GemStone®

GemStone/S Release Notes

February 28, 2002

GemStone/S

Version 6.0

IMPORTANT NOTICE

Contents

Chapter 1. Feature Changes in GemStone/S 6.0

Chapter 2. Bugs Fixed in GemStone/S 6.0

Chapter 1

Feature Changes in GemStone/S 6.0

GemStone/S 6.0 fixes a number of bugs in GemStone/S 5.1.5 and adds several new features, concentrating most on improvements in scalability and performance. Many of these improvements will not be directly visible, but we expect your applications will run faster and more reliably on GemStone/S 6.0. If your installation uses multiple server machines, gains in performance will be even more apparent.

If you have any questions regarding this release, please contact your GemStone/S account manager.

New Features

GemStone/S 6.0 adds the following new features:

Improved Garbage Collection: Dedicated Page-reclaim Gems

In order to improve performance, this release features several changes to the way garbage is identified and collected, especially GcGems specialized to reclaim pages.

GemStone/S 6.0 provides the following GcGems:

GcGem
: The original GcGem, which reclaims pages with shadow objects and dead objects repository-wide, and it finalizes voting and performs other household tasks. Symbols: `#GcGem`, `#GC`

ParallelShadowReclaim
: A GcGem dedicated to the task of reclaiming pages with shadow objects, on a specified extent. You can run one of these GcGems on every extent, if you have the hardware to make this an efficient choice. Symbols: `#ParallelShadowReclaim`, `#PSR`

ParallelDeadReclaim
: A GcGem dedicated to the task of reclaiming pages with dead objects, on a specified extent. You can run one of these GcGems on every extent, if you have the hardware to make this an efficient choice. Symbols: `#ParallelDeadReclaim`, `#PDR`

The two new GcGems are intended to deal with heavy garbage collection loads. Typically they will be run during off hours, after stopping the original GcGem. If you've traced a problem to an excess of shadow objects, you can launch one or more ParallelShadowReclaim processes. If the problem is an excess of dead objects, you can launch one or more ParallelDeadReclaim processes. The new GcGems are most useful with systems having multiple extents and multiple processors.

The ParallelDeadReclaim GcGem does not perform the GcGem's tasks of epoch garbage collection, finalizing the vote and special object processing. Therefore, `markForCollection` and `markGcCandidates` will be held up in the voting step of garbage collection. And you still need to run one or both of these methods periodically, because nothing else will identify objects created in one epoch and dereferenced in another.

The ParallelDeadReclaim GcGem has another limitation: while it's running, other users on the system can read the repository, but they cannot commit any changes to it. If they try, the commit fails with the transaction conflict `#CommitsDisabled`.

The following methods are provided to start and stop the GcGems:

```
System class>>startGC: gcSymbol
System class>>startGC: gcSymbol onExtent: extentId
System class>>startGC: gcSymbol onExtents: extentCollection

System class>>stopGC: gcSymbol
System class>>stopGC: gcSymbol onExtent: extentId
System class>>stopGC: gcSymbol onExtents: extentCollection
```

The following methods return the Session IDs of the selected GcGems, or 0 if there is no GcGem of the selected type:

```
System class>>gcSession: gcSymbol
System class>>gcSession: gcSymbol onExtent: extentId
System class>>gcSession: gcSymbol onExtents: extentArray
```

For more information about the specialized GcGems and configuration examples, see Chapter 10 in the *System Administration Guide.*

Future Improvements

Two additional specialized GcGems are planned for a future release to assist with moderate garbage collection loads. The *System Administration Guide* for Version 6.0 describes their use now so you can be aware of them while planning your garbage collection strategy. These GcGems will be:

ReclaimGem
: A GcGem dedicated to the task of reclaiming shadowed pages and dead objects repository-wide. Symbols: `#ReclaimGem`, `#RCL`

EpochGem
: A GcGem dedicated to the task of performing epoch garbage collection and finalizing voting on dead objects repository-wide. Symbols: `#EpochGem`, `#EPC`

The ReclaimGem and EpochGem pair will divide up the work currently accomplished by the original GcGem. Running with this pair can be more effective on a multiple processor machine.

A new configuration option, STN_GC_SESSION_CONFIGURATION, will provide automatic GC session management. When the GC session is enabled, the Stone automatically attempts to keep GcGems running according to the value set for this option. The valid values are:

1 Keep a single GcGem running (the default)

This configuration is recommended for light garbage collection loads.

2 Keep a ReclaimGem / EpochGem pair running

This configuration is recommended for moderate or heavy garbage collection loads. You should have multiple processors to benefit from this configuration.

In this release, valid values of STN_GC_SESSION_CONFIGURATION are recognized but ignored—either setting runs the original GcGem. Consequently, you can configure your system now in the way you want to run it when the ReclaimGem / EpochGem pair become available.

For further information about the planned GcGems and automatic GC management, see Chapter 10 in the *System Administration Guide*.

New Free Frame Page Server Processes

In order to improve performance, this release features extra free-list page servers:

- These page servers scan the cache and automatically add frames to the free frame list when the free frame count falls below a configurable threshold.
- The number of free frame page servers is configurable via a new configuration file option.
- All shared page caches, local and remote, have at least one free frame page server.

GemStone uses page servers for three purposes:

- to write dirty pages to disk,
- to transfer pages from the Stone host to the shared page cache host, if different, and
- to add free frames to the free frame list, from which a Gem can take as needed.

Page servers referred to as *AIO page servers* perform all three functions. By default, at least one such page server is running at all times, though you can add more as needed. In addition, you can add one or more *free list page servers:* page servers dedicated only to the third task in the list above, adding free frames to the free list.

Under certain circumstances, free list page servers can improve overall system performance. For example, if Gems are performing many operations requiring writing pages to disk, the AIO page server may have to spend all its time writing pages, never getting a chance to add free frames to the free list. Alternatively, if Gems are performing operations that require only reading, the AIO page server will see no dirty frames in the cache—the signal that prompts it to take action. In

that case, it may sleep for a second, even though free frames are present in the cache and need to be added to the free list.

Do You Need Free List Page Servers?

A Gem can get free frames either from the free list (the quick way), or, if sufficient free frames are not listed, by scanning the shared page cache for a free frame instead. (What constitutes sufficient free frames is determined by the configuration parameter GEM_FREE_FRAME_LIMIT; for details, see page A-13.) If a Gem has to spend a large proportion of its time scanning the shared page cache, its performance may be unacceptable. Under these circumstances, extra free list page servers can sometimes help. On a single-CPU system, one extra free list page server might be all that's required; for systems with multiple CPUs, you may wish to start one at a time, checking statistics, until the problem is resolved.

By default, when you start the Stone, it tries to spawn one free list page server process on its local node. Free list page servers require a running NetLDI process, however; if the NetLDI process is not already running on the node, the attempt fails and the Stone writes a message to its log file.

Certain cache statistics can help you determine whether additional free list page servers will improve system performance. (For details about these and other statistics, see "Cache Statistics" on page 8-24.)

- If Gems have to scan the shared page cache for free frames, the cache statistic FramesFromFindFree will be greater than zero. If this is the case—especially if it significantly greater—consider starting one or more free list page servers.
- If the FreeFrameCount is consistently lower than the FreeFrameLimit, a free list page server might help (though other factors enter into the question as well).

If FramesAddedToFreeList rises significantly after starting a free list page server, the new page server has indeed benefited you; likewise, if FramesFromFindFree is reduced to zero, or near zero.

To Add Free List Page Servers

You can change the number of free list page server processes that will be started when the shared page cache is created by setting a configuration parameter, SHR_NUM_FREE_FRAME_SERVERS.

Default: 1
Minimum: 1
Maximum: (SHR_PAGE_CACHE_NUM_PROCS – 5)

Lost OT Root Handling More Robust

A new configuration parameter, STN_GEM_LOSTOT_TIMEOUT, sets the time in seconds that the Stone will wait after sending the Gem the error ABORT_ERR_LOST_OT_ROOT before terminating the Gem. Negative timeouts other than –1 are not allowed. Resolution of timeouts is one half the specified timeout interval.

If the value is –1, the Stone does not terminate the Gem; it simply retracts the session's commit record, forcing the Gem to completely reinitialize its object caches—GemStone/S's behavior previous to the 6.0 release.

CAUTION

A value of –1 entails a slight risk that the sleeping Gem will reactivate and begin writing to the shared page cache before it responds to the ABORT_ERR_LOST_OT_ROOT *error, thus corrupting the shared page cache. Other values entail a slight risk that terminating the session could cause GemStone/S to fail due to a stuck spin lock.*

Because of these risks, design your application to minimize the chances of receiving the ABORT_ERR_LOST_OT_ROOT *error.*

The internal parameter can be changed only by SystemUser.

Default: –1
Minimum: –1
Maximum: 5000000

Faster restore from backups

Several changes have been made to reduce the time required to restore the repository from a backup:

- `Repository | restoreFromBackups:arrayOfFilesOrDevices` has been optimized. As part of this change, *all* files to be restored must be specified when the method is called.
- A new method, `Repository | restoreNoShadowsFromBackups: arrayOfFilesOrDevices` is the same as `restoreFromBackups:` except that fragmented data pages are not added to the list of pages needing reclaim during the restore. The restored repository performs better after the restore but may be larger in size.

Here are some tips on getting restores completed as soon as possible:

- Use `restoreFromBackups:` or `restoreNoShadowsFromBackups:` methods, rather than restoring backups one by one. These methods require you to specify all backup files in the correct order in an Array.
- Max out the shared cache size and the private page cache size (0.5 GB is supported in 6.0), keeping the total within the 4 GB space that a 32-bit process can address.
- Set the gem's free frame limit to be very low, 500 say.
- Restore from uncompressed files rather than compressed files. The decompression libraries are slow and do very inefficient I/O. Backups created from `fullBackupCompressedTo:` can be uncompressed using **gunzip** *before* the restore operation for best performance. This makes a big difference.
- Place extents on striped file systems.
- Make sure the backup files being restored are not on the same disk spindles as the extents or tranlogs.

New and Faster Object Audits

Two new object audit methods are designed for the fastest speed. They must be performed in single-user mode, and object statistics are not printed.

`Repository | quickObjectAuditLevel1`

This method performs many but not all of the checks performed by `Repository | auditWithLimit:`. It detects "object does not exist" errors, the most common type of repository corruption. It does not audit the object table. It is up to 30 times faster than `auditWithLimit:` on large systems.

`Repository | quickObjectAuditLevel2`

This method performs object table validation in addition to the checks performed by the `quickObjectAuditLevel1` method.

Support For Larger Caches

Several changes have been made to provide support for larger caches:

- Maximum temporary object cache size has increased from 10 to 20 MB
- Maximum private page cache size (Stone and Gems) has increased from 64 MB to 512 MB.
- Maximum shared page cache size has increased on the following platforms:

Solaris:	3.75 GB
HPUX 11.x:	1.9 GB

Using a 1GB or Larger Shared Page Cache Under HP-UX

Because of unique details in the HP-UX memory architecture, GemStone applications that require a shared page cache size of 1GB and larger must be executed within a designated memory window, using the HP-UX `setmemwindow` command. Failure to use `setmemwindow` will result in the inability to start or login to the Stone, usually with warning messages regarding the inability to create or attach to shared memory.

For general information on the `setmemwindow` command, refer to the HP-UX man page.

To use this feature:

First, confirm that your HP-UX kernel has been configured to use memory windows, as described in the *GemStone/S Installation Guide*.

Next, a unique window ID must be selected for a given GemStone repository and Stone process. The particular value chosen is not important, but must be used consistently for all GemStone processes to be used with this particular repository / stone. GemStone site administrators may wish to use the HP-UX `getmemwindow` command and `/etc/services.window` file as one way to manage this. For the examples used below, *13* will be used as this unique window ID.

All GemStone commands must be run using "`setmemwindow -i <winId> -b`". For example:

```
unix> setmemwindow -i 13 -b startnetldi -a gssadmin -g
unix> setmemwindow -i 13 -b startstone gemserver60 ...
unix> setmemwindow -i 13 -b topaz -l
...
unix> setmemwindow -i 13 -b stopnetldi
unix> setmemwindow -i 13 -b stopstone gemserver60 ...
```

The one exception to this rule is in running topaz / GBS using *only* RPC sessions. Since the memory window for the spawned gem process is determined by the NetLDI process, it is not necessary to execute topaz / GBS under `setmemwindow` in this case. However, if you intend to use a *linked* session from this process, then you must use `setmemwindow`.

For convenience you can also spawn a UNIX shell (sh, csh, and so forth) from setmemwindow, which will then automatically use the designated memory window for all subsequent commands. For example:

```
unix> setmemwindow -i 13 -b csh
csh> startnetldi -a gssadmin -g
csh> startstone gemserver60 -e system.conf ...
...
```

Failure to include the -b switch with setmemwindow may result in a Segmentation Violation error when starting the stone.

Failure to use setmemwindow when starting NetLDI (for an RPC session) or when running topaz / GBS (for a linked session) will result in an error like this:

Example 1.1 Typical Error Without setmemwindow

```
unix> topaz -l
topaz> set gemstone mystone user DataCurator pass swordfish
topaz> login
---------------------------------------------------
GemStone: Error         Fatal     The session was unable to start a
cache page server on host 'myhost'.
Reason:   GemStone could not attach to the shared memory segment
with    id XXXXXXX.
        (First attach attempt at arbitrary address.)
        shmat() error = errno=13, EACCES, Authorization failure
        (permission denied)
Help:
        Operating system kernel not configured for shared memory?
        SHR_PAGE_CACHE_SIZE_KB too large for kernel configuration.
Error Category: [GemStone] Number: 4139 Arg Count: 0
topaz>
```

GemStone site administrators may wish to develop shell scripts for frequently used GemStone activities that incorporate the setmemwindow command with the appropriate window ID.

Running Multiple Stones

If multiple Stone processes are to run on a single HP platform, different window ID's must be used for each Stone. In addition, since each RPC Gem process acquires it's window ID from the NetLDI process that spawns it, each Stone will

need to have it's own NetLDI process. Requests to login an RPC Gem will need to specify the appropriate NetLDI process as part of the `gemnetid` (`gemservice` in GBS) using an appropriate NRS string.

The following example starts two Stones, *stone1* and *stone2*, running under window ID's *13* and *14*.

Example 1.2 Running Two Stones

```
unix> setmemwindow -i 13 -b startstone stone1 <other args>
unix> setmemwindow -i 13 -b startnetldi stone1netldi <other args>
unix> topaz
topaz> set gemstone stone1
topaz> set gemnetid !#netldi:stone1netldi!gemnetobject
topaz> login
successful login
topaz 1>
...
unix> setmemwindow -i 14 -b startstone stone2 <other args>
unix> setmemwindow -i 14 -b startnetldi stone2netldi <other args>
unix> topaz
topaz> set gemstone stone2
topaz> set gemnetid !#netldi:stone2netldi!gemnetobject
topaz> login
successful login
topaz 1>
```

Refer to the *GemStone System Administration Guide* for further details on naming NetLDI processes and using NRS strings to define which NetLDI process to use.

Support for Larger Extents

The maximum extent size has increased to 16 GB for all platforms. Use of this feature for extents in the file system under HP-UX and AIX requires operating system changes as described in the GemStone/S Installation Instructions. We also recommend that you pregrow such extents to a size greater than 2 GB so any configuration difficulty is detected when the server is first started.

Repository Analysis Tool

Given a collection of objects, the following method returns an array of objects representing all those reachable from the collection of objects:

```
Repository | findObjsConnectedTo: anArray
```

Additional Data Compression Support

In additional to the previous support for compression of traversal buffers, this release supports full compression between a Gem and a client. All packets between Gem and client are compressed. While this action costs some CPU cycles, it is ideal for distributed systems with low network bandwidth.

New Environment Variable for Core Files

GS_CORE_TIME_OUT

The number of seconds to wait before a catastrophically failing GemStone/S process writes a core file and terminates—by default, 60 seconds. To determine the cause of a problem, GemStone/S Technical Support needs a stack trace, usually derived from the core file. Under some circumstances, however, core files may be impractically large or otherwise unusable; in such cases, a stack trace can be extracted directly from the failing but not yet terminated process by attaching a debugger to it. Increasing the value of this variable increases the time available to attach the debugger. If you are facing this situation, GemStone/S Technical Support will recommend a new value for this variable and work with you to analyze the problem.

Automatic pstack log before coredump on Solaris

On Solaris, the HostCoredump function now forks a process to run pstack against the process that is about to dump core. HostCoredump will wait 30 seconds for the forked process to run pstack before dumping core. The pstack process will produce a `pstack`*pid*`.log` file in the current directory of the process that is about to dump core, where *pid* is the process Id of the process that is about to dump core.

New Cache Statistics

To better monitor GemStone/S operation, a number of new cache statistics have been added since GemStone/S 5.1.5:

FramesAddedToFreeList

FramesAddedToFreeList (122) is the number of frames added to the free list since the session (for Gems), shared page cache, or Stone started. Available for all slots.

GcInReclaimAll

GcInReclaimAll (139) equals 1 if the system is reclaiming pages by executing `System reclaimAll`; 0 otherwise. Available only for the Stone slot.

GcVoteUnderway

GcVoteUnderway (140) equals 1 when Gems are voting on the possible dead set, 0 otherwise. For details about the voting phase of garbage collection, see the *GemStone/S System Administration Guide.* Available only for the Stone slot.

GemsInCacheCount

GemsInCacheCount (128) is the total number of Gems using the shared page cache whose process slot you are viewing—useful for distinguishing Gems using a local shared page cache from those using a remote shared page cache. Available only for the shared page cache monitor slot.

LostOtsReceived

LostOtsReceived (166) is the number of Lost OT Root signals received and recognized by this session. Available only for Gem slots.

LostOtsSent

LostOtsSent (165) is the number of Lost OT Root signals the Stone has sent to this session, although the session may be in a sleep or I/O wait state and not yet aware of having received the signal. (See LostOtsReceived, above.) Available only for Gem slots.

OldestCrSession

OldestCrSession (167) is the session ID of a session referencing the oldest commit record. Note that more than one session may reference a commit record. A value of -1 indicates the oldest commit record is not referenced by any session. Available only for the Stone slot.

OldSpaceOverflowCount

OldSpaceOverflowCount (162) is the number of times objects were moved from old space into the NotConnectedSet because old space filled. Available only for Gem slots.

RecoverCrBacklog

RecoverCrBacklog (171) is the size of the commit record backlog that was in effect during the generation of the tranlog record currently being replayed during system recovery or restore. Available only for the Stone slot.

RecoverTranlogBlockId

RecoverTranlogBlockId (170) is the block ID of the tranlog currently being replayed during system recovery or restore. Available only for the Stone slot.

RecoverTranlogFileId

RecoverTranlogFileId (169) is the file ID of the tranlog currently being replayed during system recovery or restore. Available only for the Stone slot.

SigAbortsReceived

SigAbortsReceived (164) is the number of times the Stone has signaled this session to abort, that it has received and recognized. Available only for Gem slots.

SigAbortsSent

SigAbortsSent (163) is the number of times the Stone has signaled this session to abort, although the session may be in a sleep or I/O wait state and not yet aware of having received the signal. (See SigAbortsReceived, above.) Available only for Gem slots.

SpinLockPageFrameSleepCount

SpinLockPageFrameSleepCount (172) is Number of times the process was forced to sleep on a semaphore while attempting to acquire a page frame spin lock. Available only for shared page cache monitor slot.

SpinLockHashTableSleepCount

SpinLockHashTableSleepCount (173) is the number of times the process was forced to sleep on a semaphore while attempting to acquire a hash table spin lock. Available only for shared page cache monitor slot.

SpinLockFreeFrameSleepCount

SpinLockFreeFrameSleepCount (174) is the number of times the process was forced to sleep on a semaphore while attempting to acquire the free frame list spin lock. Available only for shared page cache monitor slot.

SpinLockFreePceSleepCount

SpinLockFreePceSleepCount (175) is the number of times the process was forced to sleep on a semaphore while attempting to acquire the free page cache entry spin lock. Available only for shared page cache monitor slot.

SpinLockSmcQSleepCount

SpinLockSmcQSleepCount (176) is the number of times the process was forced to sleep on a semaphore while attempting to acquire the SMC (shared memory communication) queue spin lock. The SMC queue allows Gems to communicate with the Stone process via shared memory. Available only for shared page cache monitor slot.

SpinLockOtherSleepCount

SpinLockOtherSleepCount (177) is the number of times the process was forced to sleep on a semaphore while attempting to acquire either the AllSymbols or shared counter spin lock. Available only for shared page cache monitor slot.

StnLoopCount

StnLoopCount (138) is an integer specifying the total number times the Stone has executed its service loop. If this number remains unchanged for a significant period (for example, ten seconds or so), the Stone has hung. Available only for the Stone slot.

TargetFreeFrameCount

TargetFreeFrameCount (168) is the minimum number of unused page frames the free frame page server(s) will attempt to keep in the cache. The free frame count can still fall below this value if the cache contains mostly dirty pages, which free frame page servers cannot preempt. Available only for the shared page cache monitor slot.

TimeInFramesFromFindFree

TimeInFramesFromFindFree (121) is the cumulative number of milliseconds that the Gem or Stone has spent scanning the shared page cache for a free frame since the session (for Gems) or Stone started. Available for all slots.

Other Experimental Statistics

In addition, a variety of new statistics have been added for internal use only; you will see them in VSD and StatMonitor displays.

Option Added to pageaudit

A **`-f`** option has been added to `pageaudit`. When present, `pageaudit` prints all errors possible; without **`-f`** the default is to stop after the first error is found. Current usage is

`pageaudit` [**`-h`**] [**`-f`**] [**`-e`** *execfg*] [**`-z`** *syscfg*] [*name*]

where

`-h`	prints usage information and exits
`-f`	keeps running beyond errors if possible
`-e`	specifies an executable-specific configuration file
`-z`	specifies a system configuration file
name	is the name of the Stone (the default is `gemserver60-audit`)

Detach Pages Control Now Automatic

All sessions now detach all pages at each commit and abort. The following configuration options are no longer supported:

GEM_DETACH_PAGES_ON_COMMIT
GEM_DETACH_PAGES_ON_ABORT

markForCollection Waits for Other GC to Finish

The method `markForCollection` now waits up to two minutes for an ongoing garbage collection task to finish before issuing a “garbage collection in progress” error.

To have the `markForCollection` wait longer than two minutes for another garbage collection process to complete, use `markForCollection:` *waitTimeSeconds*. To wait as long as necessary for the other garbage collection to complete, pass the argument “–1”. If a GcGem was stopped before it completed

processing of previously found dead objects, an argument of –1 can cause this method to wait forever, or until a GcGem session is restarted.

Documentation Now Online

GemStone/S 6.0 now includes documentation shipped as searchable PDF files as well as hardcopy manuals. The files are located in GemStone/S `doc` directory:

```
GemBuilder_C.pdf
Programming.pdf
SysAdmin.pdf
Topaz.pdf
```

Other Changes

shrinkExtents Method Obsolete

The method `shrinkExtents` is now a private method. To shrink the size of your repository, use the full backup and restore procedure documented in “How to Shrink the Repository” on page 6-12 of the *GemStone System Administration Guide.*

stopOtherSessions Timeout Shortened

Formerly, it could take as long as five minutes for a session to terminate after you sent either `stopOtherSession` or `stopSession:` if the Gem was not active. This timeout has been shortened to 60 seconds and no longer resets if multiple messages are sent.

Remote Extents and Remote Tranlogs Not Supported

Support for remote extents, remote transaction logs, and their replicates is withdrawn in this release. While we attempt to detect and prevent the use of non-local extents, we cannot 100% guarantee that our detection is foolproof. For both performance and robustness, system administrators should ensure that all extents are local to the Stone machine.

Backups to and restores from remote devices are still supported. To guarantee the write or read on a remote device is performed correctly, specify the device using an NRS string. For example, to access a device on machine “flute”:

`SystemRepository fullBackupTo:'!@flute!`*device_name*`'`

Backup and Restore Without Shared Memory Not Supported

Creation of backup files and restoring from backup files by Gems running without a shared page cache is not supported in this release.

Backup and Restore to Raw Partitions Deprecated

Support for backup to and restore from raw partitions (`.tlf` files under Windows NT and Windows 2000) is deprecated in this release. Use this functionality at your own risk. Specifying a raw partition for backup and restore may raise an error in future releases.

Operation without Shared Page Cache Deprecated

Support for Gems running without a shared page cache is deprecated in this release and may raise an error in future releases.

Calculation of SHR_PAGE_CACHE_NUM_PROCS Changed

The calculation of the default, minimum, and maximum values of SHR_PAGE_CACHE_NUM_PROCS has changed in this release to more accurately reflect the number of processes that need to attach to the shared page cache. Please refer to "Configuration Options" in Appendix A of the *GemStone System Administration Guide* for a description of the new values.

GemConnect 1.1.3 Not Available on HP-UX

GemConnect v1.1.3 is not available for systems running this release under HP-UX.

Linkable GCI Library Requires pxdb Tagging on HP-UX

On HP-UX, customer executables that load the `libgcilnk60.sl` shared library (the linkable GCI library) must have the executable tagged with `pxdb` for loading libraries private. For example:

```
$ chatr -M executable
$ /opt/langtools/bin/pxdb -s enable executable
```

Upgrade Procedure Changed

This release includes a script to convert a version 5.1.3.1 or later repository for use with GemStone/S 6.0. *Database files from releases prior to 6.0 will not work until they*

have been converted. Be sure to follow the upgrade instructions in Chapter 2 of the *GemStone/S 6.0 Installation Guide.*

Chapter 2

Bugs Fixed in GemStone/S 6.0

The following bugs have been fixed since GemStone/S 5.1.5:

System class>>sessionsReferencingOldestCr had a page leak

Execution of System class>>sessionsReferencingOldestCr caused the number of free pages in the repository to decrease by one. These pages were not freed until the repository is shut down. (#26015, fixed in v6.0)

System class>>sessionsReferencingOldestCr could crash stone

Under certain conditions, use of System class>>sessionsReferencingOldestCr could cause the stone to crash with a SIGSEGV Segmentation Violation. (#25575, fixed in v6.0)

LostOTRoot mechanism can cause various problems

The GemStone LostOTRoot mechanism could cause various problems, including page cache faults, problems with oopmap processing, and repository read failures, although these were not seen by most users. The mechanism has been reimplemented. (#24656, fixed in v6.0)

Extent size limited to 2GB

Use of extents larger than 2GB on UNIX could lead to unexpected results. When using extents larger than 2 GB, a direct byte pointer calculation could return a quantity larger than can be handled by a 32 bit integer, which could lead to unexpected results. (#19157, fixed in v6.0)

Problems with extent growth over 16GB

If not restricted by the configuration parameter DBF_EXTENT_SIZES, the stone could inadvertently extend an extent beyond the 16GB maximum, causing page referencing problems and system crashes. (#26279, fixed in v6.0)

Stone may still startup when unable to pregrow extents

When DBF_PRE_GROW is set to TRUE, if there is insufficient space to pregrow the extents to the size specified by DBF_EXTENT_SIZES, the stone would still sucessfully start, instead of printing a warning message and shutting down. (#27130, fixed in v6.0)

Adding extents of size > 2048, on NT

On NT, executing Repository>>createExtent:withMaxSize: using max size values greater than 2047 resulted in invalid negative extent sizes (or incorrectly small values, for sizes over 4096) being entered into the system.conf file. The extents were created correctly and would work as if they had the size specified in their creation. However, the stone would not restart with negative or zero extent sizes specified in the system.conf, or with extents sizes specified smaller than the actual size of the extent. (#26800, fixed in v6.0)

ETIMEDOUT disconnects on clients outside a firewall

A firewall with a short socket timeout interval, located between a gem and smalltalk client, will inadvertently shutdown one of the 2 socket connections between the gem and client due to lack of traffic. (#27077, fixed in v6.0)

PositionableStream>>upToAll: sometimes produced incorrect results

The implementation of PositionableStream >> upToAll: could produce incorrect results in certain cases. (#23515, fixed in v6.0)

Large frameAge values locked pages in cache

Under certain circumstances, a very large value that was not a valid frameAge value could lock the page in the shared page cache until the value could eventually be whittled down to zero by the frame aging mechanism. (#25761, fixed in v6.0)

CharacterCollection>>subStrings: ignored null strings

The method subStrings: appeared to skip elements if there is no space between the character separator. For example, `'Before,,,,After' subStrings: $` returned `#('Before' 'After')` instead of `#('Before' '' '' '' 'After')`. You may need to change your code if it does not expect empty strings from multiple separators. (#18152, fixed in v6.0)

SmallInteger method always returned 0 when the receiver is 0

The method SmallInteger>> / always returned 0 when the receiver is 0, even when the argument is 0. This was different from the behavior of other number classes, which always raise an exception or answer PlusQuietNaN when the argument is zero. (#21578, fixed in v6.0)

GsFile>>skip: does allow negative counts

The method comment said that the count could be zero but not negative. The comment has been corrected to match the behavior, which was correct for negative counts. (#13988, fixed in v6.0)

Backup during reclaim can lead to "Bad Data Page" errors

If a backup was started while reclaim was going on, you could find error messages of the form: “Bad Data Page encountered in backup.” (#20361, fixed in v6.0)

Checkpoints less frequent than documented

Checkpoints did not occur after the time set in the parameter STN_CHECKPOINT_INTERVAL. The system waited until 100 commits had been processed subsequent to the passage of the STN_CHECKPOINT_INTERVAL before the checkpoint was performed. (#19253, fixed in v6.0)

Temporary performance impact when machine with remote cache crashes

If a Stone or a Gem attempted to access a page server and the page server machine had crashed or was no longer accessible to the network, it would be unresponsive for nine minutes. This was due to a TCP timer which continued trying to read / write for nine minutes before it returned ETIMEDOUT. The timeout is now configurable using configuration option STN_REMOTE_CACHE_PGSVR_TIMEOUT.(#13329, fixed in v6.0)

Index corruption can prevent removing index

Certain kinds of index corruption, which resulted in object not found errors, had to be fixed by dropping and recreating the index. Unfortunately error checking in code called by removeEqualityIndex: caused a walkback when trying to drop the index as well. (26568, fixed in v6.0)

"unable to resolve symbol: 'reclaimDeadEnabled'" appears in gcgem log of upgraded repositories

The message: "Error: unable to resolve symbol: 'reclaimDeadEnabled', using default" could appear in the GcGem log for repositories that had been upgraded. The error was innocuous and did not indicate any problems with your repository. (#19218, fixed in v6.0)

Object does not exist error when doing SelectBlock queries

SelectBlocks embedded within another block inside of a method could be prematurely garbage collected during method execution, causing "object does not exist" errors when referencing the now non-existent SelectBlock. (#26692, fixed in v6.0)

Raw partitions on AIX overwrote Logical Volume Manager

AIX raw partitions use the first 512 bytes of the disk for the Logical Volume Manager. GemStone / S raw partitions used this area for extents and transaction logs, leading to operating system problems in managing the partition.

On AIX, GemStone / S 6.0 skips the first 512 bytes of the raw partition so as not to interfere with the AIX Logical Volume Manager. (#23845, fixed in v5.1.5.1)

Using kill -TERM with RPC Gems could cause stuck spin lock

A *spin lock* is a location in memory indicating the availability of a resource, useful for serializing operations in a faster, more efficient manner than with a semaphore. The shared page cache monitor uses spin locks to allocate and deallocate pages for Gems and the Stone.

If you kill a Gem process at the operating system level, you run a small but real risk that the killed process is holding a spin lock at that moment. If that is the case, the Stone will fail.

For this reason, we recommend never using kill -9 to stop a Gem process, but instead using kill -TERM. In previous releases, this mechanism could also cause problems, but it has now been made reliable. (#24006, fixed in v5.1.5.1)

Backups during GcGem reclaim could lead to thrashing

During backups, under rare circumstances the GcGem could appear to thrash with an extremely high commit rate, while no pages were reclaimed. As the backup proceeds, the blocked pages became available for reclaim and the GcGem stopped thrashing. (#24400, fixed in v5.1.5.1)

Version incompatibilities between Gem and Stone

Accidentally trying to log into a 5.1.5 Stone with a Gem of an older version (either linked or RPC), could hang or crash the Stone. (#24867, fixed in v5.1.5.1)

Bugs Fixed Since the Beta Release

The following bugs that were reported in the Beta release notes have now been fixed:

Client disconnect during user action causes SIGSEGV in gem

If a Gem was running a user action and the client disconnected while the user action was running, the gem died with a SIGSEGV error. As part of the disconnect, the user action libraries were unloaded, which caused problems if there was user action code on the stack at that time. (#26464)

Remote copydbf of backup files to AIX raw partitions appears to fail

NOTE
Backup to and restore from raw partitions is deprecated.

Performing a remote copydbf of a backup file to an AIX raw partition erroneously reported an error when verified using 'copydbf -i', and failed on remote restore. The resulting backup could be restore locally. (#26531)

Not using shared memory may cause data corruption

Not using shared memory with the GS 6.0 Beta release could cause data corruption. (#26398)

SigAbort in middle of multi-file backups can be missed

During processing of a multi-file backup, sigAborts were handled properly by the backup code. However, if the sigAbort occurred between files, when control had been transferred back to topaz, it could be missed. On a busy system, the session not aborting could lead to a commit record backlog. (#26211)

Install script produces errors

The script "installgs" produced a number of syntax errors when checking file permissions on /opt/gemstone, /opt/gemstone/locks and /opt/gemstone/log. (If /usr/gemstone already exists, it will check this instead of /opt.) Even if permissions are correct, the script prompted to change them.

The script also produced an error when trying to execute "fixman". (#26519)

Failure of linked logins from GBS on HPUX

Attempting a linked login from GBS on HPUX produced GS error 4139. Linked logins from GBS on HPUX are no longer supported. (#26228)