support re-keying, a database file may have multiple keys associated with it.
 * This top-level Database section implicitly provides the (default) GnuPG configuration.
 */
database: {
  keys: (
    { /* Database file ONE */
      dat: "/path/to/database/file/dbONE.dat"; /* Encrypted database file */
      key: "/path/to/database/key/dbONE.key"; /* Encrypted symmetric key */
    },
    { /* Database file TWO */
      dat: "/path/to/database/file/dbTWO.dat";
      key: "/path/to/database/key/dbTWO.key";
    },
    {
      /* Database file ONE using a new key. Note the DB name remains the same */
      dat: "/path/to/database/file/dbONE.dat";
      key: "/path/to/database/key/dbONEnew.key";
    }
  ) /* Append new database symmetric keys to the end of this list */
  ...
);
}

/* File Encryption Section
 * MUMPS applications reference file encryption keys by the name.
 * Each name is unique in the configuration. One name cannot refer to more than one key.
 * This top-level Files section implicitly belongs to the (default) GnuPG plugin
configuration.
 */
files: {
  name1: "/path/to/symmetric/key1";
  name2: "/path/to/symmetric/key2";
};

/* GT.M Encryption Reference Plugin support
 * To use RSA keys for database key encryption, set the configuration version to 2
```

```

    * and create a plugin entry that includes database keys and (if needed) GT.M file
    encryption keys.
    */
plugins: {
    /* When using "openssl-pkcs8", only one RSA key encrypts keys.
    * This could be a corporate issued key+certificate that shares a common root CA.
    * With this setup, users can share database encryption keys.
    * Without the common CA, users fail to verify other users when attempting to share the
    encrypted database symmetric key.
    */
    openssl-pkcs8: {
        enabled: 1; /* Disabled by default */
        format: "PEM"; /* Currently only PEM format is supported */
        key: "/path/to/key/one.key";
        cert: "/path/to/key/one.crt";
        /* The following parameters may be used in a future release.
        * Please refer to the guidance in the TLS section
        */
        CAfile: "/path/to/tls/certs/CA/CA.crt";
        CApath: "/path/to/tls/certs/CA/";
        /* This section follows the guidelines of the top level "Files" section.
        * This plugin specific Files section belongs to this configuration.
        * These keys are encrypted using this plugin's key.
        * Key paths must be distinct from keys used by other plugins
        */
        files: {
            pkcs8name1: "/path/to/symmetric/pkcs8key1";
            pkcs8name2: "/path/to/symmetric/pkcs8key2";
        };
        /* This section follows the guidelines of the top level "Database" section.
        * This plugin specific Database section belongs to this configuration.
        * These keys are encrypted using this plugin's key.
        * Key paths must be distinct from keys used by other plugins
        */
        database: {
            keys: (
            { /* Database file ONE */
                dat: "/path/to/database/file/dbONE.dat";
                key: "/path/to/database/key/pkcs8dbONE.key";
            },
            { /* Database file TWO */
                dat: "/path/to/database/file/dbTWO.dat";
                key: "/path/to/database/key/pkcs8dbTWO.key";
            },
            { /* Database file ONE using a new key. Note the DB name remains the same */
                dat: "/path/to/database/file/dbONE.dat";
                key: "/path/to/database/key/pkcs8dbONEnew.key";
            }
            )
            /* Append new database symmetric keys to the end of this list */
            ...
        );
    };
}

```

```
};
};
};

/* TLS section */
tls: {
/* Certificate Authority (CA) verify depth provides an upper limit on the number of CAs to
look up for verifying a given certificate.
* The depth count is described as ''level 0:peer certificate'', ''level 1: CA
certificate'',
* ''level 2: higher level CA certificate'', and so on.
* The default verification depth is 9.
*/
verify-depth: 7;

/* CAfile: points to a file, in PEM format, describing the trusted CAs. The file can contain
several CA certificates identified by:
* -----BEGIN CERTIFICATE-----
* ... (CA certificate in base64 encoding) ...
* -----END CERTIFICATE-----
* sequences.
*/
CAfile: "/path/to/tls/certs/CA/CA.crt";

/* CApeth: points to a directory containing CA certificates in PEM format.
* The files each contain one CA certificate.
* The plugin looks up files, which must be available, by the CA subject name hash value.
* If more than once certificate with the same name hash value exists, the extension must be
different (e.g. 9d66eef0.0, 9d66eef0.1 etc).
* The directory is typically created by the OpenSSL tool 'c_rehash'.
*/
CApath: "/path/to/tls/certs/CA/";
/* crl: points to a file containing list of revoked certificates. Create this file with the
openssl utility. */
crl: "/path/to/tls/revocation.crl";

/* Timeout (in seconds) for a given session.
* If a connection disconnects and resumes within this time interval,
* TLS reuses the session to speed up the TLS handshake.
* A value of 0 forces sessions to not be reused.
* The default value is 1 hour.
*/
session-timeout: 600;

/* ssl-options: specifies OpenSSL options to be set or cleared. By default the TLS plugin
uses the following ssl-options,
* which disables SSLv2, SSLv3, TLSv1 and TLSv1.1
* ssl-options: "SSL_OP_NO_SSLv2:SSL_OP_NO_SSLv3:SSL_OP_NO_TLSv1:SSL_OP_NO_TLSv1_1"; //
Default
* Enable SSLv3/TLSv1 by doing the following.
* Note the use of '!' preceding the options inverts their application
```

```

* ssl-options: "SSL_OP_NO_SSLv2:!SSL_OP_NO_SSLv3:!SSL_OP_NO_TLSv1:SSL_OP_NO_TLSv1_1";
* WARNING: Even if the configuration is changed to support SSLv3/TLSv1,
* OpenSSL may have been compiled to not support SSLv3/TLSv1
* or additional changes are needed to OpenSSL's configuration to allow SSLv3/TLSv1
*/
/* cipher-list: List of acceptable TLS ciphers
* NOTE: The naming scheme changed as of TLSv1.3
* pre-TLSv1.3 ciphers default if not specified for TLSv1.2 and before cipher-list:
* "ALL:!ADH:!LOW:!EXP:!MD5:!DH:@STRENGTH";
* TLSv1.3 ciphers cipher-list:
*
"TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_8_SHA256:TLS_AES_128_CCM_SHA256:TLS_AES_128_GCM_SHA256"
*/
/* verify-mode: Verification mode
* - SSL_VERIFY_NONE does no verification of certificates
* - SSL_VERIFY_PEER (default) performs verification of all certificates
* - SSL_VERIFY_POST_HANDSHAKE TLSv1.3 specific verifies certificates after handshake
* verify-mode: "SSL_VERIFY_PEER"
*/
/* Enable post-handshake fallback for clients that do not support PHA when using TLSv1.3
* post-handshake-fallback: 1;
*/
/* List of alphanumeric TLS configuration identifiers (the "tlsid" used by GT.M processes)
which must begin with a letter.
* Each configuration consists of name/value pairs for specific options like certificate,
key and format.
* "tlsid" specific configuration options override global "tls" scope options.
* These tlsids can be used for GT.M Database Replication and for TLS SOCKET devices used by
application logic.
*/
PRODUCTION: { /* "PRODUCTION" is the "tlsid" name */
/* Format of the certificate and private key pair.
/* Currently, the GT.M TLS plug-in only supports PEM format. */
format: "PEM";
/* Path to the certificate. */
cert: "/path/to/tls/certs/Malvern.crt";
/* Path to the private key. If the private key is protected by a passphrase, use an
obfuscated version of the password
* in the environment variable which takes the form gtmssl_passwd_<identifier>.
* For instance, for the below key, the environment variable should be
'gtmssl_passwd_PRODUCTION'.
* Currently, the GT.M TLS plug-in only supports RSA private keys.
*/
key: "/path/to/tls/certs/Malvern.key";
};
DEVELOPMENT: { /* "DEVELOPMENT" is the "tlsid" name */
format: "PEM";
cert: "/path/to/tls/certs/BrynMawr.crt";
key: "/path/to/tls/certs/BrynMawr.key";
};
};

```

As this \$gtmconfig file contains database, file, plugin and tls sections, you need to set the gtm_passwd and gtmssl_passwd_<tlsid> environment variables conforming to the maskpass utility's output. For more information on these environment variables, refer to "Environment Variables".

This page is intentionally left blank.

Error and Other Messages

JNLNOTALLOWED ⓘ

JNLNOTALLOWED, Did not enable journaling on xxxx region yyyy

MUPIP Warning: MUPIP SET -JOURNAL was not able to enable journaling on region yyyy due to it being xxxx, either an AUTODB or an AUTODELETE DB,

Action: Journaling is incompatible with AUTODELETE and AUTODB. Adjust the global directory mapping to specify that the database should be -NOAUJTODB -NOAUTODELETE, recreate the database file, and enable journaling.

This page is intentionally left blank.