GT.M

Release Notes

V6.3-012

Empowering
the Financial World **FIS**

Contact Information

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

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

Legal Notice

Revision History

Revision 1.1	04 May 2020	In Platforms, correct the list of supported and supportable platforms and update the instructions for installing the package that contains the libtinfo library.
Revision 1.0	08 April 2020	V6.3-012

Table of Contents

V6.3-012

Overview

V6.3-012 includes a new FULLBLKWRT qualifier for GDE and MUPIP SET, and a LOGDENIAL setting for the restriction facility. Also $SELECT() corrections for extensively nested functions, and a Source Server change that improves GT.M replication resilience in events like network reconfiguration and cluster management. The FIS GT.M team encourages GT.M customers to look for innovative use of replication not only for business continuity but also for other use case such as (not limited to) reporting, automatic recovery, general application development, and minimally invasive remote monitoring.

V6.3-012 introduces the ^%JSWRITE utility routine which converts a GT.M global/local variable or a series of SET @ arguments to a string of JS objects. We believe that the ^%JSWRITE utility routine may interest both database administrators and developers working with modern JavaScript web application development frameworks. For sites where a JavaScript runtime environment is not available, consider using replication with an Instance Bound Global Directory to selectively replicate globals to an instance which has a JavaScript runtime environment and use ^%JSWRITE to convert global structures to JS objects. See (GTM-9244).

V6.3-012 also includes other fixes and enhancements. For more information, refer to the Change History section.

Items marked with ● document new or different capabilities.

Please pay special attention to the items marked with the symbols ● as those document items that have a possible impact on existing code, practice or process.

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

Conventions

This document uses the following conventions:

Flag/Qualifiers	-
Program Names or Functions	upper case. For example, MUPIP BACKUP
Examples	lower case. For example: mupip backup -database ACN,HIST /backup
Reference Number	A reference number is used to track software

	enhancements and support requests. It is enclosed between parentheses ().
Platform Identifier	Where an item affects only specific platforms, the platforms are listed in square brackets, e.g., [AIX]

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

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

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

Effective V6.0-000, GT.M documentation adopted IEC standard Prefixes for binary multiples. This document therefore uses prefixes Ki, Mi and Ti (e.g., 1MiB for 1,048,576 bytes). Over time, we'll update all GT.M documentation to this standard.

⊙ denotes a new feature that requires updating the manuals.

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

⊖ denotes deprecated messages.

△ denotes revised messages.

⊕ denotes added messages.

Platforms

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

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

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

Platform	Supported Versions	Notes
IBM Power Systems AIX	7.1 TL 4, 7.2	Only 64-bit versions of AIX with POWER7 as the minimum required CPU architecture level are Supported. 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.

Platform	Supported Versions	Notes
		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.**
		AIX 7.1 TL 5 is Supportable.
		Only the AIX jfs2 filesystem is Supported. Other filesystems, such as jfs1 are Supportable, but not Supported. FIS strongly recommends use of the jfs2 filesystem on AIX; use jfs1 only for existing databases not yet migrated to a jfs2 filesystem.
x86_64 GNU/Linux	Red Hat Enterprise Linux 7.7 and 7.8; Ubuntu 16.04 LTS, 18.04 LTS, and 19.10	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 (2.6.32 or later), glibc (version 2.12 or later) and ncurses (version 5.7 or later).
		Due to build optimization and library incompatibilities, GT.M versions older than V6.2-000 are incompatible with glibc 2.24 and up. This incompatibility has not been reported by a customer, but was observed on internal test systems that use the latest Linux software distributions from Fedora (26), Debian (unstable), and Ubuntu (17.10). In internal testing, processes either hung or encountered a segmentation violation (SIG-11) during operation. Customers upgrading to Linux distributions that utilize glibc 2.24+ must upgrade their GT.M version at the same time as or before the OS upgrade.
		GT.M requires a compatible version of the libtinfo library. On Red Hat, the ncurses-libs and ncurses-compat-libs packages contain the libtinfo library. On Debian/Ubuntu, libtinfo5 and libncurses5 packages contain the libtinfo library. If any of these packages is not already installed on your system, please install using an appropriate package manager.
		To support the optional WRITE /TLS fifth argument (the ability to provide / override options in the tlsid section of the encryption configuration file), the reference implementation of the encryption plugin requires libconfig 1.4.x.
		Only the ext4 and xfs filesystems are Supported. Other filesystems are Supportable, but not Supported. Furthermore, if you use the NODEFER_ALLOCATE feature, FIS strongly recommends that you use xfs. If you must use NODEFER_ALLOCATE with ext4, you *must* ensure that your kernel includes commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at https://

Platform	Supported Versions	Notes
		www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3). The Red Hat Bugzilla identifier for the bug is 1213487. With NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs. Ubuntu 20.04 is Supportable.
x86 GNU/Linux	Debian 10 (buster)	This 32-bit version of GT.M runs on either 32- or 64-bit x86 platforms; we expect the x86_64 GNU/Linux version of GT.M to be preferable on 64-bit hardware. Running a 32-bit GT.M on a 64-bit GNU/Linux requires 32-bit libraries to be installed. The CPU must have an instruction set equivalent to 586 (Pentium) or better. Please also refer to the notes above on x86_64 GNU/Linux.

Platform support lifecycle

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

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

32- vs. 64-bit platforms

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

* You must compile the application code separately for each platform. Even though the M source code is the same, the generated object modules are different - the object code differs between x86 and x86_64.

* Parameter-types that interface GT.M with non-M code using C calling conventions must match the data-types on their target platforms. Mostly, these parameters are for call-ins, external calls, internationalization (collation) and environment translation, and are listed in the tables below. Note that most addresses on 64-bit platforms are 8 bytes long and require 8 byte alignment in structures whereas all addresses on 32-bit platforms are 4 bytes long and require 4-byte alignment in structures.

Call-ins and External Calls

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

Caution

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

Internationalization (Collation)

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

Important

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

Environment Translation

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

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

Additional Installation Instructions

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

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

* FIS strongly recommends installing each version of GT.M in a separate (new) directory, rather than
  overwriting a previously installed version. If you have a legitimate need to overwrite an existing
  GT.M installation with a new version, you must first shut down all processes using the old version.
  FIS suggests installing GT.M V6.3-012 in a Filesystem Hierarchy Standard compliant location
  such as /usr/lib/fis-gtm/V6.3-012_arch (for example, /usr/lib/fis-gtm/V6.3-012_x86 on 32-bit Linux
  systems). A location such as /opt/fis-gtm/V6.3-012_arch would also be appropriate. Note that the
  arch suffix is especially important if you plan to install 32- and 64-bit versions of the same release of
  GT.M on the same system.

* Use the appropriate MUPIP command (e.g. ROLLBACK, RECOVER, RUNDOWN) of the old GT.M
  version to ensure all database files are cleanly closed.

* Make sure gtmsecshr is not running. If gtmsecshr is running, first stop all GT.M processes including
  the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP *pid_of_gtmsecshr***.

* Starting with V6.2-000, GT.M no longer supports the use of the deprecated $gtm_dbkeys and the
  master key file it points to for database encryption. To convert master files to the libconfig format,
  please click 🔽 to download the CONVDBKEYS.m program and follow instructions in the comments
  near the top of the program file. You can also download CONVDBKEYS.m from http://tinco.pair.com/
  bhaskar/gtm/doc/articles/downloadables/CONVDBKEYS.m. If you are using $gtm_dbkeys for
  database encryption, please convert master key files to libconfig format immediately after upgrading
  to V6.2-000 or later. Also, modify your environment scripts to include the use of gtmcrypt_config
  environment variable.

Recompile

* Recompile all M and C source files.

Rebuild Shared Libraries or Images

* Rebuild all Shared Libraries after recompiling all M and C source files.

* If your application is not using object code shared using GT.M's auto-relink functionality, please consider using it.

Compiling the Reference Implementation Plugin

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

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

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

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

2. Unpack $gtm_dist/plugin/gtmcrypt/source.tar to a temporary directory.

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

3. Follow the instructions in the README.

   * Open Makefile with your editor; review and edit the common header (IFLAGS) and library paths (LIBFLAGS) in the Makefile to reflect those on your system.

   * Define the gtm_dist environment variable to point to the absolute path for the directory where you have GT.M installed

   * Copy and paste the commands from the README to compile and install the encryption plugin with the permissions defined at install time

   ### ⚠ Caution

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

* Download the most recent OpenSSL 1.0.x version

* Compile and install (default installs to /usr/local/ssl)

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

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

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

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

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

* Recompile the encryption plugin following the above directions.

* Remove /usr/local/ssl/lib.donotuse to avoid future complications.

Upgrading to GT.M V6.3-012

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

GT.M upgrade procedure for V6.3-012 consists of 5 stages:

* Stage 1: Global Directory Upgrade

* Stage 2: Database Files Upgrade

* Stage 3: Replication Instance File Upgrade

* Stage 4: Journal Files Upgrade

* Stage 5: Trigger Definitions Upgrade

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V6.3-012 depends on your GT.M upgrade history and your current version.

Stage 1: Global Directory Upgrade

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

To upgrade from any previous version of GT.M:

*   Open your Global Directory with the GDE utility program of GT.M V6.3-012.

*   Execute the EXIT command. This command automatically upgrades the Global Directory.

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

1. Open your Global Directory with the GDE utility program on the 32-bit platform.

2. On GT.M versions that support SHOW -COMMAND, execute SHOW -COMMAND -FILE=file-name. This command stores the current Global Directory settings in the specified file.

3. On GT.M versions that do not support GDE SHOW -COMMAND, execute the SHOW -ALL command. Use the information from the output to create an appropriate command file or use it as a guide to manually enter commands in GDE.

4. Open GDE on the 64-bit platform. If you have a command file from 2. or 3., execute @file-name and then run the EXIT command. These commands automatically create the Global Directory. Otherwise use the GDE output from the old Global Directory and apply the settings in the new environment.

An analogous procedure applies in the reverse direction.

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

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

*   Open the global directory with the GDE utility program of V6.3-012.

*   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

Before starting the database file upgrade, use the prior GT.M version to perform an appropriate MUPIP action to removes abandoned GT.M database semaphores and releases any IPC resources.

To upgrade from GT.M V6*:

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

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

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

A database created with V6 supports up to 992Mi blocks and is not backward compatible. V6 databases that take advantage of V6 limits on key size and records size cannot be downgraded. Use MUPIP DOWNGRADE -VERSION=V5 to downgrade a V6 database back to V5 format provided it meets the database downgrade requirements. For more information on downgrading a database, refer to Downgrading to V5 or V4.

> ### ⚠ Important
>
> A V5 database that has been automatically upgraded to V6 can perform all GT.M V6.3-012 operations. However, that database can only grow to the maximum size of the version in which it was originally created. A database created on V5.0-000 through V5.3-003 has maximum size of 128Mi blocks. A database created on V5.4-000 through V5.5-000 has a maximum size of 224Mi blocks. A database file created with V6.0-000 (or above) can grow up to a maximum of 992Mi blocks. This means that, for example, the maximum size of a V6 database file having 8KiB block size is 7936GiB (8KiB*992Mi).

> ### ⚠ Important
>
> In order to perform a database downgrade you must perform a MUPIP INTEG -NOONLINE. If the duration of the MUPIP INTEG exceeds the time allotted for an upgrade you should rely on a rolling upgrade scheme using replication.

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

*   Execute the MUPIP REORG -UPGRADE command again, or

*   Execute the DSE CHANGE -FILEHEADER -FULLY_UPGRADED=1 command to stop the warnings.

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

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

* A database that was created by a V5 MUPIP CREATE

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

For additional upgrade considerations, refer to Database Compatibility Notes.

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

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

* Upgrade your database files using in-place or traditional database upgrade procedure depending on your situation. For more information on in-place/traditional database upgrade, see Database Migration Technical Bulletin.

* Run the MUPIP REORG -UPGRADE command. This command upgrades all V4 blocks to V5 format.

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

Database Compatibility Notes

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

* Databases created with V5.3-004 through V5.5-000 can grow to a maximum size of 224Mi (234,881,024) blocks. This means, for example, that with an 8KiB block size, the maximum database file size is 1,792GiB; this is effectively the size of a single global variable that has a region to itself and does not itself span regions; a database consists of any number of global variables. A database created with GT.M versions V5.0-000 through V5.3-003 can be upgraded with MUPIP UPGRADE to increase the limit on database file size from 128Mi to 224Mi blocks.

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

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

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

Stage 3: Replication Instance File Upgrade

V6.3-012 does not require new replication instance files if you are upgrading from V5.5-000. However, V6.3-012 requires new replication instance files if you are upgrading from any version prior to V5.5-000. Instructions for creating new replication instance files are in the Database Replication chapter of the GT.M Administration and Operations Guide. Shut down all Receiver Servers on other instances that are to receive updates from this instance, shut down this instance Source Server(s), recreate the instance file, restart the Source Server(s) and then restart any Receiver Server for this instance with the -UPDATERESYNC qualifier.

Note

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

Important

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

Stage 4: Journal Files Upgrade

On every GT.M upgrade:

*   Create a fresh backup of your database.

*   Generate new journal files (without back-links).

Important

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

Stage 5: Trigger Definitions Upgrade

If you are upgrading from V5.4-002A/V5.4-002B/V5.5-000 to V6.3-012 and you have database triggers defined in V6.2-000 or earlier, you need to ensure that your trigger definitions are wholesome in the older version and then run MUPIP TRIGGER -UPGRADE. If you have doubts about the wholesomeness of the trigger definitions in the old version use the instructions below to capture the definitions delete them in the old version (-*), run MUPIP TRIGGER -UPGRADE in V6.3-012 and then reload them as described below.

You need to extract and reload your trigger definitions only if you are upgrading from V5.4-000/ V5.4-000A/V5.4-001 to V6.3-012 or if you find your prior version trigger definitions have problems. For versions V5.4-000/V5.4-000A/V5.4-001 this is necessary because multi-line XECUTEs for triggers require a different internal storage format for triggers which makes triggers created in V5.4-000/ V5.4-000A/V5.4-001 incompatible with V5.4-002/V5.4-002A/V5.4-002B/V5.5-000/V6.0-000/V6.0-001/ V6.3-012.

To extract and reapply the trigger definitions on V6.3-012 using MUPIP TRIGGER:

1. Using the old version, execute a command like **mupip trigger -select="*" trigger_defs.trg**. Now, the output file trigger_defs.trg contains all trigger definitions.

2. Place -* at the beginning of the trigger_defs.trg file to remove the old trigger definitions.

3. Using V6.3-012, run **mupip trigger -triggerfile=trigger_defs.trg** to reload your trigger definitions.

To extract and reload trigger definitions on a V6.3-012 replicating instance using $ZTRIGGER():

1. Shut down the instance using the old version of GT.M.

2. Execute a command like **mumps -run %XCMD 'i $ztrigger("select")' > trigger_defs.trg** . Now, the output file trigger_defs.trg contains all trigger definitions.

3. Turn off replication on all regions.

4. Run **mumps -run %XCMD 'i $ztrigger("item","-*")** to remove the old trigger definitions.

5. Perform the upgrade procedure applicable for V6.3-012.

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 in the database header, that is, Maximum-Key-Size is assured.

5. The maximum Record size is small enough to accommodate key, overhead, and value within a block.

To verify that your database meets all of the above requirements, execute MUPIP INTEG -NOONLINE. Note that the integrity check requires the use of -NOONLINE to ensure no concurrent updates invalidate the above requirements. Once assured that your database meets all the above requirements, MUPIP DOWNGRADE -VERSION=V5 resets the database header to V5 elements which makes it compatible with V5 versions.

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

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

1. In a GT.M V6.3-012 environment:

    a. Execute MUPIP SET -VERSION=v4 so that GT.M writes updates blocks in V4 format.

    b. Execute MUPIP REORG -DOWNGRADE to convert all blocks from V6 format to V4 format.

2. Bring down all V6 GT.M processes and execute MUPIP RUNDOWN -FILE on each database file to ensure that there are no processes accessing the database files.

3. Execute MUPIP DOWNGRADE -VERSION=V4 to change the database file header from V6 to V4.

4. Restore or recreate all the V4 global directory files.

5. Your database is now successfully downgraded to V4.

Managing M mode and UTF-8 mode

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

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

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

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

*   Actual files for GT.M executable programs (mumps, mupip, dse, lke, and so on) are in the parent directory, that is, the location specified for installation.

*   Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for UTF-8 mode in the utf8 subdirectory, and one compiled without support for UTF-8 mode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install UTF-8 mode support. Note that on 64-bit versions of GT.M, the object code is in shared libraries, rather than individual files in the directory.

*   The utf8 subdirectory has files called mumps, mupip, dse, lke, and so on, which are relative symbolic links to the executables in the parent directory (for example, mumps is the symbolic link ../mumps).

*   When a shell process sources the file gtmprofile, the behavior is as follows:

    *   If $gtm_chset is "m", "M" or undefined, there is no change from the previous GT.M versions to the value of the environment variable $gtmroutines.

    *   If $gtm_chset is "UTF-8" (the check is case-insensitive),

        * $gtm_dist is set to the utf8 subdirectory (that is, if GT.M is installed in /usr/lib/fis-gtm/ gtm_V6.3-012_i686, then gtmprofile sets $gtm_dist to /usr/lib/fis-gtm/gtm_V6.3-012_i686/utf8).

        *   On platforms where the object files have not been placed in a libgtmutil.so shared library, the last element of $gtmroutines is $gtm_dist($gtm_dist/..) so that the source files in the

parent directory for utility programs are matched with object files in the utf8 subdirectory. On platforms where the object files are in libgtmutil.so, that shared library is the one with the object files compiled in the mode for the process.

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

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

Setting the environment variable TERM

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

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

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

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

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

By default, GT.M searches for the libz.so shared library in the standard system library directories (for example, /usr/lib, /usr/local/lib, /usr/local/lib64). If the shared library is installed in a non-standard location, before starting replication, you must ensure that the environment variable LIBPATH (AIX)

or LD_LIBRARY_PATH (GNU/Linux) includes the directory containing the library. The Source and Receiver Server link the shared library at runtime. If this fails for any reason (such as file not found, or insufficient authorization), the replication logic logs a DLLNOOPEN error and continues with no compression.

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

Change History

V6.3-012

Fixes and enhancements specific to V6.3-012:

Id	Prior Id	Category	Summary
GTM-5381	C9E04-002554	Admin	⚫ New -FULLBLKWRT qualifier for MUPIP SET and GDE ✅
GTM-7759	-	Admin	Logging when GT.M denies access to a process ✅
GTM-9136	-	Language	Optional standard behavior for WRITE # ✅
GTM-9157	-	Admin	Source Server gives more informative and persists when name resolution fails
GTM-9182	-	Admin	⚫ Appropriate edit checking on path/file length for MUPIP BACKUP
GTM-9238	-	Other	Handle $ZSTPPLLIM settings and STPCRIT errors so as to avoid apparent loss of heap space
GTM-9244	-	Other	%JSWRITE utility routine for using GT.M glvn strctures in a JavaScript runtime environment ✅
GTM-9247	-	Language	$SELECT() handles truely tortured argment lists
GTM-9248	-	Other	Correct "generated from" in messages
GTM-9249	-	Admin	Receiver server handles cross-endian decompression errors appropriately
GTM-9255	-	Other	Prevent harmless segmentation violation (SIG-11) at process exit when using qdbrundown
GTM-9260	-	Admin	MUPIP RUNDOWN removes abandoned memory associated with LOCK_SPACE
GTM-9269	-	Language	The -NOWARNING compilation qualifier applies to DONOBLOCK warnings
GTM-9270	-	Language	Improve alignment of $HOROLOG, $ZHORLOG, and $ZUT

Id	Prior Id	Category	Summary
GTM-9273	-	Other	dbcertify no longer maintained for this and future versions

Language

* The [NO]FFLF deviceparameter controls whether WRITE # produces only a form-feed (<FF>) or a form-feed and line-feed (<FF><LF>). Previously, GT.M used <FF><LF> which deviated from the standard, however, out of concern for existing practice the default remains <FF><LF> Additionally, the "gtm_nofflf" environment variable controls the default WRITE # behavior of GT.M. If it is unset or set to 0, N[O] or F[ALSE], the default behavior is unchanged. If it is set to 1, Y[ES] or T[RUE], the default behavior of WRITE # is changed to produce only a form-feed (<FF>), though M programs can still control behavior by specifying the FFLF deviceparameter. (GTM-9136)

* $SELECT() correctly handles a range of arguments that include literals that GT.M optimizes at compile time and extensive nesting of the function. Previously, $SELECT() with some argument patterns failed, typically with a GT.M fatal error, but in a few cases with an incorrect result. (GTM-9247)

* The -NOWARNING compilation qualifier suppresses DONOBLOCK warnings; previously it did not. (GTM-9269)

* $HOROLOG uses the same system service as $ZUT. Previously, when $HOROLOG was fetched first and scaled to microseconds, it could be less than a subsequently fetched $ZUT. (GTM-9270)

System Administration

* GDE and MUPIP SET recognize the FULLBLKWRT database segment/file characteristic to determine whether GT.M writes only valid database blocks contents, or a full block including meaningless trailing content. Full block writes are more efficient with some secondary storage because they avoid read-before-write. The format of the FULLBLKWRT qualifier is:

```
-FU[LLBLKWRT]={0|1|2}
```

* When -FULLBLKWRT=2, a process writes all newly allocated database blocks in their entirety regardless of their actual valid contents. This relieves some file systems from tracking as much unallocated space and thus reduces file system metadata maintenance.

* When -FULLBLKWRT=1, a process writes entire file system blocks in their entirety regardless of their actual valid contents, on some file systems, this avoids reading in advance of most writes and thus reduces file system load and increases response time.

* When -FULLBLKWRT=0 (the default), a process writes only valid data.

Note that when the file system block size and the database block size are the same there is no difference between the settings of 1 and 2.

Previously the gtm_fullblockwrites environment variable of the first process to access the database (or its absence) determined this characteristic. With this change, GT.M no longer recongizes the enviroment variable gtm_fullblockwrites. (GTM-5381) ⬤ ⬤

* By default GT.M uses the the syslog() facility to log a number of errors related to permissions and access. Previously, it only noted these messages on the console or, if output was redirected, in a GT.M process's error output. The GT.M restriction LOGDENIALS provides a facility for disabling this logging on a Unix group basis. If the restriction mechanism is not used, the logging takes place for all GT.M processes. If the restriction is used logging takes place for specified groups only. To support restriction based configuration of LOGDENIALS, the GT.M restriction handling code now supports group names using the POSIX Portable Filename Character Set:

```
      3.282 Portable Filename Character Set

      The set of characters from which portable filenames are constructed.

      A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
      a b c d e f g h i j k l m n o p q r s t u v w x y z
      0 1 2 3 4 5 6 7 8 9 . _ -
```

In practice, this means the characters "-" and "." are now accepted by GT.M in group names specified in the restrict.txt file whereas previously they were not. (GTM-7759) ⬤

* When the Source Server fails to resolve a secondary host name to an IP address, it reports the secondary host name with the GETADDRINFO message in the Source Server log file and continues

---

attempts to resolve the host name to an IP address as per the -CONNECTPARAMS settings. Previously, the Source Server did not display the host name in the GETADDRINFO message and terminated the Source Server process. This change increases the replication resilience in events like network reconfiguration and cluster management. (GTM-9157)

* MUPIP BACKUP issues FILENAMETOOLONG when the backup file path size exceeds the maximum allowed by GTM, 255 bytes. Previously, MUPIP BACKUP could at least partially execute in spite of file paths that exceeded the maximum length allowed by GTM. (GTM-9182) ⬤

* Receiver server connected to a cross endian Source server handles decompression errors appropriately. Previously when there was an issue with zlib uncompress, and the source server has the opposite endian, the receiver server would exit with a KILLBYSIGSINFO1 error. This was only encountered in the GT.M development environment, and was never reported by a user. (GTM-9249)

* MUPIP RUNDOWN removes shared memory associated with LOCK hashing when removing database shared memory, as needed. Previously, in the rare case that GT.M created an additional shared memory segment to hold the LOCK hash table and the database required a manual MUPIP RUNDOWN/RECOVER/ROLLBACK to run down the database, MUPIP did not remove the additional shared memory segment. (GTM-9260)

Other

*   GT.M manages detection of STPCRIT errors to minimize the chance that repeats of those errors indicate decreasing amounts of available space; previously repeated STPCRIT errors inappropriately indicated a loss of string pool space. Also, an attempt to set the limit lower than the miminum string pool size adds the requested limit to the minimum size; previously, GT.M silently ignored such attempts. (GTM-9238)

*   The ^%JSWRITE utility routine converts a GT.M global/local variable (glvn) or a series of SET @ arguments to a string of JS objects. We believe that the ^%JSWRITE utility routine may interest both database administrators and JavaScript programmers alike.

    There are several ^%JSWRITE use cases possible when there is a need to use a GT.M glvn structure in a JavaScript environment. For JS component development with contemporary JavaScript web development frameworks, ^%JSWRITE converts a GT.M glvn structures to JS objects which you can use as properties. On production sites where a JavaScript runtime environment is not available or cannot be installed, consider using an Instance Bound Global Directory to replicate only the relevant globals to another site which has a JavaScript runtime environment and use the %JSWRITE utility routine to transform those globals to JS objects. For example, you can setup an Instance Bound Global Directory to replicate a global that contains only the relevant statistics (TR0, TR1, TR2, TR3, and TR4) from the %YGBLSTAT utility to a remote instance which has node.js and the visualization tools to display TP restart trends in a graph.

    For information on usage and syntax of the ^%JSWRITE utility routine, refer to the Additional Information section.

    For ^%JSWRITE reference implementations and examples of ^%JSWRITE, refer to the fis-gtm-jswrite.tar.gz npm package.(GTM-9244)

*   GT.M includes the correct value for "generated from" in system log messages. Since V60002 the value indicated the wrong location. (GTM-9248)

*   GT.M manages the release of the rendezvous (ftok) semaphore appropriately on process exit when using qdbrundown; previously when processes bypassed certain logic, they could encounter a SIG-11, which had no additional impact. (GTM-9255)

*   The dbcertify software to upgrade V4 GT.M databases to V5/V6 is deprecated and no longer shipped as a separate package. Please use a prior GT.M version to upgrade a V4 database. (GTM-9273)

Additional Information

Additional Information about GTM-9244 - %JSWRITE Utility Routine

The ^%JSWRITE utility routine converts a glvn structure or a series of SET @ arguments to a string of JS objects. The format of the ^%JSWRITE utility is:

```
^%JSWRITE(glvnode,[expr1,expr2])
```

* glvnode specifies the string containing the subscripted/unsubscripted global or local variable name. When glvnode evaluates to an empty string ("") or there are no arguments, %JSWRITE considers all subscripted local variables in scope for conversion.

* If expr1 specifies "#", ^%JSWRITE displays JS objects of the entire tree starting from the glvnode till the end of the glvn.

* If expr1 specifies "*", ^%JSWRITE displays JS objects for all nodes descending from the specified glvn node.

* Specifying "*" and "#" together produces an ILLEGALEXPR2 error.

* Specifying [expr1], that is, with a leading "["and trailing "]", ^%JSWRITE displays the JSON objects in an array collection. Without [], you need to transform the object strings to the desired destination object format.

* If expr2 specifies "noglvname", ^%JSWRITE excludes the first key containing the name of the glvn root from the JS object output.

* The default $ETRAP for %JSWRITE is **if (""=$etrap) new $etrap set $etrap="do errorobj"_"^"_ $text(+0),$zstatus="".** To override the default error handler, set $ETRAP to any non-empty value.

* GT.M is not a JavaScript runtime environment. Therefore, we recommend parsing all output of ^ %JSWRITE either using a JSON parser such as JSON.parse() in an appropriate JavaScript run-time environment, a web server via setting its response header to **'Content-Type:application/json'**, or an application where JSON parsing is available.

* When appropriate, enclose invocations of ^%JSWRITE in a TSTART/COMMIT boundary to prevent any blurred copy of data that is actively updated.

* When appropriate, use GT.M alias containers to take appropriate local variables temporarily out of scope and then run the argumentless form of ^%JSWRITE.

Examples:

```
# Demo lv
GTM>zwrite demodevtest
demodevtest("Developer1","Token1","testSetup")="runtest holt maintest cpipe"
demodevtest("Developer1","Token1","testSetup","65401,11987")=1
demodevtest("Developer1","Token1","testSetup","holt","t")="mtest"
demodevtest("Developer1","Token1","testSetup","holt","t","SendReport",65401,12073)=1
```

```
demodevtest("Developer1","Token1","testSetup","holt","t","cpipe",65401,12025)=0
demodevtest("Developer1","Token2","testSetup")="runtest holt maintest tconv"
demodevtest("Developer1","Token2","testSetup","holt","65401,21987")=1
demodevtest("Developer1","Token2","testSetup","holt","t")="mtest"
demodevtest("Developer1","Token2","testSetup","holt","t","SendReport",65401,22073)=1
demodevtest("Developer1","Token2","testSetup","holt","t","tconv",65401,22025)=0
demodevtest("Developer2","Token3","testSetup")="runtest holt maintest tconv"
demodevtest("Developer2","Token3","testSetup","holt","65401,21987")=1
demodevtest("Developer2","Token3","testSetup","holt","t")="mtest"
demodevtest("Developer2","Token3","testSetup","holt","t","SendReport",65401,22073)=1
demodevtest("Developer2","Token3","testSetup","holt","t","tconv",65401,22025)=0
demodevtest("Developer3","Token4","testSetup")="runtest holt maintest tconv"
demodevtest("Developer3","Token4","testSetup","holt","65401,31987")=1
demodevtest("Developer3","Token4","testSetup","holt","t")="mtest"
demodevtest("Developer3","Token4","testSetup","holt","t","SendReport",65401,32073)=1
demodevtest("Developer3","Token4","testSetup","holt","t","tconv",65401,32025)=0

GTM>set glvn="demodevtest(""Developer2"")"

GTM>do ^%JSWRITE(glvn,"*") ; JS Object Strings: All descendants of demodevtest("Developer2")
{"demodevtest":{"Developer2":{"Token3":{"testSetup":"runtest holt maintest tconv"}}}}
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"65401,21987":1}}}}}}
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":"mtest"}}}}}}
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"22073":1}}}}}}}}}
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"22025":0}}}}}}}}}

GTM>do ^%JSWRITE(glvn,"[*]") ; Array: All descendants of demodevtest("Developer2")
[{"demodevtest":{"Developer2":{"Token3":{"testSetup":"runtest holt maintest tconv"}}}},
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"65401,21987":1}}}}}},
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":"mtest"}}}}}},
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"22073":1}}}}}}}}},
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"22025":0}}}}}}}}}]

GTM> do ^%JSWRITE(glvn,"#") ; JS Object Strings: All descendants of demodevtest starting from
 demodevtest("Developer2")
{"demodevtest":{"Developer2":{"Token3":{"testSetup":"runtest holt maintest tconv"}}}}
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"65401,21987":1}}}}}}
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":"mtest"}}}}}}
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"22073":1}}}}}}}}}
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"22025":0}}}}}}}}}
{"demodevtest":{"Developer3":{"Token4":{"testSetup":"runtest holt maintest tconv"}}}}
{"demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"65401,31987":1}}}}}}
{"demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":"mtest"}}}}}}
{"demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"32073":1}}}}}}}}}
```

```
{"demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"32025":0}}}}}}}}}

GTM>do ^%JSWRITE(glvn,"[#]") ; Array: All descendants of demodevtest starting from
 demodevtest("Developer2")
[{"demodevtest":{"Developer2":{"Token3":{"testSetup":"runtest holt maintest tconv"}}}},
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"65401,21987":1}}}}}},
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":"mtest"}}}}}},
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"22073":1}}}}}}}}},
{"demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"22025":0}}}}}}}}},
{"demodevtest":{"Developer3":{"Token4":{"testSetup":"runtest holt maintest tconv"}}}},
{"demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"65401,31987":1}}}}}},
{"demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":"mtest"}}}}}},
{"demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"32073":1}}}}}}}}},
{"demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"32025":0}}}}}}}}}]

$ $gtm_dist/mumps -r %XCMD 'ZWRITE ^demodevtest' | $gtm_dist/mumps -r STDIN^%JSWRITE
[{"^demodevtest":{"Developer1":{"Token1":{"testSetup":"runtest holt maintest cpipe"}}}},
{"^demodevtest":{"Developer1":{"Token1":{"testSetup":{"65401,11987":1}}}}},
{"^demodevtest":{"Developer1":{"Token1":{"testSetup":{"holt":{"t":"mtest"}}}}}},
{"^demodevtest":{"Developer1":{"Token1":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"12073":1}}}}}}}}},
{"^demodevtest":{"Developer1":{"Token1":{"testSetup":{"holt":{"t":{"cpipe":{"65401":
{"12025":0}}}}}}}}},
{"^demodevtest":{"Developer1":{"Token2":{"testSetup":"runtest holt maintest tconv"}}}},
{"^demodevtest":{"Developer1":{"Token2":{"testSetup":{"holt":{"65401,21987":1}}}}}},
{"^demodevtest":{"Developer1":{"Token2":{"testSetup":{"holt":{"t":"mtest"}}}}}},
{"^demodevtest":{"Developer1":{"Token2":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"22073":1}}}}}}}}},
{"^demodevtest":{"Developer1":{"Token2":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"22025":0}}}}}}}}},
{"^demodevtest":{"Developer2":{"Token3":{"testSetup":"runtest holt maintest tconv"}}}},
{"^demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"65401,21987":1}}}}}},
{"^demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":"mtest"}}}}}},
{"^demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"22073":1}}}}}}}}},
{"^demodevtest":{"Developer2":{"Token3":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"22025":0}}}}}}}}},
{"^demodevtest":{"Developer3":{"Token4":{"testSetup":"runtest holt maintest tconv"}}}},
{"^demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"65401,31987":1}}}}}},
{"^demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":"mtest"}}}}}},
{"^demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":{"SendReport":{"65401":
{"32073":1}}}}}}}}},
{"^demodevtest":{"Developer3":{"Token4":{"testSetup":{"holt":{"t":{"tconv":{"65401":
{"32025":0}}}}}}}}}]
```

Utility Label:

STDIN^%JSWRITE [singlesub]

With the STDIN, the %JSWRITE utility routine expects a valid SET @ argument (like the one from the ZWRITE command) as its standard input over a named/unnamed pipe device and returns an array of objects. This construct ensures that $ZUSEDSTOR remains consistently low even for processing large data for conversion. STDIN^%JSWRITE automatically terminates the process with a non-zero exit status when it does not receive a READ terminator for 120 seconds from standard input.

When "singlesub" is specified as an argument, ^%JSWRITE expects ZWRITE lines for single subscript glvns. Here ^%JSWRITE implicitly removes the unsubscripted glvn name and returns an array collection of objects in the form of **[{"key1":value,"key2":value,...},{"key1":value,"key2":value,...}]** where:

*   **key1** is the subscript

*   **value** is the right side of the =.

The subscript first received by STDIN^%JSWRITE singlesub denotes the start of the object. When ^%JSWRITE finds the same subscript, it ends the current object boundary and starts the boundary of a new object.

Example:

```
$ $gtm_dist/mumps -r ^RTN
abc("firstname")="John"
abc("lastname")="Doe"
abc("firstname")="Jane"
abc("lastname")="Doe"

$ $gtm_dist/mumps -r ^RTN | $gtm_dist/mumps -r STDIN^%JSWRITE singlesub
[{"firstname":"John","lastname":"Doe"},
{"firstname":"Jane","lastname":"Doe"}]
```

fis-gtm-jswrite.tar.gz

Overview

fis-gtm-jswrite.tar.gz is an npm package containing four ^%JSWRITE reference implementations and three examples for the ^%JSWRITE utility routine. It also includes a utility class called JSWRITE.js to help process the output of the ^%JSWRITE utility routine for use in a JavaScript runtime environment and convert JavaScript objects to SET @ arguments.

The fis-gtm-jswrite.tar.gz package is available only for supported GT.M customers from the FIS Client Portal (https://my.fisglobal.com/products/gtm). Being a supported GT.M customer entitles you to receive package patches and upgrades under the terms of your support agreement. Note that the license for the fis-gtm-jswrite.tar.gz does not permit redistribution. You may use the fis-gtm-jswrite.tar.gz package as-is or modify as appropriate to suit your needs. In both the cases, you must adequately test the reference implementations and the utility class before using them in a production environment. Please contact gtmsupport@fisglobal.com or your support channel for more information on obtaining this package.

The ^%JSWRITE reference implementations are:

1. Dynamic Journal File Progress Bar: GTMJSJNLCHAIN.m/gtmjsjnlchain.js

2. Global Buffers Dashboard: GTMJSACCESSBG.m/gtmjsaccessbg.js

3. Journal File Chain Report: GTMJSJNLCHAIN.m/gtmjsdbjnl.js

4. %YGBLSTATS Sparkline Chart and %YGBLSTATS to JSON: GTMJSGVSTATS.m/gtmjsgvstats.js

The examples are:

1. gtmjstree.js (renders ^sampletree in the form of a tree)

2. gtmjs-sql1.js (a simple example)

3. gtmjs-sql2.js (a simple example)

4. gtmjs-to-gtm.js (example of JSWRITE.js helper class)

Installation

Step 1: Install node.js and npm

```
curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
sudo apt-get install -y nodejs
curl -L https://www.npmjs.com/install.sh | sh
```

Step 2: Install the fis-gtm-jswrite npm package

```
cd $project_dir
```

```
npm install /path/to/fis-gtm-jswrite.tar.gz # This command installs the @fis-gtm/jswrite
 package and the relevant %JSWRITE reference implementation dependencies.
npm audit --fix
```

Step 3: Run the %JSWRITE reference implementations

```
# The following reference implementations use the "STDIN^%JSWRITE singlesub" entry point and
 require the gtmposix plugin.
export gtmroutines="node_modules/@fis-gtm/jswrite/reference_implementations $gtmroutines"
  # include the reference_implementations directory in the search list
node node_modules/@fis-gtm/jswrite/reference_implementations/gtmjsjnlchain.js  # journal file
 chain status sheet
node node_modules/@fis-gtm/jswrite/reference_implementations/gtmjsaccessbg.js  # %dirty
 buffer dashboard
node node_modules/@fis-gtm/jswrite/reference_implementations/gtmjsdbjnl.js     # displays the
 size of the database and journal files
# Set the environment variable LC_ALL to a UTF8 locale (needed for the %JSWRITE Sparkline
 Chart (GTMJSGVSTATS.m/gtmjsgvstats.js) reference implementation)
node node_modules/@fis-gtm/jswrite/reference_implementations/gtmjsgvstats.js   # By default,
 this reference implementation displays a sparkline chart for SETs and GETs. For monitoring
 other statistics, open gtmjsgvstats.js and update the STAT array to include the list of
 statistics which require sparkline chart style monitoring.
```

Step 4: Run the examples

Create a sandbox environment for GT.M and run the following commands:

```
$gtm_dist/mupip set -key_size=1019 -region "*"
$gtm_dist/mupip load node_modules/@fis-gtm/jswrite/examples/sample.zwr
$gtm_dist/mupip load node_modules/@fis-gtm/jswrite/examples/sampletree.zwr
export "gtmroutines=node_modules/@fis-gtm/jswrite/examples/ $gtmroutines"

# The following example uses the do ^%JSWRITE("^sample","[*]") entry point. This uses the
 lodash library to merge the nodes coming from %JSWRITE.
npm install lodash@latest
node node_modules/@fis-gtm/jswrite/examples/gtmjstree.js     # loads sample.zwr in the tree
 format

# The following two examples use the STDIN^%JSWRITE entry point of the %JSWRITE utility
 routine and then use the getter methods of the JSWRITE.js helper class to create the SQL
 queries.
npm install alasql@latest
node node_modules/@fis-gtm/jswrite/examples/gtmjs-sql1.js

npm install sqlite3@latest
node node_modules/@fis-gtm/jswrite/examples/gtmjs-sql2.js

# The following example uses the @fis-gtm/jswrite helper class (JSWRITE.js) to send data back
 to GT.M in SET @ argument form using shelljs.
npm install shelljs@latest
node node_modules/@fis-gtm/jswrite/examples/gtmjs-to-gtm.js
```

Dynamic Journal File Progress Bar

The dynamic journal file progress bar displays the size progress of a journal file till it reaches the autoswitch limit. It also displays an estimation of time and speed at which a journal file reaches its autoswitch limit. This can be use for benchmarking purposes or as a tool for GT.M performance tuning.

This %JSWRITE reference implementation uses the STDIN^%JSWRITE singlesub entrypoint of the %JSWRITE utility routine.

Example:

```
$ node node_modules/@fis-gtm/jswrite/reference_implementations/gtmjsjnlprogress.js
Press <Ctrl> + c to terminate. refreshInterval is 3 seconds
DEFAULT [===-----------------------------------] 7% | Size: 303.7 MiB | Autoswitch: 4095
 MiB | ETA: 5m40s | Speed: 15.233 MiB/s
REGIONA [========----------------------------] 21% | Size: 876 MiB | Autoswitch: 4095 MiB
 | ETA: 5m15s | Speed: 14 MiB/s
REGIONB [========----------------------------] 23% | Size: 962 MiB | Autoswitch: 4095 MiB
 | ETA: 4m40s | Speed: 15 MiB/s
REGIONC [========----------------------------] 21% | Size: 876.12 MiB | Autoswitch: 4095
 MiB | ETA: 5m15s | Speed: 14.04 MiB/s
```

This %JSWRITE reference implementation monitors the size of the journal files and displays an estimate of time and speed for reaching the autoswitch limit.

Journal File Chain Report

The journal file chain report display the journal file chain starting from the current journal file for reach region. This reference implementation also displays the "out-of-chain" journal files, that is, those journal files which are broken and do not participate in journal recovery in normal circumstances. These out-of-chain journal files can be removed at the discretion of the GT.M database administrator to make space available.

This %JSWRITE reference implementation uses the STDIN^%JSWRITE singlesub entrypoint of the %JSWRITE utility routine.

Example:

```
$ node node_modules/@fis-gtm/jswrite/reference_implementations/gtmjsjnlchain.js
# Journal File Chain Report
| Region    | Journal File                      | Size (MiB) | Switch Date         |
 Back link                      | Chain status |
| :-------- | :------------------------------- | ---------: | :------------------ | :-
------------------------- | ------------ |
| GTMWIZARDS | /path/to/gtmwizards.mjl           |       0.07 | current             |
 gtmwizards.mjl_2020063121042 | In chain     |
| GTMWIZARDS | /path/to/gtmwizards.mjl_2020063121042 |       0.07 | 03-MAR-2020 12:10:42 |
 gtmwizards.mjl_2020063121035 | In chain     |
| GTMWIZARDS | /path/to/gtmwizards.mjl_2020063121035 |       0.07 | 03-MAR-2020 12:10:35 | -
                        | In chain     |
```

```
| REGIONA    | /path/to/regiona.mjl                    |       0.07 | current                  |
 regiona.mjl_2020063121042    | In chain      |
| REGIONA    | /path/to/regiona.mjl_2020063121042      |       0.07 | 03-MAR-2020 12:10:42 |
 regiona.mjl_2020063121035    | In chain      |
| REGIONA    | /path/to/regiona.mjl_2020063121035      |       0.07 | 03-MAR-2020 12:10:35 | -
                       | In chain      |
| REGIONB    | /path/to/regionb.mjl                    |       0.07 | current                  |
 regionb.mjl_2020063121042    | In chain      |
| REGIONB    | /path/to/regionb.mjl_2020063121042      |       0.07 | 03-MAR-2020 12:10:42 |
 regionb.mjl_2020063121035    | In chain      |
| REGIONB    | /path/to/regionb.mjl_2020063121035      |       0.07 | 03-MAR-2020 12:10:35 | -
                       | In chain      |
| REGIONC    | /path/to/regionc.mjl                    |       0.07 | current                  |
 regionc.mjl_2020063121042    | In chain      |
| REGIONC    | /path/to/regionc.mjl_2020063121042      |       0.07 | 03-MAR-2020 12:10:42 |
 regionc.mjl_2020063121035    | In chain      |
| REGIONC    | /path/to/regionc.mjl_2020063121035      |       0.07 | 03-MAR-2020 12:10:35 | -
                       | In chain      |
| N/A        | /path/to/gtmwizards.mjl_2020063120753 |       0.07 | 03-MAR-2020 12:07:53 | -
                       | Out of chain |
| N/A        | /path/to/gtmwizards.mjl_2020063120800 |       0.07 | 03-MAR-2020 12:08:00 | -
                       | Out of chain |
| N/A        | /path/to/gtmwizards.mjl_2020063120801 |       0.07 | 03-MAR-2020 12:08:01 |
 gtmwizards.mjl_2020063120800 | Out of chain |
| N/A        | /path/to/gtmwizards.mjl_2020063120812 |       0.07 | 03-MAR-2020 12:08:12 |
 gtmwizards.mjl_2020063120801 | Out of chain |
| N/A        | /path/to/gtmwizards.mjl_2020063120822 |       0.07 | 03-MAR-2020 12:08:22 |
 gtmwizards.mjl_2020063120812 | Out of chain |
| N/A        | /path/to/gtmwizards.mjl_2020063121026 |       0.07 | 03-MAR-2020 12:10:26 |
 gtmwizards.mjl_2020063120822 | Out of chain |
| N/A        | /path/to/regiona.mjl_2020063120753      |       0.07 | 03-MAR-2020 12:07:53 | -
                       | Out of chain |
| N/A        | /path/to/regiona.mjl_2020063120800      |       0.07 | 03-MAR-2020 12:08:00 | -
                       | Out of chain |
| N/A        | /path/to/regiona.mjl_2020063120801      |       0.07 | 03-MAR-2020 12:08:01 |
 regiona.mjl_2020063120800    | Out of chain |
| N/A        | /path/to/regiona.mjl_2020063120812      |       0.07 | 03-MAR-2020 12:08:12 |
 regiona.mjl_2020063120801    | Out of chain |
| N/A        | /path/to/regiona.mjl_2020063120822      |       0.07 | 03-MAR-2020 12:08:22 |
 regiona.mjl_2020063120812    | Out of chain |
| N/A        | /path/to/regiona.mjl_2020063121026      |       0.07 | 03-MAR-2020 12:10:26 |
 regiona.mjl_2020063120822    | Out of chain |
| N/A        | /path/to/regionb.mjl_2020063120753      |       0.07 | 03-MAR-2020 12:07:53 | -
                       | Out of chain |
| N/A        | /path/to/regionb.mjl_2020063120800      |       0.07 | 03-MAR-2020 12:08:00 | -
                       | Out of chain |
| N/A        | /path/to/regionb.mjl_2020063120801      |       0.07 | 03-MAR-2020 12:08:01 |
 regionb.mjl_2020063120800    | Out of chain |
| N/A        | /path/to/regionb.mjl_2020063120812      |       0.07 | 03-MAR-2020 12:08:12 |
 regionb.mjl_2020063120801    | Out of chain |
```

```
| N/A         | /path/to/regionb.mjl_2020063120822    |       0.07 | 03-MAR-2020 12:08:22 |
 regionb.mjl_2020063120812    | Out of chain |
| N/A         | /path/to/regionb.mjl_2020063121026    |       0.07 | 03-MAR-2020 12:10:26 |
 regionb.mjl_2020063120822    | Out of chain |
| N/A         | /path/to/regionc.mjl_2020063120753    |       0.07 | 03-MAR-2020 12:07:53 | -
                        | Out of chain |
| N/A         | /path/to/regionc.mjl_2020063120800    |       0.07 | 03-MAR-2020 12:08:00 | -
                        | Out of chain |
| N/A         | /path/to/regionc.mjl_2020063120801    |       0.07 | 03-MAR-2020 12:08:01 |
 regionc.mjl_2020063120800    | Out of chain |
| N/A         | /path/to/regionc.mjl_2020063120812    |       0.07 | 03-MAR-2020 12:08:12 |
 regionc.mjl_2020063120801    | Out of chain |
| N/A         | /path/to/regionc.mjl_2020063120822    |       0.07 | 03-MAR-2020 12:08:22 |
 regionc.mjl_2020063120812    | Out of chain |
| N/A         | /path/to/regionc.mjl_2020063121026    |       0.07 | 03-MAR-2020 12:10:26 |
 regionc.mjl_2020063120822    | Out of chain |


# Out of chain journal files:
 /path/to/gtmwizards.mjl_2020063120753 /path/to/gtmwizards.mjl_2020063120800 /
path/to/gtmwizards.mjl_2020063120801 /path/to/gtmwizards.mjl_2020063120812 /
path/to/gtmwizards.mjl_2020063120822 /path/to/gtmwizards.mjl_2020063121026 /
path/to/regiona.mjl_2020063120753 /path/to/regiona.mjl_2020063120800 /
path/to/regiona.mjl_2020063120801 /path/to/regiona.mjl_2020063120812 /
path/to/regiona.mjl_2020063120822 /path/to/regiona.mjl_2020063121026 /
path/to/regionb.mjl_2020063120753 /path/to/regionb.mjl_2020063120800 /
path/to/regionb.mjl_2020063120801 /path/to/regionb.mjl_2020063120812 /
path/to/regionb.mjl_2020063120822 /path/to/regionb.mjl_2020063121026 /path/
to/regionc.mjl_2020063120753 /path/to/regionc.mjl_2020063120800 /path/
to/regionc.mjl_2020063120801 /path/to/regionc.mjl_2020063120812 /path/to/
regionc.mjl_2020063120822 /path/to/regionc.mjl_2020063121026
# Out of chain total        : 1.68 MiB

# Summary
| Region      | Size Total (MiB) | Recoverability up to |
| ---------- | ---------------- | -------------------- |
| GTMWIZARDS | 0.21             | 03-MAR-2020 12:10:35 |
| REGIONA    | 0.21             | 03-MAR-2020 12:10:35 |
| REGIONB    | 0.21             | 03-MAR-2020 12:10:35 |
| REGIONC    | 0.21             | 03-MAR-2020 12:10:35 |
```

Global Buffer Dashboard

The %dirty global buffer dashboard displays the region-wise percentage of global buffers vs dirty global buffers in a bar graph and the number of global buffers in a table. A large number of global buffers implies a large number of dirty global buffers to be flushed at an epoch. This dashboard along with the dynamic journal file chain can be used to determine an appropriate epoch interval / tapering settings for the application and help troubleshoot I/O spikes and process hangs during high loads.

This %JSWRITE reference implementation uses the STDIN^%JSWRITE singlesub entrypoint of the %JSWRITE utility routine.

Example:

```
$ node node_modules/@fis-gtm/jswrite/reference_implementations/gtmjsaccessbg.js
```

%YGBLSTAT Sparkline Charts

%YGBLSTAT Sparkline shows region-wise sparkline charts for the specified collection of statistics. Typically, you would group statistics by their relevance, for example, TR0,TR1,TR2,TR3, and TR4 can be grouped together to visualize variations for TP Restarts in a condensed form.

This %JSWRITE reference implementation uses the STDIN^%JSWRITE singlesub entrypoint of the %JSWRITE utility routine.

Example:

```
# Set the environment variable LC_ALL to a UTF8 locale (needed for the %JSWRITE Sparkline
 Chart (GTMJSGVSTATS.m/gtmjsgvstats.js) reference implementation)
node node_modules/@fis-gtm/jswrite/reference_implementations/gtmjsgvstats.js   # By default,
 this reference implementation displays a sparkline chart for SETs and GETs. For monitoring
 other statistics, open gtmjsgvstats.js and update the STAT array to include the list of
 statistics which require sparkline chart style monitoring.
```

^%YGBLSTAT to JSON

GTMJSGVSTATS.m is routine used in the %YGBLSTAT sparkline chart reference implementation. You can use the same routine to create a JSON file which can be imported to spreadsheet software such as Excel. The following command creates gvstats.json which includes all statistics produced with the %YGBLSTAT utility for all regions which have opted for statistics sharing.

```
$gtm_dist/mumps -r ^GTMJSGVSTATS | $gtm_dist/mumps -r STDIN^%JSWRITE singlesub > /src/
toExcel/gvstats.json
```

You can also choose the statistics that you want in the JSON file with the following:

```
$gtm_dist/mumps -r ^GTMJSGVSTATS TR0,TR1,TR2,TR3,TR4 | $gtm_dist/mumps -r STDIN^%JSWRITE
 singlesub > /src/toExcel/gvstats.json
```

You can import gvstats.json in Excel using the "Data->Get Data->From File->From JSON" option of Excel.

This %JSWRITE reference implementation uses the STDIN^%JSWRITE singlesub entrypoint of the %JSWRITE utility routine.

JSWRITE Utility Class

JSWRITE.js is a utility class for managing objects created with the ^%JSWRITE utility routine of GT.M. It helps with object mutation and cases when you need to round-trip data back to GT.M. The use

of JSWRITE.js is optional and is provided as a convenience to JavaScript developers working on ^ %JSWRITE utility routine.

The constructor of JSWRITE.js takes one argument which can be an array collection object from %JSWRITE(glvn,[#]|[*]), one object string %JSWRITE(glvn,#|*), or an empty object. After instantiation, the object has the following getter function.

Getter Method

getSubs	Returns an array containing the unsubscripted GT.M glvname and their subscripts. The first element of the array is always the glvname.
getValue	Returns the value of the GT.M glvname.
getZWRLines	Contains an array of ZWR lines with each element represents one valid argument for use with SET @ in GT.M.

For example, for following GT.M lv structure:

```
GTM>zwrite
A("One","Two","Three","Four")="Demo"

GTM>do ^%JSWRITE("A","*")
{"A":{"One":{"Two":{"Three":{"Four":"Demo"}}}}}
```

..instantiate an object as follows:

```
> var JSWRITE=require("@fis-gtm/jswrite")
> JSObj=new JSWRITE({"A":{"One":{"Two":{"Three":{"Four":"Demo"}}}}})
```

The getter functions return the following values:

JSObj.getSubs	['A', 'One', 'Two', 'Three', 'Four']
JSObj.getValue	'Demo'
JSObj.getZWRLines	['A("One","Two","Three","Four")="Demo"']

Other Methods

toZWR(JSONobj)	Attempts to traverse and convert the JSON paths and their values of JSONObj to a form that can be used with SET @. The argument must be an object. Specifying an array produces an error. Invoking the .toZWR(JSONObj) method appends the getZWRLines array.
customZWR(glvName, [subscripts], glvValue)	Returns an array containing the unsubscripted GT.M glvname and their subscripts. The first element of the array is always the glvname.

validateJSON(str)	Checks whether the string argument is a valid JSON object. If str is a valid JSON string, it returns the result of JSON.parse(str).

Examples:

.toZWR()

```
>var JSWRITE=require("@fis-gtm/jswrite")
undefined

> JSObj=new JSWRITE({})
undefined

> JSObj.toZWR({"A":{"A1":[1,2,{"C":"C3"},4]}})
undefined
> JSObj.getZWRLines
[
  'A("A1","0")="1"',
  'A("A1","1")="2"',
  'A("A1","2","C")="C3"',
  'A("A1","3")="4"'
]
>
```

.customZWR()

```
> JSObj.customZWR("A",[1,2,3,4,5],"ABCD\n")
undefined

> JSObj.getZWRLines
[ 'A(1,2,3,4,5)="ABCD"_$C(10)_""' ]
```

Error and Other Messages

DBFILNOFULLWRT ⊕

DBFILNOFULLWRT, Disabling fullblock writes. iiii tttt: bbbb

MUPIP Warning: Indicates full block writes were not successfully enabled. iiii describes the issue, tttt describes the type and bbbb is a block size.

Action: Consider planning to choose a blocksize better aligned with the file system blocksize at the next opportunity.