V6.1-000

GT.M

V6.1-000 Release Notes

Contact Information

GT.M Group
Fidelity Information Services, Inc.
2 West Liberty Boulevard, Suite 300
Malvern, PA 19355
United States of America

GT.M Support for customers: gtmsupport@fisglobal.com
Automated attendant for 24 hour support: +1 (610) 578-4226
Switchboard: +1 (610) 296-8877
Website: http://fis-gtm.com

Legal Notice

Revision History		
Revision 1.2	26 April 2014	• In GTM-2168, added the description of %GBLDEF enhancements for spanning globals.
Revision 1.1	28 January 2014	• In "Additional information for GTM-7743 - GT.M support for TLS encryption of replication streams using a plug-in" [38], added a note about the deprecated gtm_dbkeys environment variable and the downloadable CONVDBKEYS.m program. • Improved the description of GTM-6419.
Revision 1.0	30 December 2013	V6.1-000 - First published revision.

Table of Contents

V6.1-000

Overview

In the major release V6.1-000, GT.M brings virtually unlimited global storage, TLS support for replication, the ability to ZLINK new versions of routines with prior versions on the stack, and more:

- Virtually unlimited global storage: With the ability to map global variables to database regions at the level of subscripts rather than an unsubscripted global variable name, GT.M global variable storage is now limited by factors external to GT.M - available storage, file system limits, memory, and so on. This has additional benefits. For example, if there are ranges of global variables within which there are patterns of sequential access, mapping each range to a different region can take advantage of one of many optimizations in the GT.M database engine for sequential access. See GTM-2168 below.
- Transport Layer Security (TLS/SSL) for replication: With enhancements to the architecture of the encryption plug-in, GT.M can replicate over a secure connection. The reference implementation of a plug-in included with GT.M is tested with OpenSSL. For database encryption, the reference implementation also provides an option to use libgcrypt (from GnuPG) and libcrypto (OpenSSL) in "FIPS mode," removing a need to modify the plugin for sites that require certification for compliance with FIPS 140-2. Note: Achieving FIPS 140-2 certification requires actions and controls well beyond the purview of GT.M, including underlying cryptographic libraries that are certifiably FIPS compliant, administrative controls, and so on. FIS neither provides cryptographic libraries with GT.M nor recommends the use of any specific library - refer to the GT.M Administration and Operations Guide UNIX Edition for more details. See GTM-7743 below.
- Relink recursive: Processes can explicitly ZLINK new versions of routines even when they have prior versions of routines with the same name already associated with an active frame in the stack of the M virtual machine. When a process links a routine with the same name as an existing routine, future calls use the new routine. Prior versions of that routine referenced by the stack remain tied to the stack until they QUIT, at which point they become inaccessible. This enhancement provides a mechanism to patch long-running processes, one that allows a process to retain more state than previous techniques. See GTM-7551 below.

Other new features include:

- SOCKET device support for local sockets (also known as UNIX domain sockets). See GTM-7554 below.
- The ability for a parent process to pass a SOCKET device to a child process in a JOB command. See GTM-7322 below.
- A $ZCLOSE intrinsic special variable that provides the status of a CLOSE of a PIPE device. See GTM-6666 below.
- Faster database reads, especially when large numbers of processes (thousands) concurrently access a database file. See GTM-7865 below.
- Nanosecond timestamps to determine whether a source file is newer than an object file. See GTM-7750 below.
- Performance enhancements to databases on AIX that use the MM access method. See GTM-7756 below.

Support for IPv6 previously released in V6.0-003 as field-test grade functionality is considered production grade functionality in V6.1-000. There are numerous smaller enhancements, performance enhancements, robustness improvements and bug fixes, described below.

Conventions

This document uses the following conventions:

	UNIX	OpenVMS
Flag/Qualifiers	-	/
Program Names or Functions	upper case. For example, MUPIP BACKUP	
Examples	lower case. For example: $char(10) mupip backup -database ACN,HIST /backup	
Reference Number	A reference number is used to track software $char(10) enhancements and support requests. $char(10) It is enclosed between parentheses ().	
Platform Identifier	[UNIX]	[OpenVMS]
	The platform identifier does not appear $char(10) for those new features or $char(10) enhancements that apply to all platforms.	

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, HP-UX, GNU/Linux, and Solaris.

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

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
originating instance or primary instance	-rootprimary	originating instance or originating primary instance. Within the context of a replication connection between two instances, an originating instance is referred to as source instance or source side. For example, in an B<-A->C replication configuration, A is the source instance for B and C.	-updok (recommended) -rootprimary (still accepted)
replicating instance (or secondary instance) and propagating instance	N/A for replicating instance or secondary instance. -propagateprimary for propagating instance	replicating instance. Within the context of a replication connection between two instances, a replicating instance that receives updates from a source instance is referred to as receiving instance or receiver side. For example, in an B<-A->C replication configuration, both B and C can be referred to as a receiving instance.	-updnotok
N/A	N/A	supplementary instance. For example, in an A->P->Q replication configuration, P is the supplementary	-updok

Pre V5.5-000 terminology	Pre V5.5-000 qualifier	Current terminology	Current qualifiers
		instance. Both A and P are originating instances.	

Effective V6.0-000, GT.M documentation is adopting IEC standard Prefixes for binary multiples. This document therefore uses prefixes Ki, Mi and Ti (e.g., 1MiB for 1,048,576 bytes). All GT.M documentation will over time be updated to this standard.

denotes a new feature that requires updating the manuals.

denotes a new feature or an enhancement that may not be upward compatible and may affect an existing application.

denotes deprecated messages.

denotes revised messages.

denotes added messages.

Platforms

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

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

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

Platform	Supported Versions	Notes
Hewlett-Packard Integrity IA64 HP-UX	11V3 (11.31)	-
Hewlett-Packard Alpha/AXP OpenVMS	7.3-2 / 8.2 / 8.3	GT.M supports M mode but not UTF-8 mode on this platform. GT.M does not support several recent enhancements on this platform, including but not limited to database encryption, on-line backup, multi-site replication, PIPE devices and triggers. If you need to work with external calls written in C with Version 6.x of the C compiler on Alpha OpenVMS, then you must carefully review all the provided kits for that product and apply them appropriately.

Platform	Supported Versions	Notes
		Although this platform remains at present fully supported with respect to bug fixes, owing to its looming sunset by HP, new functionality is supported on this platform only for FIS' convenience. Future GT.M releases may not be supported on this platform. Regardless of ongoing plans for support of the OpenVMS platform itself, the next GT.M release will likely no longer support OpenVMS 7.x. Please contact your FIS account manager if you need ongoing support for GT.M on this platform.
IBM System p AIX	6.1, 7.1	Since GT.M processes are 64-bit, FIS expects 64-bit AIX configurations to be preferable. 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. 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.**
Sun SPARC Solaris	10 (Update 6 and above)	The deprecated DAL calls operate in M mode but not in UTF-8 mode. Please refer to the Integrating External Routines chapter in the Programmer's Guide for appropriate alternatives.
x86_64 GNU/Linux	Red Hat Enterprise Linux 6; Ubuntu 12.04 LTS; SuSE Linux Enterprise Server 11	To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware. GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.5-24 or later) and ncurses (version 5.5 or later). To install GT.M with Unicode (UTF-8) support on RHEL 6, in response to the installation question **Should an ICU version other than the default be used? (y or n)** please respond **y** and then specify the ICU version (for example, respond 3.6) to the subsequent prompt **Enter ICU version (ICU version 3.6 or later required. Enter as major-ver.minor-ver):** GT.M requires the libtinfo library. If it is not already installed on your system, and is available using the package manager, install it using the package manager. If a libtinfo package is not available (for example on SuSE 11): • Find the directory where libncurses.so is installed on your system. • Change to that directory and make a symbolic link to libncurses.so.<ver> from libtinfo.so.<ver>. Note that some of the libncurses.so entries may themselves be symbolic links, for example, libncurses.so.5 may itself be a symbolic link to libncurses.so.5.9. GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6 or later), glibc (version 2.5-49 or later) and ncurses (version 5.5-24 or later). The minimum CPU must have the instruction set of a 686 (Pentium Pro) or equivalent.

Platform	Supported Versions	Notes
x86 GNU/Linux	Red Hat Enterprise Linux 6	This 32-bit version of GT.M runs on either 32- or 64-bit x86 platforms; we expect the X86_64 GNU/Linux version of GT.M to be preferable on 64-bit hardware.

Platform support lifecycle

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

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

Migrating to 64-bit platforms

The same application code runs on both 32-bit and 64-bit platforms. Please note that:

- You must compile the application code separately for each platform. Even though the M source code is exactly the same, the generated object modules are different even on the same hardware architecture - the object code differs between x86 and x86_64.
- Parameter-types that interface GT.M with non-M code using C calling conventions must match the data-types on their target platforms. Mostly, these parameters are for call-ins, external calls, internationalization (collation), and environment translation and are listed in the tables below. Note that most addresses on 64-bit platforms are 8 bytes long and require 8 byte alignment in structures whereas all addresses on 32-bit platforms are 4 bytes long and require 4-byte alignment in structures.

Call-ins and External Calls

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

Caution

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

Internationalization (Collation)

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

Important

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

Environment Translation

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

Important

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

Recompile

- Recompile all M and C source files.

Rebuild Shared Libraries or Images

- Rebuild all Shared Libraries (UNIX) or Shareable Executable Images (OpenVMS) after recompiling all M and C source files..

Additional Installation Instructions

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

UNIX

- FIS strongly recommends installing each version of GT.M in a separate (new) directory, rather than overwriting a previously installed version. If you have a legitimate need to overwrite an existing GT.M installation with a new version, you must first shut down all processes using the old version. FIS suggests installing GT.M V6.1-000 in a Filesystem Hierarchy Standard compliant location such as /usr/lib/fis-gtm/V6.1-000_arch (for example, /usr/lib/fis-gtm/V6.1-000_x86 on 32-bit Linux

systems). A location such as /opt/fis-gtm/V6.1-000_arch would also be appropriate. Note that the ***arch*** suffix is especially important if you plan to install 32- and 64-bit versions of the same release of GT.M on the same system.

- Use the MUPIP RUNDOWN command of the old GT.M version to ensure all database files are cleanly closed.
- In UNIX editions, 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***.

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).

Additional Information for JFS1 on AIX

If you expect a database file or journal file to exceed 2GB with older versions of the JFS file system, then you must configure its file system to permit files larger than 2GB. Furthermore, should you choose to place journal files on file systems with a 2GB limit, since GT.M journal files can grow to a maximum size of 4GB, you must then set the journal auto switch limit to less than 2 GB.

OpenVMS

To upgrade from a GT.M version prior to V4.3-001, you must update any customized copy of **GTM$DEFAULTS** to include a definition for **GTM$ZDATE_FORM**.

You can ignore the following section if you choose the standard GT.M configuration or answer yes to the following question:

```
Do you want to define GT.M commands to the system
```

If you define GT.M commands locally with **SET COMMAND GTM$DIST:GTMCOMMANDS.CLD** in **GTMLOGIN.COM** or other command file for each process which uses GT.M, you must execute the same command after installing the new version of GT.M and before using it. If you define the GT.M commands to the system other than during the installation of GT.M, you must update the system DCLTABLES with the new GTMCOMMANDS.CLD provided with this version of GT.M. See the OpenVMS "Command Definition, Librarian, and Message Utilities Manual" section on "Adding a system command." In both cases, all GT.M processes must match the proper **GTMCOMMANDS.CLD** with the version of GT.M they run..

Upgrading to GT.M V6.1-000

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

GT.M upgrade procedure for V6.1-000 consists of 5 stages:

- Stage 1: Global Directory Upgrade
- Stage 2: Database Files Upgrade
- Stage 3: Replication Instance File Upgrade

- Stage 4: Journal Files Upgrade
- Stage 5: Trigger Definitions Upgrade

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V6.1-000 depends on your GT.M upgrade history and your current version.

Stage 1: Global Directory Upgrade

FIS strongly recommends you back up your Global Directory before upgrading. There is no single-step method for downgrading a Global Directory to an older format.

To upgrade from any previous version of GT.M:

- Open your Global Directory with the GDE utility program of GT.M V6.1-000.
- Execute the EXIT command. This command automatically upgrades the Global Directory.

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

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

An analogous procedure applies in the reverse direction.

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

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

- Open the global directory with the GDE utility program of V6.1-000.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

Stage 2: Database Files Upgrade

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

A V6 database file is a superset of a V5 database file and has potentially longer keys and records. Therefore, upgrading a database file requires no explicit procedure. After upgrading the Global Directory, opening a V5 database with a V6 process automatically upgrades fields in the database fileheader.

A V6 database supports global variable nodes up to 1 MiB and is not backward compatible. Use MUPIP DOWNGRADE -VERSION=V5 to downgrade a V6 database back to V5 format provided it meets the database downgrade requirements. For more information on downgrading a database, refer to Downgrading to V5 or V4.

Important

A V5 database that has been automatically upgraded to V6 can perform all GT.M V6.1-000 operations. However, that database can only grow to the maximum size of the version in which it was originally created. If the database is V5.0-000 through V5.3-003, the maximum size is 128Mi blocks. If the database is V5.4-000 through V5.5-000, the maximum size is 224Mi blocks. Only a database created with V6.0-000 or above (with a V6 MUPIP CREATE) can have a maximum database size of 992Mi blocks.

If your database has any previously used but free blocks from an earlier upgrade cycle (V4 to V5), you may need to execute the MUPIP REORG -UPGRADE command. If you have already executed the MUPIP REORG -UPGRADE command in a version prior to V5.3-003 and if subsequent versions cannot determine whether MUPIP REORG -UPGRADE performed all required actions, it sends warnings to the operator log requesting another run of MUPIP REORG -UPGRADE. In that case, perform any one of the following steps:

- Execute the MUPIP REORG -UPGRADE command again, or
- Execute the DSE CHANGE -FILEHEADER -FULLY_UPGRADED=1 command to stop the warnings.

Caution

Do not run the DSE CHANGE -FILEHEADER -FULLY_UPGRADED=1 command unless you are absolutely sure of having previously run a MUPIP REORG -UPGRADE from V5.3-003 or later. An inappropriate DSE CHANGE -FILEHEADE -FULLY_UPGRADED=1 may lead to database integrity issues.

You do not need to run MUPIP REORG -UPGRADE on:

- A database that was created by a V5 MUPIP CREATE
- A database that has been completely processed by a MUPIP REORG -UPGRADE from V5.3-003 or later.

For additional upgrade considerations, refer to Database Compatibility Notes.

To upgrade from a GT.M version prior to V5.000:

You need to upgrade your database files only when there is a block format upgrade from V4 to V5. However, some versions, for example, database files which have been initially been created with V4 (and subsequently upgraded to a V5 format) may additionally need a MUPIP REORG -UPGRADE operation to upgrade previously used but free blocks that may have been missed by earlier upgrade tools.

- Upgrade your database files using in-place or traditional database upgrade procedure depending on your situation. For more information on in-place/traditional database upgrade, see Database Migration Technical Bulletin.
- Run the MUPIP REORG -UPGRADE command. This command upgrades all V4 blocks to V5 format.

Note

Databases created with GT.M releases prior to V5.0-000 and upgraded to a V5 format retain the maximum size limit of 64Mi (67,108,864) blocks.

Database Compatibility Notes

- Changes to the database file header may occur in any release. GT.M automatically upgrades database file headers as needed. Any changes to database file headers are upward and downward compatible within a major database release number, that is, although processes from only one GT.M release can access a database file at any given time, processes running different GT.M releases with the same major release number can access a database file at different times.

- Databases created with V5.3-004 through V5.5-000 can grow to a maximum size of 224Mi (234,881,024) blocks. This means, for example, that with an 8KiB block size, the maximum database file size is 1,792GiB; this is effectively the size of a single global variable that has a region to itself; a database consists of any number of global variables. A database created with GT.M versions V5.0-000 through V5.3-003 can be upgraded with MUPIP UPGRADE to increase the limit on database file size from 128Mi to 224Mi blocks.

- Databases created with V5.0-000 through V5.3-003 have a maximum size of 128Mi (134, 217,728) blocks. GT.M versions V5.0-000 through V5.3-003 can access databases created with V5.3-004 and later as long as they remain within a 128Mi block limit.

- Database created with V6.0-000 or above have a maximum size of 1,040,187,392(992Mi) blocks.

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

Stage 3: Replication Instance File Upgrade

V6.1-000 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.1-000 requires new replication instance files if you are upgrading from any version prior to V5.5-000. Instructions for creating new replication instance files are in the Database Replication chapter of the UNIX Administration and Operations Guide. Shut down all Receiver Servers on other instances that are to receive updates from this instance, shut down this instance Source Server(s), recreate the instance file, restart the Source Server(s) and then restart any Receiver Server for this instance with the -UPDATERESYNC qualifier.

Note

Without the UPDATERESYNC qualifier, the replicating instance synchronizes with the originating instance using state information from both instances and potentially rolling back information on the replicating instance. The UPDATERESYNC qualifier declares the replicating instance to be in a wholesome state matching some prior (or current) state of the originating instance; it causes MUPIP to update the information in the replication instance file of the originating instance and not modify information currently in the database on the replicating instance. After this command, the replicating instance catches up to the originating instance starting from its own current state. Use UPDATERESYNC only when you are absolutely certain that the replicating instance database was shut down normally with no errors, or appropriately copied from another instance with no errors.

Important

You must always follow the steps described in the Database Replication chapter of the UNIX Administration and Operations Guide when migrating from a logical dual site (LDS) configuration to an LMS configuration, even if you are not changing GT.M releases.

Stage 4: Journal Files Upgrade

On every GT.M upgrade:

- Create a fresh backup of your database.
- Generate new journal files (without back-links).

Important

This is necessary because MUPIP JOURNAL cannot use journal files from a release other than its own for RECOVER, ROLLBACK, or EXTRACT.

Stage 5: Trigger Definitions Upgrade

If you are upgrading from V5.4-002A/V5.4-002B/V5.5-000 to V6.1-000, you do not need to extract and reload triggers.

You need to extract and reload your trigger definitions only if you are upgrading from V5.4-000/V5.4-000A/V5.4-001 to V6.1-000. This is necessary because multi-line XECUTEs for triggers require a different internal storage format for triggers which makes triggers created in V5.4-000/V5.4-000A/V5.4-001 incompatible with V5.4-002/V5.4-002A/V5.4-002B/V5.5-000/V6.0-000/V6.0-001/V6.1-000.

To extract and reapply the trigger definitions on V6.1-000 using MUPIP TRIGGER:

1. Using the old version, execute a command like **mupip trigger -select="*" trigger_defs.trg**. Now, the output file trigger_defs.trg contains all trigger definitions.
2. Place -* at the beginning of the trigger_defs.trg file to remove the old trigger definitions.
3. Using V6.1-000, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.1-000 replicating instance using $ZTRIGGER():

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

Note

Reloading triggers renumbers automatically generated trigger names.

Downgrading to V5 or V4

You can downgrade a GT.M V6 database to V5 or V4 format using MUPIP DOWNGRADE.

Starting with V6.0-000, MUPIP DOWNGRADE supports the -VERSION qualifier with the following format:

```
MUPIP DOWNGRADE -VERSION=[V5|V4]
```

-VERSION specifies the desired version for the database header.

To qualify for a downgrade from V6 to V5, your database must meet the following requirements:

1. The database was created with a major version no greater than the target version.
2. The database does not contain any records that exceed the block size (spanning nodes).
3. The sizes of all the keys in database are less than 256 bytes.
4. There are no keys present in database with size greater than the Maximum-Key-Size specification the database header, that is, Maximum-Key-Size is assured.
5. The maximum Record size is small enough to accommodate key, overhead, and value within a block.

If your database meets all the above requirements, MUPIP DOWNGRADE -VERSION=V5 resets the database header to V5 elements which makes it compatible with V5 versions.

To qualify for a downgrade from V6 to V4, your database must meet the same downgrade requirements that are there for downgrading from V6 to V5.

If your database meets the downgrade requirements, perform the following steps to downgrade to V4:

1. In a GT.M V6.1-000 environment:
   a. Execute MUPIP SET -VERSION=v4 so that GT.M writes updates blocks in V4 format.
   b. Execute MUPIP REORG -DOWNGRADE to convert all blocks from V6 format to V4 format.
2. Bring down all V6 GT.M processes and execute MUPIP RUNDOWN -FILE on each database file to ensure that there are no processes accessing the database files.
3. Execute MUPIP DOWNGRADE -VERSION=V4 to change the database file header from V6 to V4.
4. Restore or recreate all the V4 global directory files.
5. Your database is now successfully downgraded to V4.

Managing M mode and UTF-8 mode

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

On a system that has ICU installed, GT.M optionally installs support for both M mode and UTF-8 mode, including a utf8 subdirectory of the directory where GT.M is installed. From the same source file, depending upon the value of the environment variable gtm_chset, the GT.M compiler generates an object file either for M mode or UTF-8 mode. GT.M generates a new object file when it finds both a source and an object file, and the object predates the source file and was generated with the same

setting of $gtm_chset/$ZCHset. A GT.M process generates an error if it encounters an object file generated with a different setting of $gtm_chset/$ZCHset than that processes' current value.

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

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

- Actual files for GT.M executable programs (mumps, mupip, dse, lke, and so on) are in the parent directory, that is, the location specified for installation.
- Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for Unicode in the utf8 subdirectory, and one compiled without support for Unicode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install Unicode support.
- The utf8 subdirectory has files called mumps, mupip, dse, lke, and so on, which are relative symbolic links to the executables in the parent directory (for example, mumps is the symbolic link ../mumps).
- When a shell process sources the file gtmprofile, the behavior is as follows:
  - If $gtm_chset is "m", "M" or undefined, there is no change from the previous GT.M versions to the value of the environment variable $gtmroutines.
  - If $gtm_chset is "UTF-8" (the check is case-insensitive),
    - $gtm_dist is set to the utf8 subdirectory (that is, if GT.M is installed in /usr/lib/fis-gtm/gtm_V6.1-000_i686, then gtmprofile and gtmcshrc set $gtm_dist to /usr/lib/fis-gtm/gtm_V6.1-000_i686/utf8).
    - On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

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

Compiling ICU

GT.M versions prior to V5.3-004 require exactly ICU 3.6, however, V5.3-004 (or later) accept ICU 3.6 or later. For sample instructions to download ICU, configure it not to use multi-threading, and compile it for various platforms, refer to Appendix C: Compiling ICU on GT.M supported platforms of the UNIX Administration and Operations Guide.

Although GT.M uses ICU, ICU is not FIS software and FIS does not support ICU. The sample instructions for installing and configuring ICU are merely provided as a convenience to you.

Also, note that download sites, versions of compilers, and milli and micro releases of ICU may have changed since the dates when these instructions were tested rendering them out-of-date. Therefore, these instructions must be considered examples, not a cookbook.

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(smkx), lines(lines).
```

GT.M sends keypad_xmit before terminal reads for direct mode and READs (other than READ *) if EDITING is enabled. GT.M sends keypad_local after these terminal reads.

Installing Compression Libraries

If you plan to use the optional compression facility for replication, you must provide the compression library. The GT.M interface for compression libraries accepts the zlib compression libraries without any need for adaptation. These libraries are included in many UNIX distributions and are downloadable from the zlib home page. If you prefer to use other compression libraries, you need to configure or adapt them to provide the same API provided by zlib. Simple instructions for compiling zlib on a number of platforms follow. Although GT.M can use zlib, zlib is not FIS software and FIS does not support zlib. These instructions are merely provided as a convenience to you.

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

Solaris/cc compiler from Sun Studio:

```
./configure --sharedmake CFLAGS="-KPIC -m64"
```

HP-UX(IA64)/HP C compiler:

```
./configure --sharedmake CFLAGS="+DD64"
```

AIX/XL compiler:

```
./configure --sharedAdd -q64 to the LDFLAGS line of the Makefilemake CFLAGS="-q64"
```

Linux/gcc:

```
./configure --sharedmake CFLAGS="-m64"
```

By default, GT.M searches for the libz.so shared library (libz.sl on HPUX PA-RISC) 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 (other UNIX platforms) includes the directory containing the library. The Source and Receiver Server link the shared library at runtime. If this fails for

any reason (such as file not found, or insufficient authorization), the replication logic logs a DLLNOOPEN error and continues with no compression.

Change History

V6.1-000

Fixes and enhancements specific to V6.1-000 are:

Id	Prior Id	Category	Summary
GTM-2040	C9903-000932	MUMPS	Socket Device LISTEN, TCP Device Removed
GTM-2168	C9904-001026	Other Utilities	Global variables span multiple database regions
GTM-3731	S9B06-001882	MUMPS	SOCKET device USE and WRITE /WAIT improvements, and a new $ZKEY ISV
GTM-4797	C9D01-002214	MUMPS	Socket LISTEN Improvements
GTM-4800	C9D01-002217	MUMPS	SOCKET device [Z]LISTEN= fix to handle a bad argument
GTM-5572	C9E11-002662	Other Utilities	GDE @<file> does not accept spaces in the file name
GTM-5573	C9E11-002663	DB	MERGE deals with differing collation between source and target
GTM-5576	C9E11-002666	DB	In UNIX, give appropriate error for absent or incorrect specification of collation library
GTM-6419	D9J07-002733	MUMPS	In UNIX, GT.M allows arguments present in a function definition in the C/Java external call table to be missing from a corresponding invocation
GTM-6666	C9K08-003309	MUMPS	In UNIX, $ZCLOSE returns status of last process associated with a CLOSE'd PIPE device
GTM-6867	C9L08-003446	MUMPS	A DETACH operation on a socket now succeeds the first time it is done in a process
GTM-7074	-	MUPIP	MUPIP INTEG now generates detailed error messages for bitmap blocks.
GTM-7279	-	DB	In UNIX, GT.M defers the processing of signals while flushing dirty database buffers
GTM-7322	-	MUMPS	In UNIX, Passing Sockets via JOB Command
GTM-7453	-	MUPIP	In UNIX, journal backward recovery/rollback avoids incorrect JNLDBTNNOMATCH errors
GTM-7551	-	MUMPS	Recursive ZLINK for those UNIX platforms that support shared libraries
GTM-7554	-	MUMPS	Support for LOCAL (also known as UNIX domain) sockets

Change History

Id	Prior Id	Category	Summary
GTM-7562	-	Other Utilities	DSE remains in the same region after a DUMP -BLOCK operation where records in the dumped block contained global names mapping to a different region.
GTM-7600	-	DB	In UNIX, GT.M handles the concurrency among GT.M processes performing global buffer flushing and buffer pool recovery correctly
GTM-7603	-	MUMPS	Fix for processes with $PRINCIPAL configured for input from file and output to socket
GTM-7626	-	MUPIP	In UNIX, MUPIP issues CRITSEMFAIL only on non-transient problems
GTM-7645	-	DB	In UNIX, Instance Freeze on out-of-space configurable
GTM-7649	-	DB	In UNIX, REPLINSTNOSHM instead of REPLINSTMISMTCH if journal pool is unreachable
GTM-7693	-	Other Utilities	GDE respects comment delimiter under nearly all circumstances
GTM-7709	-	MUMPS	UTF-8 mode sequential READ with FIXED and FOLLOW has improved handling of INTRPT and maintenance of $X/$Y
GTM-7719	-	MUMPS	In UNIX, if GT.M detects an error in using a system service to start or stop a system-maintained timer, it sends a SETITIMERFAILED diagnostic message to the operator log
GTM-7739	-	MUMPS	A SOCKET OPEN or USE with the LISTEN deviceparameter properly handles errors while assigning a name to the socket
GTM-7742	-	Other Utilities	GDE provides -MUTEX_SLOTS for database segments
GTM-7743	-	MUPIP	In UNIX, TLS encryption plug-in for replication
GTM-7750	-	MUMPS	In UNIX, ZLINK decides whether to recompile based nanosecond-level timestamp comparison between the source and object file timestamps
GTM-7756	-	DB	On AIX, correct update serialization in MM mode and improve throughput in MM mode databases
GTM-7763	-	MUMPS	UTF-8 mode sequential disks have improved BADCHAR detection and timeout, and, with FOLLOW, better timeout duration
GTM-7769	-	MUPIP	MUPIP JOURNAL -EXTRACT -GLOBAL= fix for embedded quotes within subscripts
GTM-7773	-	MUMPS	SOCKET WAIT with timeout sets $TEST
GTM-7804	-	DB	Improved database critical section management

Change History

Id	Prior Id	Category	Summary
GTM-7810	-	DB	Various corrections to odd cases involving extended references
GTM-7811	-	DB	In UNIX, fix in V6.0-003 for MUPIP STOP arriving at a TRESTART involving a trigger
GTM-7820	-	MUPIP	In UNIX, MUPIP REORG -TRUNCATE -SELECT= eliminates unused space more efficiently
GTM-7825	-	DB	In UNIX, TRIGNAMEUNIQUE reports the correct number of unique run-time trigger names
GTM-7827	-	MUPIP	In UNIX, MUPIP EXTRACT in BINARY format creates valid extract files in all cases
GTM-7835	-	MUMPS	SOCKET device connect fix
GTM-7836	-	MUMPS	Sorts-after (]]) operator correction for interaction between NOUNDEF operation and some cases of NEW
GTM-7842	-	MUMPS	In UNIX, ZGOTO 0:<entryref> clears all variables including those previously NEW'd at level 1
GTM-7845	-	MUPIP	In UNIX, MUPIP RUNDOWN does not remove semaphores inappropriately
GTM-7846	-	DB	Improved permissions handling for UNIX root user
GTM-7848	-	Other Utilities	In UNIX, DSE and LKE are better at retrying database open
GTM-7853	-	MUPIP	MUPIP INTEG -SUBSCRIPTS works correctly if the -SUBSCRIPTS qualifier argument contains a null subscript
GTM-7854	-	DB	Replication filters apply global nodes containing null subscripts correctly on database regions having Standard Null Collation enabled
GTM-7856	-	DB	Replication filters apply global nodes containing null subscripts correctly when the source and receiver sides have different null collation settings
GTM-7858	-	Other Utilities	In UNIX, eliminate arbitrary GTMSECSHRVFID warnings
GTM-7865	-	DB	Non-TP read optimization
GTM-7866	-	Other Utilities	GTMJI allows assigning ISVs to M gtm_string_t-type input/output arguments in call-in invocations
GTM-7867	-	DB	MERGE issues GVSUBOFLOW and MAXNRSUBSCRIPTS error as appropriate in case the target of the merge is a global node.
GTM-7886	-	MUMPS	On AIX, better diagnostic for ICU library not loading

Change History

Id	Prior Id	Category	Summary
GTM-7890	-	MUPIP	Fix for potential SIG-11 when flow-control occurs at the receive pool boundary
GTM-7892	-	MUPIP	In UNIX, Replication servers timeout if an external filter does not respond
GTM-7896	-	MUMPS	SET $ZPIECE() allows invalid UTF-8 strings (such as $ZCHAR(255)) as piece separator, and BADCHAR error message improvements
GTM-7902	-	DB	In UNIX, calmer Instance Freeze During File Extension
GTM-7905	-	MUMPS	Fix SIG-11 from a JOB command within a shared library after VIEW "JOBPID"
GTM-7908	-	MUMPS	In UNIX, prevent JOB command hang when child process fails to start or terminates abnormally
GTM-7910	-	MUPIP	Eliminate rare access violation in deadlock check
GTM-7916	-	MUPIP	In UNIX, MUPIP ONLINE ROLLBACK fix for an odd case
GTM-7920	-	DB	Fix $INCREMENT() of a just KILL'd spanning node
GTM-7923	-	MUMPS	Fix to prevent very long patterns from terminating the process

M-Database Access

- MERGE supports copying a global into another global with different collation properties (alternative collation, numeric collation, null collation). Previously, in this circumstance, MERGE inappropriately stored the source global subscripts into the destination global without doing the needed subscript transformations.(GTM-5573)
- Specifying collation with a missing or improperly specified collation library gives an appropriate error. Previously this resulted in a segmentation violation (SIG11). [UNIX] (GTM-5576)
- GT.M defers the processing of signals, such as SIGTERM, while flushing dirty database buffers. Previously, a signal delivered to a GT.M process within a small time window during the buffer-flushing operation could leave shared resources in a state that delayed other GT.M process from making further progress for up to several minutes. FIS encountered this issue in testing and has not received a customer report of such an event. [UNIX] (GTM-7279)
- GT.M handles concurrency among GT.M processes performing global buffer flushing and buffer pool recovery correctly. Previously, such concurrency could lead to out of design situation causing a GTMASSERT error. This problem caused a process loss, not database damage. Note that buffer pool recovery is an unusual event so such a concurrent event is quite unlikely. [UNIX](GTM-7600)
- GT.M checks if DSKNOSPCAVAIL is configured in the custom errors file when handling out-of-space conditions. Previously, if any custom errors file was configured, GT.M always behaved as if DSKNOSPCAVAIL was in the custom errors, and indefinitely retried out-of-space conditions. [UNIX] (GTM-7645)
- A replicated database access can fail with REPLINSTNOSHM instead of REPLINSTMISMTCH if the journal pool is unreachable. Previously, such a database access could fail with REPLINSTMISMTCH if the journal pool id was missing or invalid in the replication instance file. In this case, the REPLINSTMISMTCH error message displayed the instance file name as an empty string, and journal pool shared memory id as 4294967295. [UNIX] (GTM-7649)
- GTM performs correct update serialization in MM mode on multi-CPU systems. Previously, GT.M did not protect against out-of-order memory accesses by some pSeries model hardware that occasionally produced inconsistent database states. This condition did not affect BG mode databases, appeared only in the GT.M development and test environment, and has not been reported by a customer. This change also improves throughput on MM mode databases. [AIX] (GTM-7756)
- GT.M and its utilities more cleanly maintain the mechanism used to manage database critical sections. Previously a terminating process could inappropriately decrease the number of available entries in a queuing structure within the critical section logic and eventually cause less efficient critical section management. Also, MUPIP SET -MUTEX_SLOTS can be set between 64 and 32768. Previously, the lower limit was 1024. The default value of 1024 should be appropriate for most situations and FIS recommends changing it only in consultation with FIS GT.M Support. (GTM-7804)
- Name-level $ORDER() appropriately deals with an extended reference argument. Previously, it ignored the extended reference. In addition, $VIEW("REGION",gname) always uses $zgbldir. Previously it improperly used the global directory used by the last global reference, which could be inappropriate if that was an extended reference. Finally, $REFERENCE includes any extended reference after a Naked Reference. Previously, a Naked Reference after an extended reference operated properly but removed the evidence of its target global directory from $REFERENCE. (GTM-7810)
- GT.M appropriately handles a MUPIP STOP received by a process within a small window while processing a TRESTART while an performing an update involving a trigger. Previously, such an unlikely circumstance could cause the process to terminate with a segmentation fault (SIG-11). FIS encountered this issue in testing and has not received a customer report of such an event. This item was actually addressed in V6.0-003 but was missed in that release note, and so is included here for those who no longer have cause to look at revisions of the V6.0-003 release note.[UNIX](GTM-7811)

- The TRIGNAMEUNIQUE error message correctly states that the maximum number of unique run-time trigger names is 3907. Previously it stated that this number was 3844. [UNIX](GTM-7825)

- When run as root, GT.M components use the owner and group of the database file as the owner and group of newly created journal files, backup files, snapshot files, shared memory, and semaphores. In addition, they set the permissions on the resulting files, shared memory, and semaphores, as if running as the owner of the database file and as a member of the database file group. Note that FIS recommends against running GT.M components as root. Previously, a process running as root set the owner to root, and the group and permissions as for a normal (non-root) user. [UNIX](GTM-7846)

- Replication filters, operating on a source or a receiving instance, apply global nodes containing null subscripts correctly to database regions with Standard Null Collation enabled. Previously, such nodes were stored with GT.M Null Collation which effectively made them inaccessible.(GTM-7854)

- GT.M replication correctly applies global nodes containing null subscripts when the source and receiver sides have different null collation settings (Standard Null Collation vs. GT.M Null Collation). Previously on receiving instances, GT.M stored such nodes with the null collation setting of the source instance, which effectively made them inaccessible. In some cases, the nodes also had incorrect values. (GTM-7856)

- GT.M optimizes non-TP database read operations by bypassing the critical section in many cases. Previously, in application patterns involving frequent updates, non-TP reads typically required a database critical section. (GTM-7865)

- MERGE issues GVSUBOFLOW and MAXNRSUBSCRIPTS error as appropriate in case the target of the MERGE is a global node. Previously, MERGE did not issue these error messages but created beyond-limit global nodes, resulting in a DBMAXNRSUBS error from MUPIP INTEG, or GVSUBOFLOW error from global accesses inside GT.M. (GTM-7867)

- GT.M now issues a single restart of a transaction which encounters an instance freeze in the database file extension process. Previously the transaction would be restarted repeatedly, leading to unnecessary locking contention and CPU consumption. [UNIX] (GTM-7902)

- $INCREMENT() handles the case where its first argument is a spanning node that another process concurrently KILLs. In prior versions starting with V6.0-000 this unusual circumstance could cause a segmentation violation (UNIX SIG11 or OpenVMS ACCVIO) or an unintended result. FIS is not aware of any occurrences of this issue in the production systems. Because spanning nodes are always string values and thus unlikely to be arguments to $INCREMENT() and the concurrency window for the issue is small, we do not expect users to encounter this problem. (GTM-7920)

M-Other Than Database Access

- SOCKET devices now accept LISTEN as an OPEN deviceparameter, treating it the same as a ZLISTEN. Previously, the LISTEN deviceparameter was used with devices in the TCP mnemonicspace. Also, GT.M produces an error - INVMNEMCSPC on UNIX and LIB-E-ACTIMAGE on OpenVMS - in response to an attempt to OPEN a device with the "TCP" mnemonicspace, which was previously obsolete and has long been deprecated. The "SOCKET" mnemonicspace provides access to TCP/IP sockets. On UNIX, other invalid mnemonicspaces also produce an INVMNEMCSPC error, where previously they produced an UNIMPLOP error; on OpenVMS, other invalid mnemonic spaces continue to produce a LIB-E-ACTIMAGE. (GTM-2040)

- USE of a SOCKET device with just a "SOCKET=" device parameter specifying a listening socket checks for an incoming connect request. It accepts the request and creates a new connected socket. $KEY provides information on the new socket in the format described below for WRITE /WAIT: CONNECT|<handle>|<remote>. If the listening socket has no request pending, $KEY has the value of an empty string. Previously USE of a listening socket just made the SOCKET device the current device.

  WRITE /WAIT on a SOCKET device checks for sockets with new connections pending, which it accepts by creating a new connected socket. WRITE /WAIT also checks for data in the internal buffers of already connected sockets as well as for new data (not yet read into the internal buffer) when determining whether a socket has data to READ. When a SOCKET device has more than one socket attached, WRITE /WAIT prioritizes connection requests on listening sockets ahead of data available for reading on already connected sockets, and, within each event type, giving highest priority for the longest ready down to least priority for the most recently ready. WRITE /WAIT selects the socket at the head of the priority list. Previously, WRITE /WAIT did not check the internal buffers and always returned the oldest socket (lowest index in the ZSHOW "D" output) in the current SOCKET device with an available event.

  After a WRITE /WAIT completes successfully (that is: without timing out if a timeout is specified), the $KEY Intrinsic Special Variable (ISV) contains information describing the selected socket in the form: CONNECT|<handle>|<remote or READ|<handle>|<remote>. For TCP sockets, <remote> shows the port number, and for LOCAL sockets it shows the path to the socket file. If the WRITE /WAIT times out or there are no additional ready sockets beyond the one selected by the WRITE /WAIT, $KEY has the value of an empty string.

  The $ZKEY ISV contains semi-colon (";") separated list of entries detailing any waiting sockets for a current SOCKET device. Each $ZKEY entry has the vertical-bar separated form: LISTENING|<handle>|<remote> for an incoming connection or READ|<handle>|<remote> for data available to READ. (GTM-3731)

- GT.M now begins listening immediately on the OPEN or USE of a SOCKET device with the LISTEN or ZLISTEN deviceparameter set. The listen queue length defaults to 1. The "WRITE /LISTEN(numexpr)" may still be used to set the queue length. Previously GT.M only started listening when the application issued a WRITE /LISTEN(numexpr). The socket state as observable through $KEY and ZSHOW "D" is now set to "LISTENING" after a successful LISTEN/ZLISTEN; previously it was set to a "BOUND" state. The "BOUND" state now only occurs in error conditions. (GTM-4797)

- A SOCKET device reports an appropriate error for an improperly formatted argument to the [Z]LISTEN deviceparameter Previously, such an error caused the process to terminate. (GTM-4800)

- GT.M allows arguments present in a function definition in the C/Java external call table to be missing from a corresponding invocation. All non-trailing missing and output-only arguments translate to default values on the receiving side: false in case of gtm_jboolean_t (Java only), 0 in case of all numeric types (Java and C), empty string in case of gtm_char_t * and gtm_char_t ** (C only), a structure with 'length' field matching the preallocation size and 'address' field being a NULL pointer in case of gtm_string_t * (C only), an empty string in case of gtm_jstring_t (Java only), and an empty array in case of gtm_jbyte_array_t (Java only). Previously, omitted non-trailing non-output-only arguments were disallowed; omitted non-

trailing output-only arguments of certain types resulted in arbitrary values passed to the target C or Java routine; and the 'length' field of output-only gtm_string_t * arguments was erroneously set to the length of the corresponding M variable. [UNIX] (GTM-6419)

- The CLOSE command for a PIPE device that is not OPEN'd with the INDEPENDENT deviceparameter uses a timed check on termination status of the PIPE co-process. The TIMEOUT=intexpr specifies the timeout in seconds, defaulting to 2 seconds. Devices other than PIPE ignore the TIMEOUT deviceparameter. The $ZCLOSE Intrinsic Special Variable (ISV) provides termination status of the last PIPE CLOSE as follows: -99 when the check times out, -98 for unanticipated problems with the check, the negative of the signal value if a signal terminated the co-process, and under normal circumstances, the exit status returned by the last co-process. Previously, GT.M did not capture the termination status, and the process executing the CLOSE could hang if the co-process did not terminate. [UNIX](GTM-6666)

- A DETACH operation on a socket now succeeds the first time it is done in a process. Previously, the first DETACH got a SOCKNOTFOUND error. (GTM-6867)

- The JOB command now allows DETACHed sockets (i.e., sockets in the socket pool) to be passed as INPUT, OUTPUT, and/or ERROR,by specifying "SOCKET:<handle>" as the file name, where "<handle>" is the socket handle. On successful completion of theJOB command, any passed sockets are closed and no longer available to the parent process. To pass a socket as the $PRINCIPAL of a child process, pass it as both INPUT and OUTPUT parameters of the JOB command. Previously only file names could be specified for INPUT, OUTPUT, or ERROR. [UNIX] (GTM-7322)

- The ZLINK command accepts and relinks routines on the GT.M invocation stack. Previously, attempting this would produce a LOADRUNNING error. The new behavior can be enabled with VIEW "LINK":"RECURSIVE", or by setting the environmental variable gtm_link to "RECURSIVE". When enabled, whenever an entryref includes a routine name, GT.M accesses the most recently ZLINK'd version of the routine. As usual, GT.M performs auto-ZLINK if no version has yet been linked, and, when an entryref includes no routine name, GT.M accesses the currently executing version. DO, GOTO, ZGOTO, and extrinsic functions invoke the appropriate version; meanwhile, older versions remain referenced by the M virtual machine stack until they execute a GOTO, explicit or implicit QUIT, or ZGOTO. Similarly $TEXT and ZPRINT retrieve the appropriate version's source when available; ZBREAK sets and clears breakpoints in the appropriate version. VIEW "LINK":"NORECURSIVE" and any other value of gtm_link provide the prior LOADRUNNING behavior; $VIEW("LINK") returns the current setting. In addition, ZSHOW offers a new information code, R, and ZSHOW "*" is now equivalent to ZSHOW "IVBDLGRC". Previously, ZSHOW "*" was equivalent to ZSHOW "IVBDLGSC". ZSHOW "R" displays the GT.M invocation stack and, for each routine on the stack, displays an MD5 checksum of its M source code. $VIEW("RTNCHECKSUM",<rtnname>) returns the source check-sum for the most recently ZLINK'd version of <rtnname>. [x86-64 Linux, AIX, Solaris] (GTM-7551)

- GT.M SOCKET devices support LOCAL (also known as UNIX domain) sockets. Except as noted below, LOCAL sockets usage and behavior is the same as TCP sockets.

  - The CONNECT and LISTEN deviceparameters for OPEN and USE accept a value of the form "<pathname>:LOCAL" where pathname is the name of the file to be used for communication. pathname may contain a dollar sign ($) followed by the name of an environment variable which GT.M expands in the same way as the device name for a sequential file. The maximum allowed length of the expanded path name depends on the OS.

    - LISTEN creates the file if it doesn't exist. If the OPEN command specifies the NEWVERSION deviceparameter, the file specified by the pathname exists, and is a socket file, GT.M deletes that file is deleted creates a new file. If the argument does not specify NEWVERSION, or the file is not a socket file, the OPEN produces a SOCKBIND error.

      - When LISTEN is specified for a LOCAL socket on an OPEN, GT.M processes the GROUP, OWNER, SYSTEM, WORLD, UIC, and NEWVERSION deviceparameters as it would for sequential files.

  - CONNECT attempts to open the file. If it doesn't exist or there is no listener, CONNECT retries until it succeeds or a specified timeout expires. Other errors such a the file existing but not being a socket file produce an OPENCONN error.
  - LOCAL sockets do not use the ZDELAY and ZIBFSIZE deviceparameters.
- When CLOSE specifies a LISTENING LOCAL socket with the SOCKET= deviceparameter, the command processes DELETE and DESTROY deviceparameters if specified.
- ZSHOW "D" shows the expanded path name as the value for the LOCAL= attribute.
- $KEY has the path name instead of an IP address after LOCAL socket operations.

(GTM-7554)

- GT.M processes appropriately handle standard input directed to a file and standard output directed to a socket. This situation could arise when an inetd invoked process in turn invoked MUMPS after redirecting its standard input to a file. Previously, such a configuration caused a KILLBYSIGSINFO1 error. [UNIX] (GTM-7603)
- In UTF-8 mode, using both the FIXED and FOLLOW deviceparameter appropriately maintain buffered input when interrupted by an INTRPT, appropriately maintain $X and $Y, and detect proper lengths for fixed length READs (READ x#n). Previously, when a READ resumed from an INTRPT, it lost any input received before the INTRPT and fixed length reads sometimes improperly maintained $X and $Y, which could lead to incorrect READ length and/or hangs. [UNIX](GTM-7709)
- If GT.M detects an error in using a system service to start or stop a system-maintained timer (used internally to schedule timed events, such as HANGs and dirty buffer flushes), it sends a SETITIMERFAILED diagnostic message to the operator log. In some cases, these errors are self-correcting but might cause some actions to take longer than intended; in those cases GT.M tags the operator log message as warning and lets the process continue. In other cases, the process may be subject to indefinite hanging, possibly holding shared resources, and GT.M terminates the process with a fatal SETITIMERFAILED error. Previously, GT.M did not verify the status of timer-setting operations. [UNIX] (GTM-7719)
- A SOCKET OPEN or USE with the LISTEN device parameter properly handles errors while assigning a IP address and port, or path for the socket file to the socket (perform a bind system call). Errors other than EADDRINUSE ("address is already in use") set $DEVICE and, if IOERROR="TRAP" is enabled, raise an error condition. For EADDRINUSE errors on TCP sockets, GT.M reties the operation until the timeout, if specified, expires and then treats it as for other errors. GT.M does not retry EADDRINUSE errors on LOCAL sockets but treats them immediately as an error. Previously, $DEVICE was not set for errors, EADDRINUSE returned without raising an error even if no timeout was specified, and other errors always raised an error regardless of the IOERROR mode. (GTM-7739)
- Auto-ZLINK and ZLINK commands without a .m or .o file extension in their argument determine the need to recompile based on whether the object file was more recently modified than the source file using time in nanoseconds, as provided by the underlying system call. Note that, although the format of the file modification timestamps provides a nanosecond granularity, many supported OSs currently update the file timestamps with an accuracy of one second. Previously, GT.M compared the modification timestamps of source and object files using time in seconds. Also, the GT.M configure installation script now preserves the original timestamps for all files that are not changed by the installation. Preserving timestamps on files, particularly GT.M source and object files, is generally a better practice. Previously, all files in the installation target directory had the installation timestamp.[UNIX] (GTM-7750)
- In UTF-8 mode, READs for a sequential disk (SD) device appropriately detect and report bad character errors (BADCHAR), including correctly setting $DEVICE, $ZA, $KEY and $ZB. Previously, some bad character conditions incorrectly caused SYSTEM-E-UNKNOWN errors and did not appropriately maintain $DEVICE, $ZA, $KEY and $ZB. Also READs for a sequential disk using the FOLLOW deviceparameter appropriately handle timeouts. Previously, such READs could timeout prematurely. [UNIX](GTM-7763)

- GT.M now sets $TEST for "WRITE /WAIT(timeout)" on SOCKET devices. Previously GT.M did not update $TEST in this case. (GTM-7773)
- GT.M handles restart of SOCKET connect operations properly. In V6.0-003, out-of-band operations which occurred during a socket connect caused GT.M to exit with a fatal GTMASSERT2 message while trying to restart the connect. (GTM-7835)
- The sorts-after operator (]]) works correctly with NOUNDEF enabled. Previously, with an operand which was NEW'd after some local variable had been KILL'd, the operator could sometimes produce an incorrect result. (GTM-7836)
- ZGOTO <n>:<entryref> discards local variables NEW'd in levels n+1 and up. Previously, ZGOTO 0:<entryref> did not discard variables NEW'd in level 1.[UNIX](GTM-7842)
- When failing to load ICU libraries on AIX, GT.M now reports the correct library and message in the output error message. Previously, GT.M produced an error message that incorrectly identified the problematic ICU library.[AIX](GTM-7886)
- In UTF-8 mode, compiling a SET $ZPIECE() command with a literal piece separator that is an invalid UTF-8 string, such as $ZCHAR(255), does not raise a BADCHAR error. GT.M does raise BADCHAR error for SET $PIECE(), and the error shows the failing routine, line-number and the line of source that caused the error as for other compilation errors. Previously, GT.M gave inappropriate BADCHAR errors for SET $ZPIECE(), and BADCHAR errors gave no indication of which routine or line caused the error. [UNIX] (GTM-7896)
- The JOB command adds the child process ID to the OUTPUT and ERROR filenames for JOB'd off process after a request to do so by a VIEW command with keyword JOBPID. Previously, if the JOB command occurred in an object residing in a shared library with this VIEW setting, it produced a Segmentation Violation (SIG-11). The workarounds were to use VIEW "NOJOBPID" or to avoid putting routines issuing JOB commands with VIEW "JOBPID" into shared libraries. [UNIX] (GTM-7905)
- A JOB command appropriately handles situations in which the JOB'd process does not start, or terminates (for example with a SIGTERM), without notifying the process issuing the JOB command. In V6.0-002 and V6.0-003 the JOB command waited indefinitely for the nonexistent process to report its status. [UNIX](GTM-7908)
- Certain complicated, deeply nested patterns raise a PATMAXLEN error. Previously, these patterns terminated the process with an segmentation violation (UNIX SIG11 or OpenVMS ACCVIO). Note that the patterns in question are very complex and seem unlikely to be encountered in real applications. FIS encountered this issue in testing and has not received a customer report of such an event. (GTM-7923)

Utilities-MUPIP

- MUPIP INTEG appends DBTN errors showing the block number following DBMBTNSIZMX errors that indicate a transaction number too large on a bit map block. Note that DBMBTNSIZMX is analogous to the DBTNTOOLG error for non-bitmap blocks. In addition, INTEG issues a DBTNLTCTN summary error with associated text including the count of DBMBTNSIZMX as well as DBTNTOOLG errors. Previously, INTEG did not report the bit map block number for a DBMBTNSIZMX or include this type of bit map error in the DBTNLTCTN summary. (GTM-7074)

- JOURNAL BACKWARD RECOVERY or ROLLBACK detects previous interrupted JOURNAL RECOVERY or ROLLBACK correctly. Previously, in some cases MUPIP failed to detect previous JOURNAL RECOVERY or ROLLBACK interruption and incorrectly issued a JNLDBTNNOMATCH error. [UNIX](GTM-7453)

- MUPIP BACKUP and MUPIP REPLIC operations only issue a CRITSEMFAIL error during journal and receiver pool initialization when they encounter a non-transient problem. Previously, these operations could fail if their shared resources were removed by another process in a small window at the journal and receiver pool initialization phase. FIS encountered this issue in testing and has not received a customer report of such an event. [UNIX](GTM-7626)

- GT.M replication provides the option to encrypt communication using Transport Layer Security (TLS) version 1. The API of the encryption plugin has functionality required for TLS, and FIS tests the reference implementation of the encryption plugin against OpenSSL (http://www.openssl.org). As with database encryption, FIS neither recommends nor supports any specific cryptographic library, and you are responsible for selecting and procuring the library that best meets your needs, including support for that library appropriate to your use. FIS GT.M support includes the unmodified reference implementation of the plugin, but does not extend to any version of the plugin containing your modifications. Note: a basic understanding of TLS is assumed - see the Wikipedia article (http://en.wikipedia.org/wiki/Transport_Layer_Security) for more information.

  For TLS support, the reference implementation requires a configuration file, pointed to by the gtmcrypt_config environment variable. This file supports both TLS and database encryption. To support the new configuration file format, the reference implementation also requires the popular libconfig library (http://www.hyperrealm.com/libconfig) to be installed and available; check for pre-packaged distributions for your operating system. Note that even if your environment is configured only for encrypted databases, the reference implementation requires libconfig library to be installed. The reference implementation continues to support the gtm_dbkeys environment variable for configurations that use encrypted databases but not TLS. For backward compatibility, if both $gtmcrypt_config and $gtm_dbkeys are defined, the reference implementation uses $gtm_dbkeys.

  The Source and Receiver Server startup commands support the: TLSID, [NO]PLAINTEXTFALLBACK, and RENEGOTIATE_INTERVAL qualifiers:

  - -TLSID=label specifies a label to identify TLS certificate and private key pairs in the configuration file for the reference implementation to use for the connection. TLSID is a required parameter if TLS is to be used. If private keys are encrypted, an environment variable of the form gtmtls_passwd_<label> specifies the obfuscated password. You can obfuscate passwords using the 'maskpass' utility provided along with the encryption plugin. If you chose to use unencrypted private keys, set the gtmtls_passwd_<label> environment variable to a non-null dummy value; this prevents inappropriate prompting for a password.

  - -[NO]PLAINtextfallback (default NOPLAINtextfallback) specifies whether the replication server is permitted to fallback to plaintext communication. If NOPLAINTEXTFALLBACK is in effect, GT.M issues a REPLNOTLS error in the event it is unable to establish a TLS connection. [Note: GT.M versions prior to V6.1-000 did not support TLS for replication - if needed it could be implemented with an external application such as stunnel (http://stunnel.org).] If PLAINTEXTFALLBACK is in effect, in the event of a failure to establish a TLS connection, GT.M issues REPLNOTLS as a

warning. Once a permitted plaintext replication connection is established for a connection session, GT.M never attempts to switch that connection session to TLS connection.

- -RENEGotiate_interval=minutes qualifier (defaults to a little over 120 minutes) specifies the time to wait before attempting to perform a TLS renegotiation. A value of zero causes GT.M never to attempt a renegotiation. The MUPIP REPLIC -SOURCE -JNLPOOL -SHOW [-DETAIL] command shows the time at which the next TLS renegotiation is scheduled, and how many such renegotiations have occurred thus far for a given secondary instance connection. As renegotiation requires the replication pipeline to be temporarily flushed, followed by the actual renegotiation, TLS renegotiation can cause momentary spikes in replication backlog.

Other configuration parameters required to establish a successful SSL/TLS connection with the reference implementation are specified in a configuration file, described below under Additional information for GTM-7743 - GT.M support for TLS encryption of replication streams using a plug-in, along with an example.(GTM-7743)

- MUPIP JOURNAL -EXTRACT -GLOBAL=<key> works correctly when the key contains a string subscript with embedded double-quotes and the key specification terminates with the wildcard (*). Previously, in this combination of circumstances, JOURNAL -EXTRACT might miss some records the specification should have included. (GTM-7769)

- MUPIP REORG -TRUNCATE -SELECT= does a thorough job of moving vestigial blocks of KILL'd global variables so that they do not linger after unused space, thereby making it possible to release unused space efficiently. Previously, GT.M did not move these vestigial blocks in all cases, and unused space before any block is not released to the file system. Note that even when an entire global variable is KILL'd, a database file retains two vestigial blocks: a root block and a single empty data block for that global variable. [UNIX](GTM-7820)

- MUPIP EXTRACT robustly handles BINARY output format. Additionally, MUPIP EXTRACT, in BINARY mode, operates faster on encrypted databases. Previously, in certain rare cases involving heavy concurrent updates, MUPIP EXTRACT, in BINARY mode, generated corruptor empty extract files and performed unnecessary work on encrypted databases. [UNIX] (GTM-7827)

- Argumentless MUPIP RUNDOWN, MUPIP RUNDOWN -FILE and MUPIP RUNDOWN -REGION do not remove semaphore of a region in use. Previously, MUPIP RUNDOWN of a file with only read-only users or using differing global directories with RUNDOWN -REGION '*' could inappropriately remove the semaphore for a region, but leave its corresponding shared memory. In this state, a subsequent attempt by a process to first access a database resulted in REQRUNDOWN error. The workaround was to do an additional MUPIP RUNDOWN with no read-only processes active. [UNIX](GTM-7845)

- MUPIP INTEG -SUBSCRIPTS works correctly if the argument to the -SUBSCRIPTS qualifier specifies a null subscript. Previously this terminated abnormally with a segmentation violation (UNIX SIG-11 or OpenVMS ACCVIO). (GTM-7853)

- The Replication Receiver Server handles a case where a flow control message arrives at the receiver pool boundary. In V6.0-003, the work done for GTM-7631 introduced a new issue managing this case which caused a segmentation violation (SIG11). [UNIX](GTM-7890)

- If an external replication filter fails to respond within just over a minute to delivery of a mini-transaction or TP transaction, a Source or Receiver Server issues a FILTERTIMEDOUT error, stops the filter, and exits. Previously, the servers could hang indefinitely waiting for a reply from the external filter. [UNIX](GTM-7892)

- Replication processes and MUPIP RUNDOWN appropriately handle a journal pool deadlock check performed while waiting for access to the journal pool resource. Previously, under unusual circumstances, such a check could fail with a segmentation violation (SIG-11). FIS encountered this issue in testing and has not received a customer report of such an event. (GTM-7910)

- MUPIP ONLINE ROLLBACK robustly handles cases where it fails to acquire standalone access on the Journal and/or Receive Pool. Previously, in such cases, ONLINE ROLLBACK terminated with a segmentation violation (SIG-11). FIS encountered this issue in testing and has not received a customer report of such an event.[UNIX](GTM-7916)

Utilities-Other Than MUPIP

- This release note is included in the Utilities - Other Than MUPIP section because the most visible changes to the user interface are in GDE. Global variables with the same unsubscripted name can span multiple database regions.

  To effect this change, global directories support mapping at the level of subscripts, not just based on the unsubscripted global name.

  Specify the subscripted name of root nodes to map entire subtrees to a regions, for example:

```
GDE> add    -name IMPL                          -region OTHERMUMPS  ! Map MUMPS implementations to OTHERMUMPS
GDE> add    -name IMPL("GT.M")                  -region MYMUMPS     ! While mapping GT.M to MYMUMPS
```

  Specify ranges with a colon (:) to map ranges of subscripted names and their subtrees to a regions. for example:

```
GDE> add    -name PRODAGE(0:10)                 -region DECADE0     ! Ranges are closed on the left and open
 on the right
GDE> add    -name PRODAGE(10:20)                -region DECADE1     ! PRODAGE(10) maps to DECADE1
GDE> add    -name PRODAGE(20:30)                -region DECADE2
```

  Ranges must be specific numbers or strings - GDE does not support wildcarding (using "*") in ranges.

  You can use $CHAR() and $ZCHAR() to specify unprintable characters; the arguments must be positive integers (exponential - E syntax not allowed), and valid code points for $CHAR() or in range for $ZCHAR(), both with respect to the current $ZCHSET. Also, "" (an empty string) or no value (e.g. 20: or :20 or :) specify open-ended ranges, which span, on the left, from the first subscript ("") to, on the right, the last possible string , for example:

```
GDE> add    -name PRODAGE(:10)                  -region DECADE0     ! This line and the next are equivalent
GDE> add    -name PRODAGE(“”:10)                -region DECADE0     ! numbers up to 10
GDE> add    -name PRODAGE(20:)                  -region DECADE2     ! 20 thru all numbers (> 20) + strings
GDE> add    -name PRODAGE(20:“”)                -region DECADE2     ! same as the add just above
```

  Here is the syntax to map numeric subscripts and strings to separate regions

```
GDE> add    -name MODELNUM                      -region NUMERIC
GDE> add    -name MODELNUM($char(0):)           -region STRING
```

  You can map global variables with the same unsubscripted name at multiple subscript levels, for example:

```
GDE> add    -name DIVISION(“Europe”,”a”:”m”)    -region EUROPEAL
GDE> add    -name DIVISION(“Europe”,”m”:”z”)    -region EUROPEM
GDE> add    -name DIVISION(“Australia”)         -region AUSTRALIA
GDE> add    -name DIVISION(“USA”,”South”,”a”:”m”)  -region USSAL
GDE> add    -name DIVISION(“USA”,”South”,”m”:”{”)  -region USSMZ
GDE> add    -name DIVISION(“USA”,”WestCoast”)   -region USWC
```

  GDE merges overlapping ranges that map to the same region, for example:

```

GDE> add    -name X(1:10)                       -region=AREG
GDE> add    -name X(6:15)                       -region=AREG
GDE> show -name

```

```
    *** NAMES ***
Global                          Region
-------------------------------------------------------------------------------
*                               DEFAULT
X(1:15)                         AREG
```

Specifying overlapping ranges that map to different regions produces an error, for example:

```
GDE> add    -name X(1:10)                       -region=AREG
GDE> add    -name X(6:15)                       -region=BREG
%GDE-E-NAMRANGEOVERLAP, Range in name specifications X(6:15) and X(1:10) overlap using collation sequence #0
```

Note that sub-ranges that map to different regions do not produce an error; for example: mapping X(1:15) and X(6:10) works because the second is entirely within the first. Only overlapping ranges (those that have a non-intersecting portion) that map to different regions produce this error.

Because string subscripts are subject to collation (the unsubscripted portion of a global variable name and numeric subscripts are not), GDE needs to know the collation sequence number associated with each unsubscripted global variable name. Zero (0) is the number for default M collation. As a consequence, when you use alternative collation(s) (other than 0), the collation transforms must be available to GDE in the same way as they are to other GT.M components. All of a global (all nodes sharing the same unsubscripted global name) must have a single collation, which is implicitly the case for globals that do not span multiple regions. Globals that do not span multiple regions and do not have any collation characteristics defined in the GBLNAME section of the global directory take on the default collation characteristics defined in the database region to which they map. On the other hand, globals that span multiple regions have their collation implicitly (collation 0), or explicitly, established by the GBLNAME section of the global directory and cannot adopt a differing collation based on the region collation characteristic. Because GT.M determines collation for globals spanning multiple regions by the GBLNAME characteristic, which cannot change once the database files are created, GDE reports collation on many error messages. In the above example, although the overlapping ranges are numeric, GDE reports the collation sequence number as part of the error message.

GDE merges and splits ranges as it reads each NAME command. Therefore, the following does not generate an error because the second command merges the ranges and the third splits an existing range, whereas reversing the second and third would trigger the NAMRANGEOVERLAP error.

```
GDE> add    -name X(1:10)                       -region=AREG
GDE> add    -name X(6:15)                       -region=AREG
GDE> add    -name X(6:15)                       -region=BREG
GDE> show   -name

       *** NAMES ***
Global                          Region
 -------------------------------------------------------------------------------
 *                               DEFAULT
 X(1:6)                          AREG
 X(6:15)                         BREG
```

A point specification overrides a range specification at the same level, for example:

```
GDE> add    -name a(1:5)                       -region=AREG
GDE> add    -name a(3:4)                       -region=BREG
GDE> add    -name a(3)                         -region=CREG
GDE> show -name
```

```
        *** NAMES ***
 Global                          Region
 ------------------------------------------------------------------------------
 *                               DEFAULT
 a(1:3)                          AREG
 a(3)                            CREG
 a(3:4)                          BREG
 a(4:5)                          AREG
```

Adding a name and deleting it can change the mapping, for example, note the "hole" left between 80 and 90:

```
GDE> add    -name a(1:100)                      -region=AREG
GDE> add    -name a(80:90)                      -region=BREG
GDE> delete -name a(80:90)
GDE> show   -name

        *** NAMES ***
 Global                          Region
 ------------------------------------------------------------------------------
 *                               DEFAULT
 a(1:80)                         AREG
 a(90:100)                       AREG
```

Specifying separate entries for a subtree provides a way to map a root to a different region, for example, note X(2) maps to BREG whereas the rest of of X is mapped to AREG:

```
GDE> add    -name X                             -region=AREG
GDE> add    -name X(2)                          -region=BREG
GDE> add    -name X(2,:)                        -region=AREG
```

Out-of-order ranges generate errors

```
GDE> add    -name a(20:10)                      -region=AREG
%GDE-E-NAMRANGEORDER, Range in name specification a(20:10) specifies out-of-order subscripts using collation
 sequence #0
```

GDE SHOW -NAME and GDE SHOW -MAP display longer lines if subscripted names exceed the 36 column limit, and display non-printable characters in string subscripts using $CHAR() or $ZCHAR() notation.

The SHOW -MAP command uses "++" as a suffix to mean "the next lexical subscript", for example:

```
GDE> add    -name a(1)                          -region=A1
GDE> show   -map

                                *** MAP ***
   -  -  -  -  -  -  -  -  -  - Names -  -  - -  -  -  -  -  -  -
 From                            Up to                           Region / Segment / File(def ext: .dat)
-------------------------------------------------------------------------------------------------------------------
 %                               a(1)                            REG = DEFAULT
                                                                 SEG = DEFAULT
                                                                 FILE = mumps.dat
```

```
a(1)                            a(1)++                          REG = A1
                                                                SEG = NONE
                                                                FILE = NONE
a(1)++                          ...                             REG = DEFAULT
                                                                SEG = DEFAULT
                                                                FILE = mumps.dat
```

Here is an example of a complex mapping:

- from ^x, upto but not including ^x(5,10), maps to REG1
- from ^x(5,10), upto but not including ^x(20,40,50), maps to to REG2
- from ^x(20,40,50) through the last subscript in ^x maps to REG 3

```
GDE> add    -name x                             -region=REG1
GDE> add    -name x(5)                          -region=REG1
GDE> add    -name x(5,10:)                      -region=REG2
GDE> add    -name x(5:20)                       -region=REG2
GDE> add    -name x(20)                         -region=REG2
GDE> add    -name x(20,40)                      -region=REG2
GDE> add    -name x(20,40,50:)                  -region=REG3
GDE> add    -name x(20,40:)                     -region=REG3
GDE> add    -name x(20:)                        -region=REG3
```

The new object-type GBLNAME (existing object-types are NAME, REGION, and SEGMENT) with the COLLATION characteristic provides a mechanism to specify the collation to use for global variables sharing the same unsubscripted name.

```
GDE> add    -gblname a                          -collation=0
GDE> show   -gblname

        *** GBLNAMES ***
 Global                            Coll  Ver
 ------------------------------------------------------------------------------
 a                                    0    0
```

GDE requires the library for any non-zero collation to be available.

```
GDE> add    -gblname a                          -collation=1
%GDE-E-GBLNAMCOLLUNDEF, Error opening shared library of collation sequence #1 library for GBLNAME a
```

Regions that contain global variables sharing the same unsubscripted name that span regions must use standard null collation; attempting to use the deprecated original null collation produces an error, for example:

```
GDE> add    -name a(1)                          -region=areg
GDE> add    -region areg                        -dynamic=areg
GDE> add    -segment areg                       -file=areg
GDE> verify
%GDE-I-STDNULLCOLLREQ, Region DEFAULT needs Standard Null Collation enabled because global ^a spans through it
%GDE-I-STDNULLCOLLREQ, Region AREG needs Standard Null Collation enabled because global ^a spans through it
%GDE-I-VERIFY, Verification FAILED
```

Notes

- At this time, triggers are not supported for global variables sharing the same unsubscripted name that span regions.
- When subscripts are real numbers that are not exactly represented in GT.M's internal numeric representation, or which result in numeric errors such as overflow, GDE generates an error.
- No part of a region-spanning global variable sharing the same unsubscripted name can map to a remote database (accessed via GT.CM GNP).
- By accessing multiple regions, a single global reference, for example a call to $DATA(), can update $VIEW("GVSTAT",region) counters for multiple regions.
- To ensure the atomicity, consistency and isolation of operations such as KILL that can update multiple database regions, the GT.M database engine can invoke transaction processing logic even when application logic does not use transaction processing. Thus, operations on global variables having the same unsubscripted name that span regions can update TP transaction counters in database file headers.
- When a global spans regions, you need to ensure all the spanned regions have appropriate characteristics (region KEY_SIZE, NULL_SUBSCRIPTS, and RECORD_SIZE) for the nodes they store. If the data is uniform across the regions, this typically means the spanned regions should have the same characteristics. You can use GDE templates to easily do this. Note that segment RESERVED_BYTES impacts available record size.
- When a TP transaction contains data in more than one segment, to avoid TRANS2BIG errors, you need to ensure that no one segment exceeds the transaction size limit. BLOCK_SIZE and GLOBAL_BUFFERS contribute to determining the transaction size limit for a database segment. This is not a new issue, but spanning regions may make the determination of transaction size limits more complex.

Caution

As MUPIP EXTRACT processes all nodes in one region of a global variable sharing the same unsubscripted name and mapping across multiple regions before moving on to the next region, nodes in the output of MUPIP EXTRACT may not be in sequential order, especially if subscripts in one region bracket subscripts in another. In the edge case where blocks in a region contain nodes with subscripts mapped to that region as well as nodes with subscripts mapped to another region according the current global directory, the output of MUPIP EXTRACT (which extracts all the nodes in a block before moving on to the next block) includes those nodes, whereas the output of ZWRITE which steps subscript by subscript through a global variable having the same unsubscripted name as determined by the current global directory does not. On the other hand, if the current global directory does not map any of a global to a region, EXTRACT does not check that region for nodes in that global.

Caution

GT.M associates LOCKs with resource names starting with an up-arrow (^) with the database region holding the global with the same unsubscripted name as the resource. For globals spanning multiple regions and multiple global directories this means that up-arrow LOCK resource names may not address a locking protocol in the same way as for globals that do not span multiple regions. To address this, ensure that all overlapping global directories used with up-arrow LOCK resource names and globals spanning multiple nodes map the unsubscripted global name to the same region. Alternatively, you can use LOCK

resource names with no leading up-arrow and control LOCK interactions with the LOCKS global directory characteristic.

In conjunction with this change:

- M-Database Access
  - For global variables having the same unsubscripted name that span regions, the global directory must use exactly the same collation sequence used by the database files. A mismatch results in an ACTCOLLMISMTCH error. This requirement does not apply to global variables having the same unsubscripted name that are mapped to a single region by the global directory
  - Globals that span multiple regions currently only support a value of zero for numeric collation (that is, numbers collate as numbers, not as strings).
  - $VIEW("REGION",gvn) accepts a subscripted or unsubscripted global variable name, and reports all regions spanned by the tree with that unsubscripted global variable name as the root. The first region is the region to which the root maps; other regions are in the order in which they would be encountered by traversing the subscripts in order, with duplicates removed. For example, given the global directory configuration created with:

```
GDE> add    -name a(1:10)                       -region=a1
GDE> add    -name a(120:300)                    -region=a2
GDE> add    -name a(60:325)                     -region=a3
GDE> show   -name
     *** NAMES ***
 Global                            Region
 ------------------------------------------------------------------------------
 *                                 DEFAULT
 a(1:10)                           A1
 a(60:120)                         A3
 a(120:300)                        A2
 a(300:325)                        A3
```

    Here are some $VIEW("REGION",gvn) outputs:

```
GTM>w $view("REGION","^a(1)")
A1
GTM>w $view("REGION","^a(10)")
DEFAULT
GTM>w $view("REGION","^a(60)")
A3
GTM>w $view("REGION","^a")
DEFAULT,A1,A3,A2
```

- To increase the useful information reported, the GVSUBOFLOW error message truncates overly long keys (gvns) it reports at GT.M's maximum key size (currently 1019 bytes). Previously, it truncated overly long keys at the maximum key size for that database file.
- M-Other Than Database Access

- GT.M detects invalid numeric representations where E notation does not end with a valid integer and issues a SPOREOL error. Previously if the E was followed by nothing or only a sign, GT.M evaluated the expression as having an exponent of 0, that is: only the value to the left of the E - for example 10E as 10.

- As the representation of subscripts in the database differs from that in process address space, new $VIEW() commands allow application code to view the database representation of subscripts. $VIEW("YGVN2GDS",<gvn>[,collnum]) provides the database representation of gvn, for example:

```
GTM>set x=$VIEW("YGVN2GDS","^A(1,""abcd"")") zwrite x for i=1:1:$zlength(x) write $zascii($zextract(x,i)),"
 "
x="A"_$C(0)_" "_$C(17,0,255)_"abcd"_$C(0,0)
65 0 191 17 0 255 97 98 99 100 0 0
GTM>
```

  $VIEW("YGDS2GVN",<gds>[,collnum]) is the inverse function, e.g., with the value of x from above:

```
GTM>set y=$VIEW("YGDS2GVN",x)  zwrite y
 y="^A(1,""abcd"")"

GTM>
```

- The $VIEW("REGION",<namevalue>) function produces the GVSUBSERR when <namevalue> is a subscripted global variable not in the same form as that generated by $NAME(). You can use $NAME() inside $VIEW() to ensure that subscripts are in a correct form, for example, $VIEW("REGION",$NAME(^abcd(1,2E4))) instead of $VIEW("REGION","^abcd(1,20000)").

- GT.M appropriately handles the case where the application code makes a legal global reference, makes a global reference that produces a KEY2BIG error followed by a reference to a global with the same unsubscripted name as the reference before the KEY2BIG error. A regression introduced in V6.0-000 caused this sequence to produce a GTMASSERT error.

- %GBLDEF enhancements include:

  - $$kill^%GBLDEF and $$set^%GBLDEF return 0 for spanning globals. Use the global directory (GBLNAME object in GDE) to set collation characteristics for spanning globals.

  - $$kill^%GBLDEF and $$set^%GBLDEF return 0 if $tlevel is non-zero. Ensure these collation characteristic modifying commands are executed outside of a TSTART/TCOMMIT fence.

  - $$get^%GBLDEF now accepts a "reg" argument i.e. $$get^%GBLDEF(gname,reg) where reg is optional. This examines the collation characteristics of global "gname" in a specific region in case the global spans multiple regions. If "gname" global does not map to input "reg" region in the current global directory, $$get^%GBLDEF returns 0.

  - $$get^%GBLDEF(gname[,reg]) checks for collation information in 3 locations : the directory tree, global directory (GBLNAME section), database file header. Information found in one location (comma separated list of 3 values) is returned right away. If not it moves on to the next location in that order. If not found in any of the 3 locations, it returns a "0". Any errors also return "0". The directory tree and database file header locations correspond to the optional "reg" parameter (if specified) or the region to which the current global directory maps <gname>.

- Utilities-MUPIP

- When MUPIP CREATE adjusts the autoswitchlimit parameter to ensure proper alignment with allocation and extension, it may issue a JNLALLOCGROW message and set the journal allocation value to the minimum supported autoswitchlimit (currently 16K).

- MUPIP EXTRACT displays details by region for output derived from globals that span regions, and a summary line at the end, for example, when ^w and ^xx do not span regions, but ^x does:.

```
$ mupip extract x.ext
%GTM-I-RECORDSTAT, ^w:    Key cnt: 10  max subsc len: 6  max data len: 200  max rec len: 210
%GTM-I-RECORDSTAT, ^x (region REG1):      Key cnt: 3333  max subsc len: 10  max data len: 250  max rec len:
 264
%GTM-I-RECORDSTAT, ^x (region REG2):      Key cnt: 3333  max subsc len: 10  max data len: 150  max rec len:
 164
%GTM-I-RECORDSTAT, ^x (region REG3):      Key cnt: 3334  max subsc len: 10  max data len: 200  max rec len:
 214
%GTM-I-RECORDSTAT, ^x:    Key cnt: 10000  max subsc len: 10  max data len: 250  max rec len: 264
%GTM-I-RECORDSTAT, ^xx:   Key cnt: 10  max subsc len: 7  max data len: 200  max rec len: 211
%GTM-I-RECORDSTAT, TOTAL:         Key cnt: 10020  max subsc len: 10  max data len: 250  max rec len: 264
```

- For a global that spans multiple regions, the RECORDSTAT informational messages generated by MUPIP EXTRACT include per-region data for that global as well as aggregate data across all spanned regions. The per-region RECORDSTAT message indicates the region name in parentheses after the global name. The summary RECORDSTAT message does not have any region name.

- MUPIP REORG and MUPIP SIZE print the region name (as mapped to by the current global directory) in parenthesis after the global name for each global they operate on.

- Utilities - Other Than MUPIP

  - When DSE FIND -KEY determines that the specified key (gvn) does not map to the current region, it additionally displays the region that that key maps to in the current global directory. The -NOGBLDIR qualifier supresses the region warning functionality. -GBLDIR is the default, but may also be explicitly specified. This addresses the need to locate the appropriate region when multiple database files contain nodes from the same unsubscripted global variable name.

  - GDE accepts LOCKSPACE up to the maximum of 64Ki (512 byte) pages for MM databases. Previously, it restricted MM databases to 1,000 pages, requiring a MUPIP SET after MUPIP CREATE to configure larger LOCKSPACE sizes.

  - For MM, GDE prints the VALTOOBIG error if reserved_bytes is specified more than 64K. Previously, the errors it produced in this situation were confusing.

  - GDE uses space-dash (" -") as the delimiter to introduce a qualifier, consistent with UNIX shell parsing conventions. Previously, GDE did not require the space before the dash.

  - In OpenVMS, GDE restricts keys to a maximum supported size of 255 bytes. This corrects a regression introduced when the maximum supported key size on UNIX grew to 1018 bytes. [OpenVMS]

  - In OpenVMS, the PREFIXBAD error message mentions that the default region ($DEFAULT) need not start with an alphabetic character; in UNIX the message remains unchanged. [OpenVMS]

  - GDE does not require BEFORE or NOBEFORE to be specified when defining journaling characteristics - when this characteristic is no explicitly specified, GDE inherits the setting defined in the appropriate template. Previously, skipping this specification produced a "%GDE-E-QUALREQD, Before_image required" error.

- GDE SHOW -COMMANDS -FILE=filename restricts output to the specified file. Previously it produced output to the screen as well as to the filename specified.
- GDE SHOW -COMMANDS produces a compact and readable output, adjusting templates to minimize the quailfiers needed on other commands. Please consider this when creating new regions and segments in a global directory created from a SHOW -COMMANDS output as the defaults come from the templates. Previously, SHOW -COMMANDS produced a more literal representation of the current state, retaining the template defaults unchanged.
- When invoked from an M program with a DO, GDE returns control to the calling M program. Previously, GDE terminated the process with a HALT on an EXIT or a QUIT issued to GDE. This behavior of GDE is a change that is not backwards compatible. Note that if you invoke GDE from a GT.M process, and modify the map of global directly currently opened by that process, you must HALT and restart the process for the revised mapping to take effect. FIS intends for an invocation of GDE that does not change the mapping of any global directory loaded by the process to leave the process state unchanged on return; please report any change in process state under these conditions to your GT.M support channel. FIS expects users normally run GDE from the shell via mumps -run GDE, and in this normal use case, there is no change to GDE behavior.

- The GBLNAME object and COLL qualifier in GDE specify collation characteristics for a global name; a value of zero for COLL qualifier to specifies standard M collation. The first time that a GT.M processes accesses a global variable name in a database file, it determines the collation sequence as follows:
  - If a Global Variable Tree (GVT) exists (that is, global variable nodes exist, or have previously existed, even if they have been KILL'd), use the existing collation:
    - If there is a collation specified in the Directory Tree (DT) for that variable, use it after confirming that this matches the collation in the global directory.
    - else (that is, there is no collation specified in the DT):
      - If there is collation specified for that global variable in the global directory use it
      - else if there is a default for that database file, use it
      - else (that is, neither exists), use standard M collation
  - else (that is, a GVT does not exist, which in turn means there is no DT):
    - If there is collation specified for that global variable in the global directory use it
    - else, if there is a default for that database file, use it
    - else (that is, neither exists), use standard M collation

  If the collation specified in the DT does not match that specified in the global directory, GT.M issues an ACTCOLLMISMTCH error. Previously, GT.M did not store collation information in the global directory. Also, previously, GT.M did not protect itself against inadvertently reporting and using incorrect subscripts of an existing global variable due to a change in the collation sequence in the database file header made with DSE. Note that such a change with DSE would have required abnormal, and not recommended, actions on the part of an operator.

[UNIX, except as noted] (GTM-2168)

- GDE ignores spaces in file names supplied for scripted operation invoked with @; Previously, accidental trailing spaces could cause hard to diagnose errors. (GTM-5572)
- DSE remains in the same region after a DUMP -BLOCK operation where records in the dumped block contained global names mapping to a different region. Previously DSE incorrectly switched to the region mapped, causing subsequent DSE commands to operate on the wrong database file. (GTM-7562)
- GDE treats text after an exclamation point (!) as a comment, whether or not in a command file. Exclamation points within quoted strings do not create comments. Previously, GDE only treated command file lines starting with an exclamation point as comments. (GTM-7693)
- GDE accepts the -MUTEX_SLOTS=<i> qualifier for segment objects with minimum of 1Ki and a maximum of 32Ki. This facilitates establishing an initial value for this database parameter. Previously, setting -MUTEX_SLOTS required MUPIP SET, which is still available for adjusting existing databases. (GTM-7742)
- The bypass mechanism DSE and LKE use to open a busy database uses more robust logic to report their action. Previously the process could generate a series of misleading error messages and fail. The workaround was to retry the operation. [UNIX] (GTM-7848)
- gtmsecshr ignores requests to ensure that a non-existent process is not suspended. Previously, it issued a warning under these circumstances, which can arise when a process holds a shared resource just before terminating. [UNIX](GTM-7858)
- GTMJI allows assigning ISVs to M gtm_string_t-type input/output arguments in call-in invocations. Previously, assigning the values of certain ISVs to gtm_string_t-type input/output arguments caused segmentation violations. [IBM System p AIX, Sun SPARC Solaris, x86_64 GNU/Linux, x86 GNU/Linux] (GTM-7866)

More Information

Additional information for GTM-7743 - GT.M support for TLS encryption of replication streams using a plug-in

Configuration File

The configuration file, pointed to by $gtmcrypt_config, is divided into two sections: Database encryption section and the TLS section.

```
/* Database encryption section */

database: {
      keys: (
            {
                  dat: "/tmp/mumps.dat";  /* Encrypted database file. */
                  key: "/tmp/mumps.key";  /* Encrypted symmetric key. */
            },
            {
                  dat: "/tmp/a.dat";
                  key: "/tmp/a.key";
            },
            ...
      );
}

/* 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: "/gtc/staff/gtm_test/current/tls/certs/CA/gtmCA.crt";

      /* CApath: points to a directory containing CA certificates in PEM format. The files each contain one CA
 certificate. The files are
       * looked up by the CA subject name hash value, which must hence be available. 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: "/gtc/staff/gtm_test/current/tls/certs/CA/";

     /* Diffie-Hellman parameters used for key-exchange. Either none or both have to be specified. If neither is
specified, then
      * then the data is encrypted with the same keys that are used for authentication.
      */
     dh512: "/gtc/staff/gtm_test/current/tls/dh512.pem";
     dh1024: "/gtc/staff/gtm_test/current/tls/dh1024.pem";

     /* crl: points to a file containing list of revoked certificates. This file is created by the openssl
utility. */
     crl: "/gtc/staff/gtm_test/current/tls/revocation.crl";

     /* Timeout (in seconds) for a given session. If a connection disconnects and resumes within this time
interval, the session
      * is reused 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;

     /* List of certificate/key pairs specified by identifiers. */
     PRODUCTION: {
           /* 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: "/gtc/staff/gtm_test/current/tls/certs/Malvern.crt";
           /* Path to the private key. If the private key is protected by a passphrase, an obfuscated version of
the password
            * should be specified in the environment variable which takes the form gtmtls_passwd_<identifier>. For
instance,
            * for the below key, the environment variable should be 'gtmtls_passwd_PRODUCTION'.
            * Currently, the GT.M TLS plug-in only supports RSA private keys.
            */
           key: "/gtc/staff/gtm_test/current/tls/certs/Malvern.key";
     };

     DEVELOPMENT: {
           format: "PEM";
           cert: "/gtc/staff/gtm_test/current/tls/certs/BrynMawr.crt";
           key: "/gtc/staff/gtm_test/current/tls/certs/BrynMawr.key";
     };
};
```

On certain platforms, with an OpenSSL or Libgcrypt installation that provides a validated FIPS 140-2 implementation (see http://www.openssl.org/docs/fips/), with the environment variable $gtmcrypt_FIPS set to 1 (or other case-insensitive affirmative forms: "yes" or "true"), the reference implementation attempts to use OpenSSL or Libgcrypt to provide encryption that complies with FIPS 140-2. See the table below the list of platforms on which FIS has tested the reference implementation with FIPS 140-2 compliant libraries.

More Information

Platform	Libgcrypt	OpenSSL	OpenSSL FIPS
Linux x86_64	1.4.5	1.0.0	1.0.1e
Linux x86	1.4.5	1.0.0	1.0.1e
AIX RS600	1.5.1	1.0.0e	1.0.1e
SunOS SPARC	1.5.1	1.0.0j	1.0.1e
HP-UX IA64	1.4.1	1.0.1e	1.0.1e

On establishing a TLS connection, the Source and Receiver servers record pertinent information (the protocol, encryption algorithm, remote certificate details, etc.) in their log files. Below is an example of how this information might look:

```
SSL/TLS Session:
  Protocol Version: TLSv1
  Session Algorithm: DHE-RSA-AES256-SHA
  Compression: NONE
  Session Reused: NO
  Session-ID: 53326BB9724A08C3884409A9B8415A6FBFBDDFEB3F572344B732941E698E106F
  Session Expiry: Mon Dec 16 09:39:45 2013
  Secure Renegotiation IS supported
Peer Certificate Information:
  Asymmetric Algorithm: rsaEncryption (4096 bit)
  Subject: /C=US/ST=PA/O=GT.M, Inc./L=Malvern/CN=instance2.gtm.com/OU=GT.M Test System Instance2/
emailAddress=instance2@gtm.com
  Issuer: /C=US/ST=PA/O=GT.M, Inc./L=Malvern/CN=www.gtm.com/OU=GT.M Certificate Authority/
emailAddress=gtmca@gtm.com
  Validity:
    Not Before: Oct  4 04:09:20 2013 GMT
    Not After: Oct  2 04:09:20 2023 GMT
```

TLS authentication

To use TLS, the communicating parties need to authenticate each other. If the authentication succeeds, the parties encrypt the subsequent communication. TLS authentication uses certificates signed by Certificate Authorities (CAs). Certificate Authorities' certificates themselves are signed (and trusted) by other CAs eventually leading to a Root CA, which self-signs. its own certificate Although the topic of certificates, and the use of software such as OpenSSL is well beyond the scope of GT.M documentation, the steps below illustrate the quick-start creation of a test environment using Source and Receiver certifications with a self-signed Root CA certificate.

Generating a self-signed root certificate authority

Creating a root certificate authority involves three steps.

1. Generate a private key with the OpenSSL command: **openssl genrsa -des3 -out ca.key 4096**. The command prompts for a password with which to protect the private key.
2. Generate a self-signed certificate with the OpenSSL command: **openssl req -new -x509 -days 365 -key ca.key -out ca.crt**. The command first prompts for the password of the private key followed by a series of interactive queries regarding the attributes of the certificate. Below is sample output:

```
Enter pass phrase for ca.key:
You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:PA
Locality Name (eg, city) []:Malvern
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Example Pvt. Ltd
Organizational Unit Name (eg, section) []:Certificate Authority
Common Name (e.g. server FQDN or YOUR name) []:www.example.com
Email Address []:example@example.com
```

At this point, ca.crt is a root certificate that can be used to sign other certificates (including intermediate certificate authorities). The private key of the root certificate must be protected from unauthorized access.

Generating leaf-level certificates

The root certificate is used to sign regular, leaf-level certificates. Below are steps showing the creation of a certificate to be used to authenticate a GT.M Source Server with a GT.M Receiver Server (and vice-versa).

1. Generate a private key. This is identical to step (a) of root certificate generation.

2. Generate a certificate sign request with the OpenSSL command **openssl req -new -key client.key -out client.csr**. The command first prompts for the password of the private key followed by a series of interactive queries regarding the attributes of the certificate. Below is sample output:

   ```
   Enter pass phrase for client.key:
   You are about to be asked to enter information that will be incorporated into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   ```

   ```
   Country Name (2 letter code) [AU]:US
   State or Province Name (full name) [Some-State]:PA
   Locality Name (eg, city) []:Malvern
   Organization Name (eg, company) [Internet Widgits Pty Ltd]:XYZQ International
   Organizational Unit Name (eg, section) []: OurSourceServer
   Common Name (e.g. server FQDN or YOUR name) []:www.xyzq.com
   Email Address []:xyzq@xyzq.com
   Please enter the following 'extra' attributes to be sent with your certificate request
   A challenge password []:challenge
   An optional company name []:XYZQ Pvt. Ltd
   ```

   Typically, organization that generates the certificate sign then sends it to a certificate authority (or a root certificate authority), which audits the request and signs the certificate with its private key, thereby establishing that the certificate

authority trusts the company/organization that generated the certificate and requested its signing. In this example, we sign the certificate sign request with the root certificate generated above.

3. Sign the certificate sign request with the OpenSSL command: **openssl x509 -req -days 365 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out client.crt**. The command also specifies the number of days that the certificates is valid (unless revoked earlier). Below is sample output:

```
Signature ok
subject=/C=US/ST=PA/L=Malvern/O=XYZQ International/OU=OurSourceServer/CN=www.xyzq.com/emailAddress=xyzq@xyzq.com
Getting CA Private Key
Enter pass phrase for ca.key:
```

Important

Keep the self-signed root certificate authority and leaf-level certificates in a secure location. Protect their directories with 0500 permissions and the individual files with 0400 permissions so that unauthorized users cannot access them.

Generating a configuration file

At this point, we have created a leaf-level certificate that is signed by a self-signed root certificate for the GT.M Source instance. After creating and signing a similar leaf-level certificate for the GT.M Receiver Server, we can create a minimal configuration file that can use TLS communication.

```
tls: {
     verify-depth: 7;
     CAfile: "/path/to/ca.crt";
     OurSourceServer: {
           format: "PEM";
           cert: "/path/to/client.crt";
           key: "/path/to/client.key";
     };
};
```

The environment variable $gtmcrypt_config must specify the path to the above configuration file. The environment variable gtmtls_passwd_ OurSourceServer must specify an obfuscated version of the password for the client's private key. Use the maskpass utility provided with your GT.M distribution to create an obfuscated password.

Important

Effective V6.1-000, the gtm_dbkeys environment variable and the master key file it points to are deprecated in favor of the libconfig format encryption configuration file pointed to by the gtmcrypt_config environment variable. Although V6.1-000 supports the use of $gtm_dbkeys for database encryption, FIS plans to discontinue support for it in the very near future. To convert master key files to libconfig format configuration files, please click on  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. Please convert master key files to libconfig format at your earliest convenience.

Known Issues/Limitations

During internal testing, the GT.M group identified that, in rare cases, the Source and Receiver Server failed to successfully establish SSL/TLS handshake (typically after a restart) and exited with the following error:

```
Source:
%GTM-E-TLSHANDSHAKE, Connection to remote side using SSL/TLS protocol failed
%GTM-I-TEXT, error:1408C095:SSL routines:SSL3_GET_FINISHED:digest check failed

Receiver:
%GTM-E-TLSIOERROR, Error during SSL/TLS recv operation
%GTM-I-TEXT, error:1409441B:SSL routines:SSL3_READ_BYTES:tlsv1 alert decrypt error
```

If this happens frequently, consider disabling session reuse by setting "session-timeout" configuration parameter to 0.

Next Steps

Please refer to OpenSSL documentation http://www.openssl.org/docs/ for information on how to create intermediate CAs, Diffie-Hellman parameters, Certificate Revocation Lists, and so on.

Error and Other Messages

ACTCOLLMISMTCH

Global ^gggg inherits alternative collation sequence #nnnn from global directory but database file dddd contains different collation sequence #mmmm for this global

Run Time Error: This indicates that the global gggg inherits collation nnnn from the global directory (globals that span multiple regions inherit collation 0 by default) but the directory tree in database file dddd contains a different collation sequence mmmm for this global.

Action: If nnnn is the right collation sequence to use, fix the database file dddd by using a temporary global directory that maps all names to dddd, extract the global, KILL it, use $$set^%GBLDEF to fix the alternative collation for gggg from mmmm to nnnn, reload the global from the extract, switch back to the regular global directory. If mmmm is the right collation sequence to use, recreate the global directory and define GBLNAME gggg to have that collation instead.

DEVPARMTOOSMALL

DEVPARMTOOSMALL, Deviceparameter must be greater than zero (0)

Run/Compile Time Error: This error occurs when the TIMEOUT=<seconds> deviceparameter of a CLOSE command specifies a value less than one second. For PIPE devices that are not OPEN'd with the INDEPENDENT deviceparameter, the CLOSE command waits for a maximum of TIMEOUT=<seconds> before checking the termination status of the PIPE co-process.

Action: Specify an integer value greater than 0 as the TIMEOUT.

FILEOPENFAIL

FILEOPENFAIL, Failed to open file ffff.

MUPIP error: This message indicates that the MUPIP LOAD failed to open input file ffff.

Action: Please verify path and permission of input file ffff.

FILTERTIMEDOUT

FILTERTIMEDOUT, Replication server timed out attempting to read seqno ssss from external filter

MUPIP Error: This indicates that either a Source or Receiver Replication Server using an external filter took more than 30 sec to read a transaction with journal sequence number ssss from the user supplied external filter's output. The replication server reports this error in its log, stops the filter and terminates.

Action: Determine the cause for the filter's write delay. Fix the filter and restart the replication server with the fixed filter. If you cannot determine reason for delay, report the entire incident context to your GT.M support channel.

GBLNAMCOLLRANGE

GBLNAMCOLLRANGE, Collation sequence #nnnn is out of range (0 thru 255)

GDE Error: This indicates that the collation sequence nnnn is out of the supported range of 0 thru 255.

Action: Specify a collation sequence number inside the supported range.

GBLNAMCOLLUNDEF

GBLNAMCOLLUNDEF, Error opening shared library of collation sequence #nnnn for GBLNAME gggg

GDE Error: This indicates that there was an error opening the shared library for collation sequence nnnn.

Action: Define the environment variable gtm_collate_<nnnn> to point to the shared library for collation sequence nnnn. Also ensure the path to the library is readable and the library is usable on that platform.

GBLNAMCOLLVER

GBLNAMCOLLVER, Global directory indicates GBLNAME gggg has collation sequence #nnnn with a version #vvvv but shared library reports different version #llll

GDE Error: This indicates that the shared library for collation sequence nnnn reported the version as vvvv when collation properties for global name gggg were first added by GDE into the global directory and that this invocation of GDE noticed the shared library reporting an incompatible version llll.

Action: See Action section for COLLTYPVERSION error in the GT.M Message and Recovery Procedures Manual.

GBLNAMEIS

GBLNAMEIS, in gblname gggg

GDE Information: This indicates the gblname where an out-of-range value was specified. This is usually a secondary message and is preceded by a VALTOOSMALL or VALTOOBIG error.

Action: Fixing the preceding error would automatically address this accompanying informational message.

GBLNOMAPTOREG

GBLNOMAPTOREG, Global gggg does not map to region rrrr in current global directory

Run Time Error: This indicates that a VIEW "YDIRTREE" or $VIEW("YDIRTREE") was done with global gggg and region rrrr as parameters but the global does not map to that region in the current global directory.

Action: VIEW "YDIRTREE" or $VIEW("YDIRTREE") is an undocumented feature and so should NOT be used directly. $$get^%GBLDEF is the only tool that uses this but internally catches the GBLNOMAPTOREG error. This means the GBLNOMAPTOREG error message will never be visible to the end-user.

GVSUBSERR

GVSUBSERR

GVSUBSERR, Invalid subscripted global name specification in $VIEW() function

Run Time Error: This indicates that an invalid subscripted global name was specified as the second parameter in a $VIEW("REGION",namevalue) function call.

Action: Fix the syntax error in the subscripted global name. The subscripted global variable name must be in the form returned by $NAME(). For example, use $VIEW("REGION","^abcd(1,20000)") instead of $VIEW("REGION","^abcd(1,2E4)")

INSTFRZDEFER

INSTFRZDEFER, Instance Freeze initiated by eeee error on region rrrrr deferred due to critical resource conflict.

Run Time informational message: eeee error encountered on region rrrrr triggered the Instance Freeze mechanism in an attempt to set the freeze, but couldn't do complete the freeze due to a critical resource conflict. Any process subsequently attempting an update will reattempt the freeze later until one succeeds or the error subsides.

Action: None necessary.

INVCTLMNE

INVCTLMNE, Invalid control mnemonics

Run Time Error: The current device does not support the specified controlmnemonic.

Action: Check the spelling of the controlmnemonic, and be sure the mnemonicspace (if any) for the current device supports the requested usage of the controlmnemonic.

INVMNEMCSPC

INVMNEMCSPC, Unsupported mnemonicspace xxxx

Run Time Error: The mnemonicspace xxxx specified in an OPEN command is not supported by GT.M.

Action: Replace the mnemonicspace with a supported one.

IPADDRREQ

IPADDRREQ, Active connection requires IP address

Run Time Error: This indicates that an OPEN command specified a TCP mnemonicspace; however, it did not have a socket for an active connection or a LISTEN deviceparameter to designate a passive connection.

Action: Determine the type of connection desired and add a SOCKET=socket-id and/or a LISTEN deviceparameter.

ISSPANGBL

ISSPANGBL, Operation cannot be performed on global ^gggg as it spans multiple regions in current global directory

Run Time Error: This indicates that a $$set^%GBLDEF or $$kill^%GBLDEF was attempted on a global that spans multiple regions.

Action: Only $$get^%GBLDEF is supported for spanning globals. Specify a non-spanning global or change the set/kill to a get. For spanning globals, use the GDE ADD -GBLNAME command to set collation characteristics.

JOBSETUP

JOBSETUP, Error receiving aaaa from parent process

Run Time Error: This message indicates that a process created by the JOB command was unable to receive setup information aaaa from the process which issued the JOB command.

Action: Report this and the associated SYSTEM-E-ENO## message to your GT.M support channel.

JOBSTARTCMDFAIL

JOBSTARTCMDFAIL, JOB command STARTUP script invocation failed.

Runtime error: This message indicates STARTUP script specified as JOB command process parameter failed.

Action: Verify the STARTUP script is present, and check it has appropriate permissions to execute.

KEYFORBLK

KEYFORBLK, But block size bbbb and reserved bytes rrrr limit key size to kkkk.

GDE Error: The maximum key for a region must fit in the block size less record overhead and any reserved bytes for that region; kkkk is the maximum key size for block size bbbb.

Action: Reduce the key size or reserved bytes or increase the block size.

KEYWRDBAD

KEYWRDBAD, xxxx is not a valid yyyy in this context

GDE Error: This indicates that GDE did not encounter a valid syntax element. xxxx is the invalid element. yyyy designates whether the element in context is a verb (command), object, or qualifier.

Action: Look for and correct typographical errors.

LITNONGRAPH

LITNONGRAPH, standard requires graphics in string literals; found non-printable: $ZCHAR(cccc)

Compile-time warning: flags a standard violation. The generated code will accept the string, even though it contains cccc, which is not a visible character.

Action: Consider revising the literal to use $[Z]CHAR() and possibly concatenation to make the code more maintainable.

MERGEDESC

MERGEDESC, Merge operation not possible. xxxx is descendent of yyyy.

Run Time Error: This indicates that GT.M was not able to MERGE xxxx into yyyy or vice versa, because xxxx is a descendent of yyyy. When merging global variables specifications included extended references, the MERGE command issues a MERGDESC

error if any part of the source or target tree, as mapped, is a descendant of the other. In MERGE ^|**"x.gld"|a(1)**=^|**"mumps.gld"|^a** there is no error if mumps.gld maps ^a to different database files than those to which x.gld maps ^a(1). A MERGDESC error occurs if any part of ^a as mapped by mumps.gld overlaps any part of ^a(1) as mapped by x.gld.

Action: Modify the routine to avoid MERGE operation between two variables where one is the descendant of the other.

MISSINGDELIM

MISSINGDELIM, Delimiter dddd expected before qqqq vvvv

GDE Error: This indicates that the delimiter dddd (usually dash character in Unix, slash character in VMS) is expected just before vvvv is specified. vvvv is a GDE object or qualifier indicated by qqqq.

Action: Specify the delimiter as indicated.

MURNDWNOVRD

MURNDWNOVRD, OVERRIDE qualifier used with MUPIP RUNDOWN on database file dddd

MUPIP informational message: This message records use of the OVERRIDE qualifier with a MUPIP RUNDOWN command to bypass an an error, which would normally suggest a more appropriate action.

Action: No action required. This message serves primarily to facilitate analysis of database crashes and recovery procedures.

NAMENDBAD

NAMENDBAD, Subscripted name ssss must end with right parenthesis

GDE Error: This indicates that a subscripted name ssss (global name immediately followed by a left parenthesis) was specified without a balancing right parenthesis at the end of the subscripts.

Action: Specify the subscripted name with the appropriate right parenthesis.

NAMGVSUBOFLOW

NAMGVSUBOFLOW, Subscripted name hhhh...tttt is too long to be represented in the database using collation value #nnnn

GDE Error: This indicates that the subscripted name is too big to be represented in the database (exceeds the maximum limits of GT.M for the key size). The message also reports the alternative collation nnnn which was used to arrive at the subscript/key representation inside the database. The head (hhhh) and tail (tttt) of the long subscript is displayed with a ... in the middle.

Action: Specify a shorter subscripted name.

NAMGVSUBSMAX

NAMGVSUBSMAX, Subscripted Name specification nnnn has more than the maximum # of subscripts (mmmm)

GDE Error: This indicates that a name nnnn was specified with more than the maximum allowed number mmmm.

Action: Specify a name within the maximum allowed number of subscripts.

NAMLPARENNOTBEG

NAMLPARENNOTBEG, Subscripted Name specification nnnn needs to have a left parenthesis at the beginning of subscripts

GDE Error: This indicates that a name was specified using : or , or), which indicates a subscripted name, but the left parenthesis was missing.

Action: Specify a name with the appropriate left parenthesis.

NAMNOTSTRSUBS

NAMNOTSTRSUBS, Subscript #nnnn with value vvvv in name specification is not a properly formatted string subscript

GDE Error: This indicates that the nnnn'th subscript in a name did not have a valid string subscript. For example usages like GBL("AB"_) where the _ syntax is used to do concatenation of string subscripts but the right side of the _ operator is missing the string specification.

Action: Specify the name with a properly formatted string subscript.

NAMNUMSUBSOFLOW

NAMNUMSUBSOFLOW, Subscript #nnnn with value vvvv in name specification has a numeric overflow

GDE Error: This indicates that the nnnn'th subscript in a name specification includes a number that is too big to be represented in GT.M.

Action: Specify a subscript with a number that is inside the numeric range supported by GT.M.

NAMONECOLON

NAMONECOLON, Subscripted Name specification nnnn must have at most one colon (range) specification

GDE Error: This indicates that a subscripted name was specified with a range specification using : in the last subscript but more than one colon character was used.

Action: Specify a range with only one colon.

NAMRANGELASTSUB

NAMRANGELASTSUB, Ranges in name specification nnnn are allowed only in the last subscript

GDE Error: This indicates that one or more ranges (using : syntax) were specified in a name somewhere other than the last subscript.

Action: Specify any ranges only in the last subscript of a name.

NAMRANGEORDER

NAMRANGEORDER, Range in name specification nnnn specifies out-of-order subscripts using collation sequence #cccc

GDE Error: This indicates that the range in the name specification is out-of-order. For example yy(10:1) specifies numeric subscripts that are not in order (10 is greater than 1 and so it should have been 1:10 instead). In case of string subscripts, the

collation sequence cccc is used to arrive at the subscript representation in the database and this is what gets compared for the left and right ends of the range to determine if they are in order or not. For example yy("a":"g") is in order in case of collation sequence 0 (ascii ordering) but might not be in order if the name yy has a non-zero collation defined and that collation sorts strings in reverse ascii order.

Action: Specify ranges in order i.e. lower end of the range on the left hand side and the higher end of the range on the right hand side.

NAMRANGEOVERLAP

NAMRANGEOVERLAP, Range in name specifications mmmm and nnnn overlap using collation sequence #cccc

GDE Error: This indicates that the subscripted name specifications mmmm and nnnn belong to the same unsubscripted global name and map to different regions but define ranges that overlap.

Action: Ranges that overlap cannot map to different regions. The only exception is sub-ranges where one range lies completely inside of another. Fix the range specifications to either map to the same region or split the ranges further to avoid overlap.

NAMRPARENNOTEND

NAMRPARENNOTEND, Subscripted Name specification nnnn cannot have anything following the right parenthesis at the end of subscripts

GDE Error: This indicates that a subscripted name was specified where the right parenthesis denoting the end of the subscripts was followed by more characters.

Action: Specify all subscripts of a name inside the left and right parenthesis that immediately follow the unsubscripted name.

NAMSTARSUBSMIX

*NAMSTARSUBSMIX, Name specification nnnn cannot contain * and subscripts at the same time*

GDE Error: This indicates that the name nnnn contains both * and subscripts which is not allowed.

Action: Specify a wildcard (*) or subscripts, but not both.

NAMSTRSUBSCHARG

NAMSTRSUBSCHARG, Subscript #nnnn with value vvvv in name specification specifies a $C/$ZCH with number cccc that is invalid in the current $zchset

GDE Error: This indicates that the nnnn'th subscript in a name specifies a string subscript using $CHAR/$ZCHAR but one of the arguments to this function (potentially in a comma-separated list) is invalid.

Action: An invalid integer argument is one which returns a null string when passed to $CHAR/$ZCHAR with the current $zchset setting. Specify the string subscript with a valid argument to $CHAR/$ZCHAR.

NAMSTRSUBSCHINT

NAMSTRSUBSCHINT, Subscript #nnnn with value vvvv in name specification does not have a positive integer inside $C/$CHAR/$ZCH/$ZCHAR

GDE Error: This indicates that the nnnn'th subscript in a name specifies a string subscript using $CHAR/$ZCHAR but one of the arguments to this function (potentially in a comma-separated list) is not a positive integer.

Action: Specify the string subscript with a positive integer argument to $CHAR/$ZCHAR.

NAMSTRSUBSFUN

NAMSTRSUBSFUN, Subscript #nnnn with value vvvv in name specification uses function other than $C/$CHAR/$ZCH/$ZCHAR

GDE Error: This indicates that the nnnn'th subscript in a name specifies a string subscript using an unsupported function. The only two supported functions are $CHAR or $ZCHAR (long form and short forms).

Action: Specify the string subscript using only the supported functions.

NAMSUBSBAD

NAMSUBSBAD, Subscript #nnnn with value vvvv in name specification is an invalid number or string

GDE Error: This indicates that the nnnn'th subscript in a name specification is neither a valid number or a string.

Action: Specify a valid subscript.

NAMSUBSEMPTY

NAMSUBSEMPTY, Subscript #nnnn is empty in name specification

GDE Error: This indicates that the nnnn'th subscript in a name specification is empty. For example the 2nd subscript in a(1,,3) is empty.

Action: Specify the subscripted name with a non-empty subscript.

NCTCOLLSPGBL

NCTCOLLSPGBL, Database region rrrr contains portion of spanning global ^gggg and so cannot support non-zero numeric collation type

Run Time Error: This indicates that region rrrr contains parts of a global gggg that spans multiple regions according to the current global directory but the directory tree in rrrr indicates gggg has a non-zero numeric collation type.

Action: Spanning globals only support a value of zero for numeric collation (i.e. numbers collate as numbers, not as strings). Access region rrrr using a temporary global directory that maps all names (including gggg) to rrrr, extract the gggg global, KILL it, use $$set^%^GBLDEF to fix the numeric collation to 0, reload the global from the extract, switch back to the regular global directory. An alternative recovery action that does not require extract/load is to change the global directory so gggg is no longer a spanning global or has no mappings into any region that collates numbers using their string value.

NOENDIANCVT

NOENDIANCVT, Unable to convert the endian format of file dddd due to eeee

MUPIP error: One of the requirements for the MUPIP ENDIANCVT command was not met. The problems include: "database format is not the current version", "minor database format is not the current version", "some blocks are not upgraded to

the current version", "kills in progress", "the database is frozen", "a GT.CM server accessing the database", "recovery was interrupted", "database creation in progress", "wc_blocked is set- rundown needed", "the database is corrupted".

Action: Resolve the reported conditions and repeat the command or use the -OVERRIDE qualifier if it is appropriate to bypass the error condition.

NOSOCKETINDEV Δ

NOSOCKETINDEV, There is no socket in the current socket device

Run Time Error: This indicates that either no sockets have been established for the device or that all the sockets attached to the device have been closed prior to the current command.

Action: Review the logic managing the sockets and correct it.

OPENCONN Δ

OPENCONN, Error opening socket connection

Run Time Error: This indicates that the process of opening a socket resulted in a device error.

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

PARFILSPC Δ

PARFILSPC, Parameter: xxxx file specification: yyyy

Run Time Error: This indicates that a JOB command jobparameter xxxx specified an invalid file-specification yyyy. For file specifications of the form "SOCKET:<handle>", this message indicates that "<handle>" is not a valid socket handle in the socket pool.

Action: Review the file-specification for valid syntax based on the operating system. For sockets, verify that the socket handle is in the socket pool.

PROTNOTSUP Δ

PROTNOTSUP, Protocol xxxx not supported

Run Time Error: This indicates that the protocol specified on the CONNECT or LISTEN deviceparameters is not currently supported.

Action: Use TCP/IP domain sockets by specifying TCP for the protocol string or LOCAL (aka UNIX) domain sockets by specifying LOCAL.

RECORDSTAT Δ

RECORDSTAT, gggg: Key cnt: kkkk max subsc len: ssss max rec len: dddd max node len: rrrr

MUPIP information: LOAD and EXTRACT use this to report on some characteristics of the global variables they processed, where gggg is an unsubscripted global name (region name appears in parentheses if gggg spans multiple regions), kkkk is the

number of unique data cells in the array, ssss is the maximum subscripted key length, dddd is the maximum data length and rrrr is the maximum combined length of keys and subscripts.

Actions: Use the information as appropriate.

REMOTEDBNOSPGBL

REMOTEDBNOSPGBL, Database region rrrr contains portion of a spanning global and so cannot point to a remote file

Run Time Error: This indicates that region rrrr of the current global directory contains parts of a spanning global and therefore cannot point to a remote database file.

Action: Fix the global directory file so region rrrr points to a local file or remove the global nodes that span into this region.

REPLINSTMISMTCH

REPLINSTMISMTCH, Process has replication instance file ffff (jnlpool shmid = ssss) open but database dddd is bound to instance file gggg (jnlpool shmid =tttt)

Run Time Error: The process attempted an update on the replicated database dddd associated with the replication instance file ffff and journal pool shared memory id ssss; however, the process has a different replication instance file gggg or journal pool shmid tttt open (On OpenVMS, replication instance file corresponds to active global directory).

Action: A replicated database can only accept updated by processes that have the same replication instance file (defined by the environment variable gtm_repl_instance) open. Ensure the environment variable gtm_repl_instance is defined to be the same for all processes that update the same replicated database file. This error can also occur if the replication instance file was recreated (while processes were still accessing the replication instance). In this case, the name ffff and gggg would be the same but the corresponding journal pool shared memory ids would be different. To recover from this situation, shut down all processes accessing the instance from before and after the instance file recreate. Run an argumentless MUPIP RUNDOWN to clean up the older journal pool tttt and restart the instance. The following applies to OpenVMS only: Verify this database file is in the global directory used by the Source Server and restart the instance. The Source Server (which is the first process to start on a replicated instance) only binds replicated databases from its global directory to the journal pool that it creates. No other replicated database file can be bound with this journal pool.

REPLINSTNOSHM

REPLINSTNOSHM, Database dddd has no active connection to a replication journal pool; please verify that the database is listed in your instance file

Run Time Error: The Source server was started with a repication instance that had this database file listed but later the source server and this particular database file was shut down while other database files in this instance file were still active.

Action: To recover from this, restart the source server.

REPLNOTLS

REPLNOTLS, xxxx requested TLS/SSL communication but the yyyy was either not started with TLSID qualifier or does not support TLS/SSL protocol

MUPIP Error: This indicates that xxxx (Source Side or Receiver Side) requested TLS/SSL communication but the other side was not started with a TLSID qualifier or was pre-V6.1-000 version that does not support TLS/SSL protocol.

Action: If both sides are running with GT.M version >= V6.1-000, then make sure the TLSID qualifier is specified for both the Source and Receiver Server startup commands. If one of the instances involved in the replication is a pre V6.1-000 GT.M version, upgrade it to V6.1-000 to support TLS.

SETITIMERFAILED

SETITIMERFAILED, A setitimer() call returned an error status of ssss

Runtime warning or fatal error: The above error is issued when GT.M fails to schedule or stop a system timer using the setitimer() system call.

Action: Verify the normal state of the OS kernel. Report the entire incident context to your GT.M support channel along with any GT.M operator log messages within the same time frame.

SOCKBIND

SOCKBIND, Error in binding socket

Runtime error: This message indicates a problem binding a socket to a port or file.

Action: Check the associated ENO message for more details.

SOCKETEXIST

SOCKETEXIST, Socket xxxx already exists

Run Time Error: This error is issued:

- On OPEN: ATTACH=xxxx is used to name a newly created socket but the name already exists in the current Socket device's collection.
- On USE: DETACH=xxxx is used to move a socket to the socketpool but the socketpool already has a socket with the specified name. ATTACH=xxxx is used to move a socket from the socketpool but the current Socket device already has a socket with the specified name.

Action: Review the names of the sockets already present on that device and specify a unique name.

SOCKINIT

SOCKINIT, Error initializing socket: (errno == aaaa) xxxx

Run Time Error: This indicates that the process of opening a socket resulted in a device error. xxxx is the text description of the failure for the OS service.

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

SOCKNOTFND

SOCKNOTFND, Socket xxxx not found

Run Time Error: This error is issued:

- On CLOSE when SOCKET=xxxx is used to specify which socket to close.
- On USE:
  - DETACH=xxxx when the specified socket in not in the current Socket device.
  - ATTACH=xxxx when the specified socket is not in the socketpool.
  - SOCKET=xxxx when the specified socket is not in the current Socket device.

Action: Make sure the socket is created before an I/O operation attempts using it.

SOCKPARMREQ

SOCKPARMREQ, Socket device parameter is required for TCP open

Run Time Error: This indicates that a socket deviceparameter was not defined to OPEN a TCP connection.

Action: Establish a TCP connection by specifying the SOCKET deviceparameter for the OPEN command. For a more complete description of this deviceparameter refer to the Input/Output Processing chapter of the Programmer's Guide.

SOCKWAIT

SOCKWAIT, Error waiting for socket connection

Run Time Error: This indicates that the process of waiting for an event on a socket resulted in a device error.

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

SOCKWRITE

SOCKWRITE, Write to a socket failed

Run Time Error: This indicates that GT.M was unable to write to a socket.

Action: Review the accompanying messages for more information on the cause of the failure.

STDNULLCOLLREQ

STDNULLCOLLREQ, Region rrrr needs Standard Null Collation enabled because global gggg spans through it

GDE Information: This indicates that the global gggg spans through region rrrr but the region has GT.M Null collation enabled.

Action: All regions containing parts of spanning globals need to have Standard Null Collation enabled. Fix region rrrr to have Standard Null Collation enabled.

STRMISSQUOTE

STRMISSQUOTE, Missing double-quote at end of string specification ssss

GDE Error: This indicates that a subscripted name was specified with string subscripts parts of which were enclosed inside double-quotes but the closing double-quote is missing.

Action: Specify the subscripted name with the appropriate double quote(s).

TLSCONNINFO

TLSCONNINFO, Failed to obtain information on the TLS/SSL connection

Run Time Warning: This indicates that an attempt to establish TLS/SSL connection failed.

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

TLSCONVSOCK

TLSCONVSOCK, Failed to convert UNIX TCP/IP socket to TLS/SSL aware socket>/error

MUPIP Error: This indicates that an attempt to establish TLS/SSL connection failed.

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

TLSDLLNOOPEN

TLSDLLNOOPEN, Failed to load GT.M TLS/SSL library for secure communication

MUPIP Error: This indicates that the attempt to load the GT.M dynamically linked library for TLS/SSL operation failed.

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

TLSHANDSHAKE

TLSHANDSHAKE, Connection to remote side using TLS/SSL protocol failed

MUPIP Error: This indicates that an attempt to establish SSL/TLS connection failed.

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

TLSINIT

TLSINIT, Failed to initialize GT.M TLS/SSL library for secure communication

MUPIP Error: This indicates that the attempt to initialize a secure context for TLS/SSL operation failed.

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

TLSIOERROR

TLSIOERROR, Error during TLS/SSL oooo operation

MUPIP Error: This indicates that while attempting oooo operation (receive or send), the TLS/SSL library encountered an error.

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

TLSRENEGOTIATE

TLSRENEGOTIATE, Failed to renegotiate TLS/SSL connection

MUPIP Error: This indicates that an attempt to renegotiate the SSL/TLS connection failed.

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

TMPSTOREMAX

TMPSTOREMAX, Maximum space for temporary values exceeded

Compile Time Error: GT.M uses 1024 temporary registers to hold intermediate results of expression evaluation. Because of the line-oriented nature of the M language, these temporary registers get reused for each line. This error indicates that there is a single calculation on a line that requires more than 1024 temporary registers.

Action: Look for excessive function nesting on a very long line of code. Reduce the amount of nesting by assigning intermediate results to a variable. Alternatively, if compiling the routine with the -DYNAMIC_LITERALS qualifier, try compiling with -NODYNAMIC_LITERALS.

TPNOSUPPORT

TPNOSUPPORT, Operation cannot be performed while inside of a TP transaction

Run Time Error: This indicates that a $$set^%GBLDEF or $$kill^%GBLDEF was attempted while inside of a TP transaction (TSTART/TCOMMIT fence).

Action: ^%GBLDEF only supports TP from the $$get entry point; it does not support inclusion of $$set or $$kill entry points within a TP transaction.

TRIGNOSPANGBL

TRIGNOSPANGBL, Triggers cannot be installed/deleted for global name gggg as it spans multiple regions in current global directory

Run Time Error: This indicates that the global gggg spans multiple regions and therefore cannot have triggers installed or deleted in its name.

Action: Triggers are not currently supported for globals that span multiple regions.

VALUEBAD

VALUEBAD, xxxx is not a valid yyyy

GDE Error: This indicates that GDE encountered something other than the valid syntax element it was expecting. xxxx is the invalid element. yyyy is the valid element type.

Action: Specify a valid element. This error occurs if GDE is expecting an element (such as a file-specification, qualifier, or number) but receives a value that does not evaluate to the expected element type.