

# GT.M

## Release Notes

V7.1-002

## Contact Information

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

GT.M Support for customers: [gtmsupport@fisglobal.com](mailto:gtmsupport@fisglobal.com)  
Automated attendant for 24 hour support: +1 (484) 302-3248  
Switchboard: +1 (484) 302-3160

## Legal Notice

Copyright ©2023 Fidelity National Information Services, Inc. and/or its subsidiaries. All Rights Reserved.






Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts.

GT.M™ is a trademark of Fidelity National Information Services, Inc. Other trademarks are the property of their respective owners.

This document contains a description of GT.M and the operating instructions pertaining to the various functions that comprise the system. This document does not contain any commitment of FIS. FIS believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. FIS is not responsible for any errors or defects.

<b>Revision History</b>		
Revision 1.2	30 October 2023	Fix typo : DE53892 -> DE538928
Revision 1.1	12 October 2023	RHEL9 is supported
Revision 1.0	19 September 2023	V7.1-002

# Table of Contents

V7.1-002 .....	1
Overview .....	1
Conventions .....	1
Platforms .....	3
Platform support lifecycle .....	5
Additional Installation Instructions .....	6
Recompile .....	6
Rebuild Shared Libraries or Images .....	7
Compiling the Reference Implementation Plugin .....	7
Re-evaluate TLS configuration options .....	8
Upgrading to V7.1-002 .....	8
Stage 1: Global Directory Upgrade .....	8
Stage 2: Database Files Upgrade .....	9
Stage 3: Replication Instance File Upgrade .....	14
Stage 4: Journal Files Upgrade .....	14
Stage 5: Trigger Definitions Upgrade .....	14
Managing M mode and UTF-8 mode .....	14
Setting the environment variable TERM .....	16
Installing Compression Libraries .....	16
Change History .....	17
V7.1-002 .....	17
Database .....	19
Language .....	21
System Administration .....	23
Other .....	25
Error and Other Messages .....	27
BSIZTOOLARGE  .....	27
BUFFSIZETOOSMALL  .....	27
EXCEEDRCTLRNDWN  .....	27
ORLBKROLLED  .....	27
RSVDBYTE2HIGH  .....	28

**This page is intentionally left blank.**

# V7.1-002

## Overview

V7.1-002 provides the ability to specify differing reserved bytes for data and index blocks, which permits a possible additional tuning approach. MUPIP REORG has an additional feature and some fixes that should make it a more effective tool, and there are an assortment of other enhancements and fixes.

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

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



### Note

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

## Conventions

This document uses the following conventions:

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



### Note

The term UNIX refers to the general sense of all platforms on which GT.M uses a POSIX API. As of this date, this includes: AIX and GNU/Linux x86\_64.

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

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

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

✔ denotes a new feature that requires updating the manuals.

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

⊖ denotes deprecated messages.

⚠ denotes revised messages.

⊕ denotes added messages.

## Platforms

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


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

As of the publication date, FIS supports this release on the hardware and operating system versions below. Contact FIS for a current list of Supported platforms. The reference implementation of the encryption reference plugin has its own additional requirements.

Platform	Supported Versions	Notes
IBM Power Systems AIX	7.1 TL 5, 7.2 TL 5, 7.3 TL 1	<p>Only 64-bit versions of AIX with POWER7 as the minimum required CPU architecture level are Supported.</p> <p>While GT.M supports both UTF-8 mode and M mode on this platform, there are problems with the AIX ICU utilities that prevent FIS from testing 4-byte UTF-8 characters as comprehensively on this platform as we do on others.</p> <p>Running GT.M on AIX 7.1 requires APAR IZ87564, a fix for the POW() function, to be applied. To verify that this fix has been installed, execute <b>instfix -ik IZ87564</b>.</p> <p>Only the AIX jfs2 filesystem is Supported. Other filesystems, such as jfs1 are Supportable, but not Supported. FIS strongly recommends use of the jfs2 filesystem on AIX; use jfs1 only for existing databases not yet migrated to a jfs2 filesystem.</p>
x86_64 GNU/Linux	Red Hat Enterprise Linux 7.9, 8.8, 9.2; Ubuntu 20.04 LTS, 22.04 LTS, and 23.04;	<p>To run 64-bit GT.M processes requires both a 64-bit kernel as well as 64-bit hardware. As of V7.1-001, GT.M on x86-64 requires hardware/virtualized support for AVX instructions.</p> <p>GT.M should also run on recent releases of other major Linux distributions with a contemporary Linux kernel (2.6.32 or later), glibc (version 2.12 or later) and ncurses (version 5.7 or later).</p>

Platform	Supported Versions	Notes
	Amazon Linux 2	<p>Due to build optimization and library incompatibilities, GT.M versions older than V6.2-000 are incompatible with glibc 2.24 and up. This incompatibility has not been reported by a customer, but was observed on internal test systems that use the latest Linux software distributions from Fedora (26), Debian (unstable), and Ubuntu (17.10). In internal testing, processes either hung or encountered a segmentation violation (SIG-11) during operation. Customers upgrading to Linux distributions that utilize glibc 2.24+ must upgrade their GT.M version at the same time as or before the OS upgrade.</p> <p>GT.M requires a compatible version of the libtinfo library. On Red Hat, the ncurses-libs and ncurses-compat-libs packages contain the libtinfo library. On Debian/Ubuntu, libtinfo5 and libncurses5 packages contain the libtinfo library. If any of these packages is not already installed on your system, please install using an appropriate package manager.</p> <p>To support the optional WRITE /TLS fifth argument (the ability to provide / override options in the tlsid section of the encryption configuration file), the reference implementation of the encryption plugin requires libconfig 1.4.x or later.</p> <p>Only the ext4 and xfs filesystems are Supported. Other filesystems are Supportable, but not Supported. Furthermore, if you use the NODEFER_ALLOCATE feature, FIS strongly recommends that you use xfs. If you must use NODEFER_ALLOCATE with ext4, you <b>must</b> ensure that your kernel includes commit d2dc317d564a46dfc683978a2e5a4f91434e9711 (search for d2dc317d564a46dfc683978a2e5a4f91434e9711 at <a href="https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3">https://www.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.0.3</a>). The Red Hat Bugzilla identifier for the bug is 1213487. With NODEFER_ALLOCATE, do not use any filesystem other than ext4 and a kernel with the fix, or xfs.</p> <p>Our testing has shown an interaction between glibc 2.36 and all versions of GT.M on Linux/x86_64 systems without AVX2 support. This can cause segmentation violations (SIG-11) in processes performing concurrent updates to the same database block, which terminate the process, but do not damage the database. The issue is due to the way glibc performs certain memory operations when using SSE2 instructions. The glibc behavior was subsequently modified to avoid this issue, and the change was included in glibc 2.37, however, we have not yet confirmed the change resolved issue. Linux/x86_64 systems with support for AVX2 instructions are not vulnerable, as glibc chooses its AVX2 implementation, when available, over its SSE2 implementation, and the problematic behavior is</p>



Platform	Supported Versions	Notes
		<p>specific to SSE2. Note, depending on how CPU virtualization is configured, that virtual environments may not support AVX2 even if the underlying hardware does.</p> <p>Ubuntu 18.04 LTS and 21.10 are Supportable.</p> <div style="display: flex; align-items: center;">  <div style="border: 1px solid black; padding: 5px;"> <p><b>Note</b></p> <ul style="list-style-type: none"> <li>● To use TLSv1.3 with OpenSSL 1.1.1 and up, you must recompile the reference encryption plugins</li> <li>● RHEL 8 includes compat-openssl10.x86_64 for binaries compiled against OpenSSL 1.0.2 on RHEL 7</li> </ul> </div> </div>



**Important**

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

**Platform support lifecycle**

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

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

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

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

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


## Additional Installation Instructions

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



### Caution

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

- FIS strongly recommends installing each version of GT.M in a separate (new) directory, rather than overwriting a previously installed version. If you have a legitimate need to overwrite an existing GT.M installation with a new version, you must first shut down all processes using the old version. FIS suggests installing GT.M V7.1-002 in a Filesystem Hierarchy Standard compliant location such as `/usr/lib/fis-gtm/V7.1-002_arch` (for example, `/usr/lib/fis-gtm/V7.1-002_x86_64` on Linux systems). A location such as `/opt/fis-gtm/V7.1-002_arch` would also be appropriate.
- Use the appropriate MUPIP command (e.g. ROLLBACK, RECOVER, RUNDOWN) of the old GT.M version to ensure all database files are cleanly closed.
- Make sure `gtmsecshr` is not running. If `gtmsecshr` is running, first stop all GT.M processes including the DSE, LKE and MUPIP utilities and then perform a **MUPIP STOP *pid\_of\_gtmsecshr***.
- Starting with V6.2-000, GT.M no longer supports the use of the deprecated `$gtm_dbkeys` and the master key file it points to for database encryption. To convert master files to the libconfig format, please click  to download the CONVDDBKEYS.m program and follow instructions in the comments near the top of the program file. You can also download CONVDDBKEYS.m from <http://tinco.pair.com/bhaskar/gtm/doc/articles/downloadables/CONVDDBKEYS.m>. If you are using `$gtm_dbkeys` for database encryption, please convert master key files to libconfig format immediately after upgrading to V6.2-000 or later. Also, modify your environment scripts to include the use of `gtmencrypt_config` environment variable.

## Recompile

- Recompile all M and C source files.

## Rebuild Shared Libraries or Images

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

## Compiling the Reference Implementation Plugin

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

1. Install the development headers and libraries for libgcrypt, libgpgme, libconfig, and libssl. On Linux, the package names of development libraries usually have a suffix such as -dev or -devel and are available through the package manager. For example, on Ubuntu\_x86\_64 a command like the following installs the required development libraries:

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

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

2. Unpack \$gtm\_dist/plugin/gtmcrypt/source.tar to a temporary directory.

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

3. Follow the instructions in the README.
  - Open Makefile with your editor; review and edit the common header (IFLAGS) and library paths (LIBFLAGS) in the Makefile to reflect those on your system.
  - Define the gtm\_dist environment variable to point to the absolute path for the directory where you have GT.M installed
  - Copy and paste the commands from the README to compile and install the encryption plugin with the permissions defined at install time



### Note

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

4. When reinstalling or upgrading GT.M, stop existing gpg-agents. The agents may be working with information about the prior GT.M installation, such as GNUPGHOME, that will not work with the

new version. Additionally, if the process deletes the GPG agent's socket, proper operation requires a new agent.

5. It is a good idea to read the A&O Guide section entitled "Special note - GNU Privacy Guard and Agents" and re-evaluate the GPG configuration options in use.

## Re-evaluate TLS configuration options

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

## Upgrading to V7.1-002



### Before you begin

GT.M supports upgrade from V5\*, V6.\* and V7.\* versions to V7.1-002.

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

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

GT.M upgrade procedure for V7.1-002 consists of 5 stages:

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

Read the upgrade instructions of each stage carefully. Your upgrade procedure for GT.M V7.1-002 depends on your GT.M upgrade history and your current version.

## Stage 1: Global Directory Upgrade

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

**To upgrade from any previous version of GT.M:**

- Open your Global Directory with the GDE utility program of GT.M V7.1-002.
- Execute the EXIT command. This command automatically upgrades the Global Directory.
- If you inadvertently open a Global Directory of an old format with no intention of upgrading it, execute the QUIT command rather than the EXIT command.

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

- Open the global directory with the GDE utility program of V7.1-002.
- Execute the SHOW -COMMAND -FILE=file-name command. This command stores the current Global Directory settings in the file-name command file. If the old version is significantly out of date, edit the command file to remove the commands that do not apply to the old format. Alternatively, you can use the output from SHOW -ALL or SHOW -COMMAND as a guide to manually enter equivalent GDE commands for the old version.

An analogous procedure applies in the reverse direction.

## Stage 2: Database Files Upgrade

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

There are three upgrade paths available when you upgrade to V7.1-002.

### V7 Upgrade Path 1: In-place Upgrade

To upgrade from GT.M V7\*:

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

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

There are two phases to upgrade from V6 to V7:

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

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

While these are the basic steps, customers must integrate them with appropriate operational practice and risk mitigating procedures, such as comprehensive testing, backup, integrity checks, journal and replication management, and so on based on their environments and risk tolerance. FIS strongly

recommends performing a MUPIP INTEG (-FAST), of the database and creating a backup prior to upgrade. Customers must test these utilities against copies of their own production files, using their planned procedures, before undertaking the conversion of current production files.

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

### Phase 1: Standalone MUPIP UPGRADE

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

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

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

UPGRADE:

- Requires standalone access
- Turns off journaling and replication for the duration of UPGRADE
- When encountering an error where the command specifies multiple regions, UPGRADE moves on to the next region, while for a single file/region, it terminates; avoid any unnecessary <CTRL\_C> or MUPIP STOP (or kill) of an active MUPIP UPGRADE process, as such an action leaves the database region effectively unusable
- Estimates and reports the space required for its work
  - UPGRADE estimates are intended to be generous, and, particularly for small databases, they may seem unnecessarily large
  - If MUPIP is not authorized to perform a required file extension, that is, the extension amount is defined as zero (0), it produces an error before it does anything that would damage the selected database file
- Moves blocks from immediately after the existing master map to make room for a V7 master map
  - Depending on the block size and the GT.M version with which it was created, the new starting Virtual Block Number (VBN), the location of block zero for the database file, may exceed the starting VBN for a database created with V7, which causes a minor waste of space
  - UPGRADE relocates blocks in multiples of 512 to align blocks with their local bitmaps

- Eliminates any globals that previously existed, but have been KILL'd at the name level; these global variable trees (GVTs) contain only a level one (1) root block and an empty data (level zero) block and are "invisible" to the GT.M process run-time
- Stores the offset GT.M must apply to the original block pointers as a consequence of the relocation of the starting VBN
- Upgrades the directory tree (DT) block pointers from 32- to 64-bits; this requires splitting any blocks that do not have sufficient space to accommodate the larger block pointers
- Ensures that all is work is flushed to secondary storage
- Reports completion of its activity on a database file with a "MUPIP MASTERMAP UPGRADE completed" message

At this point, after a successful MUPIP UPGRADE:

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

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

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

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

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

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

MUPIP REORG -UPGRADE:

- Requires standalone access

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

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

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

- MUPIP REORG

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

- MUPIP REORG -ENCRYPT
- MUPIP INTEG -TN\_RESET

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

Failure to perform Phase 3 has **NO** implications for V7.1-002 but might be an issue for any as-yet unplanned further enhancement.

### V7 Upgrade Path 2: EXTRACT and LOAD

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

- MUPIP EXTRACT -FREEZE followed by a MUPIP LOAD

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

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



- MERGE command with two global directories and Extended References

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

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



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

### V7 Upgrade Path 3: No change

Continue using your V6 databases with GT.M V7.1-002. In case you do not wish to operate with files of differing format, specify the -V6 qualifier when invoking MUPIP CREATE.

### Choosing the right upgrade path

Choose V7 Upgrade Path 1 or 2 if you anticipate a database file to grow to over 994Mi blocks or require trees of over 7 levels as V7.1-000 supports 16Gi blocks and 11 levels. Note that the maximum size of a V7 database file having 8KiB block size is 114TiB (8KiB\*16Gi).

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

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

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

## Database Compatibility Notes

- Changes to the database file header may occur in any release. GT.M automatically upgrades database file headers as needed. Any changes to database file headers are upward and downward compatible within a major database release number, that is, although processes from only one GT.M release can access a database file at any given time, processes running different GT.M releases with the same major release number can access a database file at different times.
- Databases created with V5.3-004 through V5.5-000 can grow to a maximum size of 224Mi (234,881,024) blocks. This means, for example, that with an 8KiB block size, the maximum database file size is 1,792GiB; this is effectively the size of a single global variable that has a region to itself and does not itself span regions; a database consists of any number of global variables. A database created with GT.M versions V5.0-000 through V5.3-003 can be upgraded with the V5 version of MUPIP UPGRADE to increase the limit on database file size from 128Mi to 224Mi blocks.

- Databases created with V5.0-000 through V5.3-003 have a maximum size of 128Mi (134, 217,728) blocks. GT.M versions V5.0-000 through V5.3-003 can access databases created with V5.3-004 and later as long as they remain within a 128Mi block limit.
- Database created with V6.0-000 through V6.3-014 have a maximum size of 1,040,187,392 (992Mi) blocks.
- Database created with V7.0-000 and up have a maximum size of 17,179,869,184 (16Gi) blocks.

### Stage 3: Replication Instance File Upgrade

GT.M V7.1-002 does not require new replication instance files when upgrading from any version after V6.0-000.

### Stage 4: Journal Files Upgrade

On every GT.M upgrade:

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



#### Important

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

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

### Stage 5: Trigger Definitions Upgrade

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

### Managing M mode and UTF-8 mode

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

On a system that has ICU installed, GT.M optionally installs support for both M mode and UTF-8 mode, including a utf8 subdirectory of the directory where GT.M is installed. From the same source file, depending upon the value of the environment variable gtm\_chset, the GT.M compiler generates

an object file either for M mode or UTF-8 mode. GT.M generates a new object file when it finds both a source and an object file, and the object predates the source file and was generated with the same setting of `$gtm_chset/$ZCHset`. A GT.M process generates an error if it encounters an object file generated with a different setting of `$gtm_chset/$ZCHset` than that processes' current value.

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

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

- Actual files for GT.M executable programs (`mumps`, `mupip`, `dse`, `lke`, and so on) are in the parent directory, that is, the location specified for installation.
- Object files for programs written in M (GDE, utilities) have two versions - one compiled with support for UTF-8 mode in the `utf8` subdirectory, and one compiled without support for UTF-8 mode in the parent directory. Installing GT.M generates both versions of object files, as long as ICU 3.6 or greater is installed and visible to GT.M when GT.M is installed, and you choose the option to install UTF-8 mode support. During installation, GT.M provides an option that allows placing the object code in shared libraries in addition to individual files in the directory.
- The `utf8` subdirectory has files called `mumps`, `mupip`, `dse`, `lke`, and so on, which are relative symbolic links to the executables in the parent directory (for example, `mumps` is the symbolic link `../mumps`).
- When a shell process sources the file `gtmprofile`, the behavior is as follows:
  - If `$gtm_chset` is "m", "M" or undefined, there is no change from the previous GT.M versions to the value of the environment variable `$gtmroutines`.
  - If `$gtm_chset` is "UTF-8" (the check is case-insensitive),
    - `$gtm_dist` is set to the `utf8` subdirectory (that is, if GT.M is installed in `/usr/lib/fis-gtm/gtm_V7.1-002_i686`, then `gtmprofile` sets `$gtm_dist` to `/usr/lib/fis-gtm/gtm_V7.1-002_i686/utf8`).
    - On platforms where the object files have not been placed in a `libgtmutil.so` shared library, the last element of `$gtmroutines` is `$gtm_dist($gtm_dist/..)` so that the source files in the parent directory for utility programs are matched with object files in the `utf8` subdirectory. On platforms where the object files are in `libgtmutil.so`, that shared library is the one with the object files compiled in the mode for the process.

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

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

## Setting the environment variable TERM

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

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

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

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

## Installing Compression Libraries

If you plan to use the optional compression facility for replication, you must provide the compression library. The GT.M interface for compression libraries accepts the zlib compression libraries without any need for adaptation. These libraries are included in many UNIX distributions and are downloadable from the zlib home page. If you prefer to use other compression libraries, you need to configure or adapt them to provide the same API as that provided by zlib.

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

By default, GT.M searches for the libz.so shared library in the standard system library directories (for example, /usr/lib, /usr/local/lib, /usr/local/lib64). If the shared library is installed in a non-standard location, before starting replication, you must ensure that the environment variable LIBPATH (AIX) or LD\_LIBRARY\_PATH (GNU/Linux) includes the directory containing the library. The Source and Receiver Server link the shared library at runtime. If this fails for any reason (such as file not found, or insufficient authorization), the replication logic logs a DLLNOOPEN error and continues with no compression.

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

## Change History

### V7.1-002

Fixes and enhancements specific to V7.1-002:

<b>Id</b>	<b>Prior Id</b>	<b>Category</b>	<b>Summary</b>
GTM-DE324947	-	Language	Correct error message when gtmtnls_passwd_TLSID is not set
GTM-DE533918	-	Language	Error on READ or WRITE to a SOCKET device with no active sockets
GTM-DE534846	-	Language	\$ZTIMEOUT presents the time remaining value to microsecond precision 🟢
GTM-DE538928	-	Admin	Update Process properly closes prior generation journal files
GTM-DE549071	-	Admin	REORG traverses the database correctly and accepts restrictions on which levels it processes
GTM-DE549072	-	Admin	REORG succeeds in splitting any blocks that require a split
GTM-DE549073	-	Admin	REORG no longer accepts a combination of reserved bytes and fill factor which together target an impossible block size
GTM-DE556365	-	Admin	Receiver Server continues to accept connections after a TLSCONVSOCK error
GTM-DE556760	-	Admin	MUPIP UPGRADE appropriately processes V6 database files that exceed the maximum tree depth (7 levels) associated with pre-V7 versions
GTM-F135405	-	Admin	Syslog message on Replication state when Journaling turns off
GTM-F135409	GTM-9418	Other	Replace an assertpro in relinkctl_open() with an error message
GTM-F167609	-	Other	SOCKET Devices support TLSv1.3 Post Handshake Authentication 🟢
GTM-F167995	-	Language	The GT.M TLS plugin library exposes an external call interface providing cipher suite and version information 🟢
GTM-F197635	-	Admin	GT.M supports independent index and data reserved bytes values 🟢
GTM-F217678	-	DB	Online Rollback syslogs change to database logical state
GTM-F229760	-	Language	\$ZICUVER provide the ICU version if available 🟢

Id	Prior Id	Category	Summary
GTM-F235980	-	Admin	✔ GT.M supports increased user control of tcp buffer sizing in replication ✔

## Database

- When MUPIP JOURNAL -ROLLBACK -ONLINE changes the logical state of a database it issues an ORLBKROLLED message to the system log. Previously, MUPIP did not announce this state change in the system log. (GTM-F217678)

**This page is intentionally left blank.**



## Language

- In the event of failure to acquire the TLS password, the GT.M TLS reference plugin issues an appropriate error message. Previously, in some cases, failure to acquire the TLS password resulted in a garbled error message. (GTM-DE324947)
- GT.M issues an NOSOCKETINDEV error the first time a process in direct mode attempts to read from or write to a socket principal device with no socket descriptors, and issues a fatal NOPRINCIO error upon a subsequent attempt. Previously a process that entered direct mode with an empty socket principal device would loop indefinitely and consume CPU resources. (GTM-DE533918)
- \$ZTIMEOUT presents the time remaining value to microsecond precision; previously it only showed time with precision in milliseconds or less. (GTM-DE534846) 🟢
- The GT.M TLS plugin library exposes an external call interface providing ciphersuite and version information. The four new functions are:

```
getversion:                xc_long_t gtm_tls_get_version(0:xc_char_t*[2048])
gettlslibsslversion:      xc_long_t
  gtm_tls_get_TLS_version(0:xc_char_t*[2048],I:xc_char_t*,0:xc_char_t*[2048])
getdefaultciphers:        xc_long_t
  gtm_tls_get_defaultciphers(0:xc_char_t*[4096],I:xc_char_t*,0:xc_char_t*[2048])
getciphers:               xc_long_t
  gtm_tls_get_ciphers(0:xc_char_t*[4096],I:xc_char_t*,I:xc_char_t*,I:xc_char_t*,I:xc_char_t*,0:xc_char_t*[2048])
```

The following entry points provide the supported cipher suite information. Except where noted, the `$gtmconf` configuration file and `gtmtls_passwd_<TLS_ID>` are not required.

### **getdefaultciphers**

- 1st parameter contains the default list of ciphers based on the 2nd parameter
- 2nd parameter directs the interface to report the OpenSSL default cipher suite for TLSv1.2 ("tls1\_2") or TLSv1.3 ("tls1\_3")
- 3rd parameter is an error string (allocated by the external call interface).
- The function returns negative as failure and positive for the number of colon delimited pieces in the return string.

### **getciphers**

- 1st parameter contains the list of available ciphers based on the 2nd parameter
- 2nd parameter directs the interface to report the OpenSSL default cipher suite for TLSv1.2 ("tls1\_2") or TLSv1.3 ("tls1\_3")

## Language

- 3rd parameter directs the interface to report the cipher suite using the cipher suite defaults for "SOCKET" Device or "REPLICATION" server
- (optional) 4th parameter directs the interface to use the name TLS ID from the \$gtmencrypt\_conf configuration file. Using the null string makes \$gtmencrypt\_config optional. Using a TLS ID with certificates requires \$gtmtls\_passwd\_<TLS ID>
- (optional) 5th parameter directs the interface to use the supplied cipher suite string when determining supported ciphers
- 6th parameter is an error string (allocated by the external call interface)
- The function returns negative as failure and positive for the number of colon delimited pieces in the return string

The following entry points provide version information.

### **getversion**

- 1st parameter contains the GT.M TLS plugin version as a string.
- The function returns the GT.M TLS plugin version as a number.

### **gettlslibversion**

- 1st parameter contains the OpenSSL string
  - 2nd parameter directs the function to report the "run-time" or "compile-time" OpenSSL version
  - 3rd parameter is an error string (allocated by the external call interface)
  - The function returns the OpenSSL version number or negative on failure (GTM-F167995) 🟢
- The \$ZICUVER Intrinsic Special Variable provides the current International Character Utilities (ICU) version or an empty string if ICU is not available. GT.M requires ICU to support UTF-8 operation. Previously, GT.M did not make this information available to the application code. (GTM-F229760) 🟢

## System Administration

- The Update Process properly closes prior generation journal files. Previously, due to a regression in V6.3-005, it was possible for the Update Process to miss closing a prior journal file when a MUPIP SET switched a journal file and the Update Process was waiting to receive updates from the Receiver Server. The workaround was to avoid explicitly switching journal files on the replicating instances. (GTM-DE538928)
- MUPIP REORG traverses all index blocks and achieves a compact and optimally-structured database with a single pass. Previously, REORG failed to visit certain categories of blocks, including the root block and blocks it newly created or modified, and it required an indefinite number of passes to achieve optimal block structure. As a workaround for previous releases, users may consider repeating REORG operations until the command reports few blocks coalesced and split. In addition, REORG now recognizes a new qualifier, `-MIN_LEVEL=n`, which specifies the minimum level of block it may process. The default is `-MIN_LEVEL=0`. `-MIN_LEVEL=1` instructs reorg only to process index blocks and can be understood as the REORG equivalent of an INTEG `-FAST`. (GTM-DE549071)
- MUPIP REORG correctly handles block splits at the index level which are caused by a reorg-related block split at a lower-level index or data block. Previously, GT.M would sometimes fail to split these blocks, which prevented the original lower data- or index-level split from taking place. Together, these could prevent REORG from enforcing the provided fill factor and/or reserved bytes setting; a failure which persisted on all subsequent attempts, or until the database structure has changed as a result of new updates. As a workaround, users of previous releases can address a subset of the failures-to-split by passing a slightly different `-fill_factor`, provided that reserved bytes are disabled. (GTM-DE549072)
- MUPIP REORG enforces a minimum target block size of the block size minus the maximum reserved bytes. Previously, GT.M failed to reject a combination of reserved bytes and `-fill_factor` which effectively reserved a space larger than the database block size and caused database damage. Users of previous releases should use either reserved bytes or `-fill_factor`, but not both, as a means of achieving REORG sparseness. (GTM-DE549073)
- The Receiver Server, configured to use TLS, continues waiting for new connections when a Source Server fails to establish a TLS session (e.g. misconfiguration). Previously, a Receiver Server configured for TLS but without `-PLAINTEXTFALLBACK` would exit with `TLSCONVSOCK` error message when it failed to establish a TLS session. (GTM-DE556365)
- MUPIP UPGRADE correctly handles global trees with depths greater than supported by GT.M version 6.x, but possible when running version 7.x on a version 6.x database. Previously, attempts to upgrade V6 databases containing these tall global trees would result in a segmentation violation and the process would terminate early without upgrading the database. (GTM-DE556760)
- GT.M issues `JNLCLOSED` and `REPLSTATE` syslog messages for a database file when journaling is closed and the replication status changes from `ON` to `WAS_ON`. In the `WAS_ON` state, replication continues until and unless replication requires records that are no longer available from the replication journal pool, as it can no longer use the journal files to find those records. While in

this WAS\_ON state if you can resume journaling, you may be able to avoid having to refresh the associated Secondary Instances. The need to use journal files in replication is a function of the size the replication journal pool, the rate of updates and operator actions that impact replication. Previously, GT.M issued a REPLJNLCLOSED message when journaling was disabled for a database file, but did not indicate that the replication status changed to WAS\_ON. (GTM-F135405)

- The -INDEX\_RESERVED\_BYTES and -DATA\_RESERVED\_BYTES qualifiers for MUPIPSET (and DSE CHANGE -FHEAD) allow independent adjustment of reserved bytes for each block type. Previously GT.M did not provide the flexibility to set these values independently. The -RESERVED\_BYTES qualifier continues to adjust both types of block to the same value. When the command specifies -RESERVED\_BYTES along one of the more specific qualifiers, MUPIP applies the more general -RESERVED\_BYTES value to the block type unspecified by the other qualifier. MUPIP DUMPFHEAD reports the number of bytes reserved for each type of block (as does DSE DUMP -FHEAD). Reserving additional bytes in index blocks can reduce the number of records in any given index block and may reduce invalidation and search restarts in some workloads. (GTM-F197635) ✓
- GT.M recognizes the -[NO]{SEND|RCV}BUFFSIZE=n qualifiers for use with MUPIP REPLICATE -{SOURCE|RECEIVER} -START invocations. These qualifiers affect the TCP send and receive buffers associated with the socket GT.M uses for replication, rather than the receive/journal pools. When invoked with -{SEND|RCV}BUFFSIZE=n, where n is a positive decimal or hexadecimal (0x) integer, GT.M attempts to increase the specified socket buffer size to match the provided value. If the specified buffer is already larger than the provided value, GT.M does not attempt to reduce its size. By default, GT.M attempts to increase the size of the send buffer (SO\_SNDBUF) or receive buffer (SO\_RCVBUF) if either is smaller than the internal target value specified below.

	SO_SNDBUF	SO_RCVBUF
Source	1MiB	1KiB
Receiver	1KiB	1MiB

If a user requests a size for either buffer smaller than necessary to support GT.M replication, GT.M will print a BUFFSIZETOOSMALL warning, act as if the minimum size had been specified instead, and continue. Due to a quirk in how the Linux kernel reports socket buffer sizes, users of GT.M on Linux can expect GT.M to report a final size approximately twice what is requested; the additional space is used internally by the kernel. When invoked with -NO{SEND|RCV}BUFFSIZE, GT.M leaves the management of the initial size of the specified buffer to the execution environment, including the system defaults, local configuration settings, and operating system. In some cases, such as when the operating system dynamically manages the size of the relevant buffers, this can may lead to better performance in conditions which call for larger sizes than the GT.M default. Previously GT.M enforced a fixed minimum size for each buffer and did not permit the explicit request of receive and send buffer sizes. In addition, GT.M now correctly sets the buffer sizes, when warranted, before establishing a connection. Previously, GT.M waited until after the source and receiver had established a connection to perform any modification of the buffer sizes, which prevented the TCP connection from taking full advantage of any increase in size. (GTM-F236066) (GTM-F235980) ✓✓

## Other

- GT.M issues an EXCEEDRCTLNDWN error message when competing processes repeatedly rundown the relinkctl and another process tries to connect to the relinkctl more than 16 times. Previously, under these circumstances GT.M failed with an GTMASSERT2 error. (GTM-F135409)
- When using TLS configuration verify-mode option SSL\_VERIFY\_PEER, GT.M enables TLSv1.3 Post Handshake Authentication (PHA) for client connections. When using TLS configuration verify-mode options SSL\_VERIFY\_PEER and SSL\_VERIFY\_POST\_HANDSHAKE, GT.M enables TLSv1.3 PHA for server connections. By itself, SSL\_VERIFY\_POST\_HANDSHAKE does not enable PHA. Previously, GT.M did not support TLSv1.3's PHA capability. This could cause problems when connecting to some TLSv1.3 servers that require PHA. (GTM-F167609) 🟢

**This page is intentionally left blank.**

## Error and Other Messages

### BSIZTOOLARGE

Last used version: undefined

**BSIZTOOLARGE**, xxxx Block larger than specified maximum size

MUPIP Error: This is a MUPIP INTEG error. Refer to the topic MUPIP INTEG Errors in About This Manual section of this manual.

Action: -

### BUFFSIZEOOSMALL

Last used version: undefined

**BUFFSIZEOOSMALL**, TCP xxxx buffer size passed to yyyy too small, setting to minimum size of zzzz.

MUPIP Warning: This indicates that the value specified in `-{SEND|RECV}BUFFSIZE` as the desired send or receive tcp buffer size was smaller than GT.M's minimum value, and that the minimum value was used instead. Note that if the buffer is initially larger than this minimum, GT.M will not attempt to reduce its size.

Action: No action necessary. Consider passing `-NO{SEND|RECV}BUFFSIZE` to enforce no minimum size at all and leave management of the buffer size to the execution environment.

### EXCEEDRCTLRNDWN

**EXCEEDRCTLRNDWN**, Maximum relinkctl rundown retries limit of nnnn exceeded

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

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

### ORLBKROLLED

**ORLBKROLLED**, ORLBKROLLED, ONLINE ROLLBACK took the database for instance iiii region rrrr corresponding to dddd to a prior state

MUPIP Warning: This message is issued to the system log when an online rollback has taken the database back to an earlier state. iiii indicates the instance ID and rrrr indicates the region for the database file ffff.

Action: Check the content of any broken or lost transaction files - this warning indicates a wholesome change in the GT.M database state, but one that may cause inconsistent application data.

### RSVDBYTE2HIGH

**RSVDBYTE2HIGH**, Record size ssss is greater than the maximum allowed for region rrrr with Block size bbbb and cccc reserved bytes dddd

Run Time Error: The attempted database update would result in a record size that is greater than what is allowed by the current database block size and index or data reserved byte setting.

Action: If the Reserved Bytes setting for the database region and block type identified is non-zero, try reducing it to allow this update, or modify the update to reduce the resulting record size. Otherwise, consider increasing the block size, but this can present operational challenges.