GemStone®

GemStone/S 64 Bit™ Release Notes

Version 3.2

April 2014

GemTalk Systems
15220 NW Greenbrier Parkway
Suite 240
Beaverton, OR 97006

Preface

About This Documentation

These release notes describe changes in the GemStone/S 64 Bit™ version 3.2 release. Read these release notes carefully before you begin installation, conversion testing, or development with this release.

For information on installing or upgrading to this version of GemStone/S 64 Bit, please refer to the *GemStone/S 64 Bit Installation Guide* for version 3.2.

These documents are available on the GemTalk support website, as described below.

Terminology Conventions

The term "GemStone" is used to refer to the server products GemStone/S 64 Bit and GemStone/S, and the GemStone family of products; the GemStone Smalltalk programming language; and may also be used to refer to the company, now GemTalk Systems, previously GemStone Systems, Inc. and a division of VMware, Inc.

Technical Support

Support Website

http://gemtalksystems.com/techsupport

GemTalk's Technical Support website provides a variety of resources to help you use GemTalk products:

- **Documentation** for released versions of all GemTalk products, in PDF form.
- **Downloads**, including current and recent versions of GemTalk products.
- **Bugnotes**, identifying performance issues or error conditions that you may encounter when using a GemTalk product.

- **TechTips**, providing information and instructions that are not in the documentation.
- **Compatibility matrices**, listing supported platforms for GemTalk product versions.

This material is updated regularly; we recommend checking this site on a regular basis.

Help Requests

You may need to contact Technical Support directly, if your questions are not answered in the documentation or by other material on the Technical Support site. Technical Support is available to customers with current support contracts.

Requests for technical assistance may be submitted online, by email, or by telephone. We recommend you use telephone contact only for more serious requests that require immediate evaluation, such as a production system down. The support website is the preferred way to contact Technical Support.

Website: http://techsupport.gemtalksystems.com

Email: techsupport@gemtalksystems.com

Telephone: (800) 243-4772 or (503) 766-4702

When submitting a request, please include the following information:

- Your name and company name.
- The versions of GemStone/S 64 Bit and of all related GemTalk products, and of any other related products, such as client Smalltalk products.
- The operating system and version you are using.
- A description of the problem or request.
- Exact error message(s) received, if any, including log files if appropriate.

Technical Support is available from 8am to 5pm Pacific Time, Monday through Friday, excluding GemTalk holidays.

24x7 Emergency Technical Support

GemTalk offers, at an additional charge, 24x7 emergency technical support. This support entitles customers to contact us 24 hours a day, 7 days a week, 365 days a year, for issues impacting a production system. For more details, contact GemTalk Support Renewals.

Training and Consulting

GemTalk Professional Services provide consulting to help you succeed with GemStone products. Training for GemStone/S is available at your location, and training courses are offered periodically at our offices in Beaverton, Oregon. Contact GemTalk Professional Services for more details or to obtain consulting services.

Contents

Chapter
1

GemStone/S 64 Bit 3.2 Release Notes

Overview

GemStone/S 64 Bit 3.2 is a new version of the GemStone/S 64 Bit object server. This release provides new features, API changes and feature enhancements, and fixes a number of bugs; we recommend everyone using GemStone/S 64 Bit upgrade to this new version.

These release notes provide changes between the previous version of GemStone/S 64 Bit, version 3.1.0.5, and version 3.2. If you are upgrading from a version prior to 3.1.0.5, review the release notes for each intermediate release to see the full set of changes. In particular, if you are upgrading from version 2.4.x, note that there were substantial changes in v3.0 that impact your application.

Users of older versions of GemBuilder® for Smalltalk (GBS), GemConnect®, and GemBuilder for Java (GBJ) will need to upgrade these products as well.

For details about installing GemStone/S 64 Bit 3.2 or upgrading from earlier versions of GemStone/S 64 Bit, see the *GemStone/S 64 Bit Installation Guide* for v3.2 for your platform.

Supported Platforms

Platforms for Version 3.2

GemStone/S 64 Bit version 3.2 is supported on the following platforms:

- Solaris 10 and 11 on SPARC
- Solaris 10 on x86
- AIX 6.1, TL1, SP1, and AIX 7.1
- Red Hat Linux ES 6.1 and Ubuntu 10.04, on x86
- Mac OSX 10.6.4 (Snow Leopard), with Darwin 10.4.0 kernel, on x86

For more information and detailed requirements for each supported platforms, please refer to the GemStone/S 64 Bit v3.2 Installation Guide for that platform.

Other Product Version Updates

GBS Versions

The following version of GBS is supported with GemStone/S 64 Bit version 3.2. You must use GBS version 7.6.1 or later for VisualWorks, or 5.4.2 or later for VA Smalltalk, with GemStone/S 64 Bit v3.2.

GBS version 7.6.1

VisualWorks 7.10 32-bit	VisualWorks 7.10 64-bit	VisualWorks 7.9.1 32-bit
‣ Windows 8, Windows 2008 R2 and Windows 7 ‣ Solaris 10 on SPARC ‣ Ubuntu 10.04 and RedHat Linux ES 6.1	‣ Windows 8, Windows 2008 R2 and Windows 7 ‣ Solaris 10 on SPARC ‣ Ubuntu 10.04 and RedHat Linux ES 6.1	‣ Windows 2008 R2 and Windows 7 ‣ Solaris 10 on SPARC ‣ Ubuntu 10.04 and RedHat Linux ES 6.1

GBS version 5.4.2

VA Smalltalk 8.6	VA Smalltalk 8.5.2
‣ Windows 8, Professional or above ‣ Windows 2008 R2 ‣ Windows 7, Professional or above	‣ Windows 2008 R2 ‣ Windows 7

For more details on supported GBS and client Smalltalk platforms and requirements, see the *GemBuilder for Smalltalk Installation Guide* for that version of GBS.

Server changes impacting GBS

Changes in the shared library distribution will affect GBS logins from Windows; see "Windows distribution includes 64-bit libraries, path change required" on page 19.

Linked GBS logins require loading a different library; see "New library required for linked logins from GBS" on page 19.

The GBS tools rely on services provided by server tools. Server changes that impact GBS are included in the *GemBuilder for Smalltalk Release Notes* for version 7.6.1 and 5.4.2. Among the changes that impact GBS are changes in stack frame display, as seen in the debugger; see page 55.

Version 3.2 also includes several bug fixes that impact GBS specifically; see "Bugs primarily affecting GBS" on page 91.

GemConnect Version

GemConnect requires an updated versions for GemStone/S 64 Bit v3.2.

If you are using GemConnect, you should install the GemConnect version 2.3 after performing the upgrade to GemStone/S 64 Bit v3.2.

You should also ensure that your new keyfile has the appropriate permissions.

See the *GemConnect Release Notes* for v2.3 for details on the changes, and the *GemConnect Installation Guide* for v2.3 for System requirements and installation instructions.

GemBuilder for Java Version

GemBuilder for Java (GBJ) requires an updated versions for GemStone/S 64 Bit v3.2.

If you are using GBJ, you should install the GemBuilder for Java version 3.1.1 after performing the upgrade to GemStone/S 64 Bit v3.2.

You should also ensure that your new keyfile has the appropriate permissions.

See the *GemBuilder for Java Release Notes* for v3.1.1 for details on the changes, and the *GemBuilder for Java Installation Guide* for v3.1.1 for System requirements and installation instructions.

Documentation

GemStone/S 64 Bit version 3.2 includes a complete set of updated documentation.

The format has been changed to be a 8.5x11 letter size, matching the Release Notes and Installation Guide formatting, for more efficient end-user printing.

The changes described in these release notes have been incorporated into the appropriate manuals, along with other corrections and improvements.

In addition, the following are some significant changes:

Programming Guide

- The Querying and Indexing chapter has been rewritten to include the new Indexing API, with more detail than is in these release notes.
- The material on Strings from the Collections chapter has been abstracted into a new Chapter, describing Characters and traditional and Unicode strings and collation.
- The Advanced Class Protocol chapter has been revised, with material added on ClassOrganizer and Deprecation.
- New chapters added on System Sets and External Sessions, with more detail than provided in these release notes.

System Administration Guide

- Chapter 12, Managing Growth, has changes related to the new GcGem configuration and multi-threaded reclaim; the reclaim process is significantly simplified.
- Appendix F removed: Character Data Table and Decimal point internationalization material moved to Programming Guide; Date localization for log files moved to Chapter 5.
- Appendix G and H removed; VSD and cache statistics material moved to the new VSD manual.

VSD

- New manual, also available separately from the server distribution. The material was previously an appendix in the System Administration Guide, now significantly rewritten and improved.
- This manual now includes the cache statistics descriptions for 32-bit GemStone/S as well as GemStone/S 64 Bit.

Installation Guides

- The instructions for updating shared libraries for GBS has been reorganized. Each platform’s Installation Guide includes instructions for GBS clients on that platform only. Specifics on how to configure GBS are now in the GBS Installation Guide.

GemStone/S v3.2 - Enhancements and Changes

1. Keyfile changes

New keyfiles required

New keyfiles are required for version 3.2. To obtain a new keyfile for GemStone/S v3.2, write to keyfiles@gemtalksystem.com. In your request, include your license information, platform and any updates to contact information. It may be helpful to include your current keyfile with the request. Contact GemTalk Technical Support if you have issues or questions.

Changes in limits of Free GS/S Web Edition (Seaside) keyfile

The GemStone/S 64 Bit distribution includes a limited keyfile, which is distributed with the Seaside product. The limits on this keyfile have changed in v3.2, and new low-cost Web Edition licensing is available. For more information on repository limits and pricing, see:

```
http://seaside.gemtalksystems.com/docs/WebEditionPricing.htm
```

If you have further questions on licensing options, contact GemTalk Sales at sales@gemtalksystems.com.

GemConnect and GBJ now managed by keyfiles

GemStone keyfiles now include access control for GemConnect and GBJ. If you are using these products, please note this in your request for a 3.2 keyfile.

Note that the Web Edition keyfiles do not provide access to GemConnect and GBJ, so you will not be able to use GemConnect or GBJ with that keyfile.

2. Login and Authentication changes

PAM now used for UNIX authentication

All host userid authentication now uses PAM (Pluggable Authentication Module). For host userId authentication, PAM must be installed and configured.

Host userId authentication is used:

- When running netldi other than in guest mode; i.e. when netldi is running as root, or when all logins use a single unix userid. PAM is not directly accessed when running in guest mode with captive account, since in this case users supply their host login credentials.
- When GemStone login is configured to use UNIX for authentication, rather than traditional GemStone userId and password, or authentication directly to LDAP.

The PAM service names used by GemStone are `gemstone.gem` for gems and `gemstone.netldi` for netldi. See the *Installation Guide* for more details on how to configure PAM on each platform.

The method `System >> validatePasswordForUser:password:isEncrypted:` is deprecated, since PAM cannot validate encrypted passwords. It is replaced with the new method `System >> validatePasswordForUser:password:`.

To debug PAM login issues, you can set the new environment variable GS_DEBUG_PAM to any value. This will cause login debugging information (other than passwords) to be written to stdout.

Support for authenticated binds with LDAP

For system configurations that do not allow anonymous LDAP binds, it is now possible to pass in the connection name and password for the LDAP bind, to validate userId and password. The following method has been added:

```
System class >> validatePasswordUsingLdapServers:baseDn:
   filterDn:userId:password:bindDn:bindPassword:
```

The additional bind arguments should be nil if validating in explicit mode, or with anonymous bind. For example:

Explicit mode

```
   System validatePasswordUsingLdapServers:
          (Array with: 'ldaps://myldap.mydomain.com')
      baseDn: 'uid=%s,ou=Users,dc=mycompany,dc=com'
      filterDn: nil
      userId: 'MyUserId' password: 'swordfish'
      bindDn: nil bindPassword: nil
```

Search mode with anonymous bind

```
   System validatePasswordUsingLdapServers:
          (Array with: 'ldaps://myldap.mydomain.com')
      baseDn: 'ou=Users,dc=mycompany,dc=com'
      filterDn: '(uid=%s)'
      userId: 'MyUserId' password: 'swordfish'
      bindDn: nil bindPassword: nil
```

Search mode with authenticated bind

```
   System validatePasswordUsingLdapServers:
          (Array with: 'ldaps://myldap.mydomain.com')
      baseDn: 'ou=Users,dc=mycompany,dc=com'
      filterDn: '(uid=%s)'
      userId: 'MyUserId' password: 'swordfish'
      bindDn: 'LdapBindUser' bindPassword: 'LdapBindPassword'
```

For further details, see the method comments in the image.

Changes in UserProfile rename using userId:

Changing the userId of an account was previously only allowed by SystemUser; now, this ability can be provided to other user profiles with a new privilege.

New Privilege

A new privilege has been added, **ChangeUserId**, which controls if a user account can be renamed (if the user id of a UserProfile can be changed) by a session logged in as a user other than SystemUser. By default, DataCurator and other accounts do not have this privilege.

When this privilege is given to a UserProfile, that user can invoke userId:password: on another users's account.

userId: API change

Renaming a userId unsets the account's password. To avoid inadvertently allowing a security gap, rather than using userId:, the method userId:password: should be used.

The method `userId:`, which was previously deprecated, has been removed in this release.

3. Interprocess, library changes

Windows distribution includes 64-bit libraries, path change required

The GemStone distribution now includes 64-bit Windows libraries, which can be loaded into 64-bit VisualWorks for Windows. This allows RPC logins from 64-bit VW on Windows; linked logins are not possible on Windows, since the required server components are unavailable.

The 64 bit libraries are distributed in the %GEMSTONE%\bin directory. The 32-bit libraries are now in a new directory, %GEMSTONE%\bin32. There are further changes in the distribution; most utilities are provided in both 64-bit and 32-bit versions, and VSD is now distributed.

Note that this \bin to \bin32 name change means existing %PATH% and `libraryName:` settings may need to update the directory name to include "32", as well as the usual changes for a new release.

GemStone has updated the compile platform for the Windows shared libraries to Windows 7. With this release, Windows XP is no longer a supported platform for clients, including GBS. Windows XP has reached end of life with Microsoft.

New library required for linked logins from GBS

When logging in linked from GBS, in order to correctly load the set of client shared libraries, the $GEMSTONE environment variable must be defined, and you now must load a new library:

```
libgbslnk-3.2.0-64.so
```

The other client libraries are also required, and will be loaded from $GEMSTONE/lib. The new library will load the other libraries.

Change in internal sockets and role of netldi

The connection between GBS or another GCI application and a gem process has been simplified, so only one socket is used (rather than the previous regular and OOB sockets).

This single socket is now inherited by the gem from netldi. Previously, connections on another socket were used for the gem login. Since the netldi no longer creates a separate socket, the startnetldi -p option is no longer required and has been removed.

When a linked GCI application will be running on a node that is remote from the stone, it is no longer required to have a netldi running on that node. A netldi is used if it is present, and is still required to spawn a gem process. Otherwise, the first linked login will start the remote cache directly.

The page manager is no longer a separate process, it is now a thread in the stone. It still creates its own specific log file, to avoid cluttering the stone log.

Upgraded versions of open source libraries

GemStone relies on a number of open source libraries. There have been version updates in these libraries:

- OpenLDAP is now version 2.4.33
- zlib is now version 1.2.8
- openSSL is now 1.0.1g
- libICU is now 51.2

Open source licences change in distribution

The licenses for open source packages are now distributed in the new directory $GEMSTONE/licences, in individual files. Additional licenses are included in v3.2.

4. Improved support for SPC large memory pages

Starting with version 3.1.0.2, GemStone provided support for large memory pages in the SPC via an environment variable, GS_SPC_USE_LARGE_PAGES. In version 3.2, the way this is specified has been changed and now uses a configuration parameter, SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_POLICY. This operates substantially the same as the environment variable. The environment variable GS_SPC_USE_LARGE_PAGES is no longer used.

The default setting for SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_POLICY is not to use large memory pages. The following options are provided:

- O (default): large memory pages are not used.
- 1 (advisory mode): large memory pages are requested, but the cache still starts if the request is not granted by the system. **This setting is not recommended on AIX**, particularly with large caches, since later permission failures can result in memory overcommit.
- 2 (mandatory mode): large memory pages are requested and the cache fails to start if the request is not granted by the system.

For details on this configuration parameter, see "SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_POLICY" on page 72.

Large page support is application and hardware dependant, and is only supported on AIX and Linux. Solaris automatically uses large pages, if available, so this features is not relevant on, and not provided for, Solaris.

A large page size of 16MB on AIX and 2MB on Linux is used.

When you start GemStone, the shared page cache monitor log will use a line of the form:

```
[Info]: Shared page cache was successfully created using large
memory pages.
```

The SPC monitor process will print extra debugging information to its log file if the following environment variable is set:

```
export GS_DEBUG_SHARED_MEM=1
```

Determining the required number of large memory pages

You will first need to determine how much space is needed for the shared page cache. This can be computed using utilities in the shared page cache monitor shell.

1. Note the desired shared page cache size, maximum number of GemStone processes, and the number of shared counters you will be using.

1. Start the shared page cache monitor shell

   ```
   unix> $GEMSTONE/sys/shrpcmonitor
   ```

   This displays header information and leaves you at the shell prompt, `SHRPCMON>`.

2. Enter the value from step 1. The shrpcmon shell uses postfix notation; the operators to set the requires values are shown in the example below.

   This example is for an (approximately) 2GB shared page cache, 200 processes, and 1900 shared counters:

   ```
   SHRPCMON>2000000 setcachesizekb
   SHRPCMON>200 setnumprocs
   SHRPCMON>1900 setnumsharedctrs
   ```

3. The command `getrequiredsize` computes the large page information. This command computes the number of pages from the memory required, rounds up for alignment, and divides by the OS large page size to determine the number of large pages you need to configure.

   ```
   SHRPCMON>getrequiredsize
   For 125000 pages, 200 processes and 1900 shared counters,
   required cache size is 2166276096 bytes.
   To use 2 MB large memory pages: 2166358016 bytes are required
   for alignment.
   Number of 2 MB large pages required: 1033
   ```

   This example is for Linux, and calculates that the OS will require 1033 2-MB pages. The process is the same on AIX, although the results are different since AIX uses 16 MB large pages. For the given input values, on AIX, 130 pages would be required.

Configure Linux to use the required number of Large Memory Pages

Before large memory pages can be used, Linux must be configured to use a specific number of large memory pages. These commands must be run as root.

To enable large pages until the next system reboot:

```
UNIX> echo numPages > /proc/sys/vm/nr_hugepages
```

To permanently enable large pages:

```
UNIX> echo "vm.nr_hugepages=numPages" >> /etc/sysctl.conf
```

You must also provide the `cap_ipc_lock` capability to the SPC monitor:

```
UNIX> /sbin/setcap cap_ipc_lock=pe $GEMSTONE/sys/startshrpcmon
UNIX> /sbin/setcap cap_ipc_lock=pe $GEMSTONE/sys/shrpcmonitor
```

Alternatively, the SPC monitor process can be run with an effective user ID of root:

```
UNIX> chown root $GEMSTONE/sys/shrpcmonitor
   $GEMSTONE/sys/startshrpcmon
UNIX> chmod u+s $GEMSTONE/sys/shrpcmonitor
   $GEMSTONE/sys/startshrpcmon
```

You can query the status of large pages using:

```
UNIX> grep Huge /proc/meminfo
```

Configure AIX to use the required number of Large Memory Pages

AIX by default is not configured to allocate large memory pages. Before large memory pages can be used, AIX must be configured to use a specific number of large memory pages. These commands must be run as root.

the following example command configures AIX with 64185 large (16MB) memory pages:

```
UNIX> vmo -r -o lgpg_regions=numPages -o lgpg_size=16777216
UNIX> vmo -p -o v_pinshm=1
```

After this has been executed, the bosboot command must be run to build a new kernel image, and the system must be rebooted

The UNIX user running the shared cache monitor must also be given permission to use large memory pages.

```
UNIX> chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE
<user id>
```

Alternatively, you can run the SPC monitor process with an effective user ID of root:

```
UNIX> chown root $GEMSTONE/sys/shrpcmonitor
   $GEMSTONE/sys/startshrpcmon
UNIX> chmod u+s $GEMSTONE/sys/shrpcmonitor
   $GEMSTONE/sys/startshrpcmon
```

You can confirm large memory pages are available for use using `vmstat -l`.

5. Deprecation changes

Better control over deprecation configuration

Version 3.0 introduced mechanisms for GemTalk to deprecate individual methods and to allow customers to configure their applications to detect any calls to deprecated methods.

Previously, changing the handling required recompiling a method as SystemUser. Now, this can be done by any user with write access to the DataCurator object security policy.

When deprecated methods are executed, it can be configured to do nothing, raise an error, or write information to a log file. By default, no action occurs when a deprecated method is invoked.

Enabling and disabling deprecation handling is repository wide; for example, if you configure deprecations to raise an error, and commit, any user who executes a deprecated method in this repository will see an error.

When configured to write the deprecation error or the error and the stack to a log file, each session creates its own deprecation log file. This log is named Deprecated*PID*.log, and is in the same location as a the gem log for an RPC login, or in the current working directory.

`Deprecated class >> doNothingOnDeprecated`
This is the default, and this is invoked to turn deprecations actions off. There is no impact on calling a deprecated method.

`Deprecated class >> doErrorOnDeprecated`
After this is executed, any invocation of a deprecated method will raise an exception.

`Deprecated class >> doLogOnDeprecated`
After this is executed, invocation of deprecated methods will write the error to the session's deprecation log file.

`Deprecated class >> doLogStackOnDeprecated`
After this is executed, invocation of deprecated methods will write the error and the call stack to the session's deprecation log file.

Cleanup of internal calls to deprecated methods

The remaining references to deprecated messages have been corrected in version 3.2. In addition, a number of deprecation messages have been enhanced to provide more information.

Some methods may include a comment that they are deprecated. If they do not invoke deprecated:, they are not officially deprecated, but are expected to be deprecated in a future release.

Checking for deprecated methods as part of upgrade process

GemTalk will be removing some of the obsolete methods that have been deprecated over the past several releases. Once these methods are removed, it is too late for the deprecation mechanism to be helpful.

After upgrading to 3.2, or before a subsequent upgrade, you should enable deprecation errors or logging in your application environment, to verify that you are not relying on deprecated methods that are at risk for removal. This will allow you to make updates to these calls as required for GemStone upgrade.

Release notes for each version include both newly deprecated and removed methods. You may find all deprecated methods by searching the image for senders of `deprecated:`.

Deprecated methods

The following methods, not otherwise mentioned in these release notes, have been deprecated in this release.

```
Behavior >> nameForFileout
ExecBlock >> numberArgs
IcuCollator >> hiraganaQuarternary:
Repository >> flushAllExtents
Repository >> auditWithLimit:
Repository >> quickObjectAuditWithLevel:
Repository >> repairWithLimit:
Repository >> setArchiveLogDirectories:
Repository >> setArchiveLogDirectory:
TraversalBuffer >> actualBufferSize
TraversalBuffer >> incrementActualSize:
TraversalBuffer >> segment
TraversalBuffer >> setSegmentId:
String >> decodeFromUTF8
GsFile nextLineInto:startingAt:
GsFile >> +
Float >> asSmallFloat
Number >> asSmallFloat
ClusterBucket class >> newForExtent:
ClusterBucket >> extentId:
MultiByteString >> asDoubleByteSymbol
DoubleByteSymbol >> asDoubleByteSymbol
System class >> standbyVmSessionId
System class >> startStandbySession
System class >> stopStandbySession
```

The classes ComplexBlock, ComplexVCBlock, ExecutableBlock, and SimpleBlock, which are not used since the reimplementation of blocks in 3.0, have been moved to ObsoleteClasses.

Private methods (methods that begin with an underscore) may also invoke deprecated:. These methods are not included in the Release Notes, since private methods may be removed without warning.

Collating Table sort

The following methods relied on a older collating mechanism that does not work properly for characters over 255. This sorting mechanism is deprecated, and bugs in this code are not being fixed. Unicode strings should be used instead, when control over collation is needed.

The following methods are deprecated in String and in subclasses of MultiByteString:

```
equals:collatingTable:
greaterThan:collatingTable:
greaterThanOrEqual:collatingTable:
lessThan:collatingTable:
lessThanOrEqual:collatingTable:
```

This method is deprecated in Object and all Subclasses:

```
asciiLessThan:
```

Removed Deprecated methods

The following previously deprecated methods have been removed in this release:

```
Array >> at:caseInsenstiveEquals:
Array >> at:caseSenstiveEquals:
```

These misspelled method names were replaced by methods with the correct spelling in v2.0.

The method `UserProfile>>usedId:` has also been removed; see "userId: API change" on page 19.

Un-deprecated methods

DateTime methods `hours`, `hoursGmt`, `minutes`, `minutesGmt`, `seconds`, `secondsGmt`, `monthOfYear`, and `monthOfYearGmt` were deprecated in previous releases, but are no longer deprecated. While these are duplicate methods, they will be retained in the image for convenience.

6. ClassOrganizer improvements

ClassOrganizer is a utility class that provides services to search for classes, methods, method categories, and other information for the image or for a set of classes within the image. This class is used by GBS to perform server queries from the browsers.

This class has more meaningful method names, method comments, improved method categories, and changes in behavior from some methods. Some documentation has been added to the Programming Guide, although the image method comments are the primary documentation for this class.

Scope of root class

There is a change in behavior that applies for ClassOrganizer that are created with a root class other than Object. Now, an instance of ClassOrganizer includes all classes in the specified symbolList which are subclasses of the rootClass, plus all superClasses of rootClass up to Object. ClassOrganizer protocol such as those to return the senders and implementors of a selector, will query over rootClass, all its subclasses, and superclasses up through Object.

Methods added to get references to literal

The following methods have been added, to collect GsNMethods that reference a literal.

```
ClassOrganizer >> referencesToLiteral:in:
ClassOrganizer >> referencesToLiteral:
```

7. ProfMonitor Changes

In addition to the listed changes, ProfMonitorTree and ProfMonitor methods include some refactoring and internal method changes.

Object creation tracking enhancements

Object creation tracking has been enhanced significantly in ProfMonitor and ProfMonitorTree. Object creation tracking now includes tracking of creation for both persistent and transient objects, hidden internal objects such as LargeObjectNodes. Object creation tracking now provides the full level of object creation call stack, rather than just the two levels provided in previous releases. Object creation tracking is provided with creation method on top, and in tree form when using ProfMonitorTree.

Access has been provided to set the sampleDepth of a ProfMonitor or ProfMonitorTree. This may be useful in limiting the results for object creation tracking to get results comparable to previous releases; however, note that this also limits the method senders stack results.

Report of overruns

When high-resolution profiling is performed (using the intervalNs: keyword), the VM may miss samples because it cannot service the profiling interrupts fast enough.

If there are such overruns during a profiling, the report will now include a line describing the overrun count.

For example:

```
profiling overun count 690 , total overrun time = 6.74% of
sampling time
```

Added ProfMonitor method

The following method has been added:

`reportAfterRunDownTo:` tally
Formats and returns a string holding a report of the receiver's most recent profile run.

8. Multi-threaded reclaim and MGC

Multi-threaded reclaim

Previously, reclaim was performed by one or more Reclaim GcGem processes, each of which were responsible for reclaim within one or more extents. Now, there is a single Reclaim Gem, which is multi-threaded; individual threads within the Reclaim Gem process act as lightweight gem sessions. These sessions are no longer assigned to a specific extent or extent range; the Reclaim Gem more efficiently manages reclaiming pages across all extents.

If your reclaim performance is not limited by disk I/O, multi-threaded reclaim with multiple Reclaim Gem Sessions should provide significant performance improvement, as well as being easier to administer.

Configuring the number of reclaim sessions

Multiple reclaim sessions can perform reclaim over a single extent, so the number of reclaim sessions is no longer limited to the number of extents in a specific image. The total number of reclaim sessions cannot be larger than 256.

The configuration parameter #STN_NUM_GC_RECLAIM_SESSIONS now configures the number of reclaim session within the single Reclaim Gem; the resulting behavior is substantially the same as in previous releases.

The new configuration parameter #STN_MAX_GC_RECLAIM_SESSIONS (see page 73) specifies the largest number of reclaim sessions that will ever be running on this system. This is used to size internal structures. By default, this is the number of extents. You cannot specify a setting for #STN_NUM_GC_RECLAIM_SESSIONS that is larger than #STN_MAX_GC_RECLAIM_SESSIONS.

The number of configured reclaim sessions can be changed at runtime using #StnNumGcReclaimSessions; the maximum number can only be changed when the stone is restarted.

Changing the number of reclaim sessions

On startup, the number of reclaim sessions started within the Reclaim Gem is #STN_NUM_GC_RECLAIM_SESSIONS.

During runtime, you may change the number of reclaim sessions using the new method

```
System >> changeNumberOfReclaimGemSessions:
```

This will stop or start sessions until the number of sessions equals the requested count. While you should limit this to be not more than #STN_MAX_GC_RECLAIM_SESSIONS, if you specify a larger number it may use some user sessions to provide additional reclaim sessions; this will reduce the number of user sessions that can be logged in. This method returns the number of sessions it will target to have running. As in previous releases, GcGem startup methods do not block, so it may take a little while for the actual number of reclaim sessions to be started or stopped.

You may start the maximum configured number of Reclaim Gem sessions using the new method:

```
System class >> startMaxReclaimGemSessions
```

Alternately, you can stop the Reclaim Gem, change the configured number of Reclaim Gem sessions using the runtime configuration parameter #StnNumGcReclaimSessions, and restart the Reclaim Gem, as in previous versions. This will not allow you to specify more than STN_MAX_GC_RECLAIM_SESSIONS sessions.

Renamed methods to manage Reclaim, Admin, and Symbol Gems

Since a single a Reclaim Gem can manage a number of individual sessions performing the reclaim, the names of the image methods that manage reclaim sessions were unclear and potentially confusing.

For consistency, the methods that start and stop system gems and get information on system gems have been renamed in this release. The old methods are deprecated.

Deprecated Method	Replace by
`startAllReclaimGcSessions`	`startReclaimGem`
`stopAllReclaimGcSessions`	`stopReclaimGem`
`startReclaimGemForExtentRange:to:`	`startReclaimGem`
`startAdminGcSession`	`startAdminGem`
`stopAdminGcSession`	`stopAdminGem`
`adminGcGemSessionId`	`adminGemSessionId`
`startAllGcSessions`	`startAllGcGems`
`stopAllGcSessions`	`stopAllGcGems`
`startSymbolCreationSession`	`startSymbolGem`
`stopSymbolCreationSession`	`stopSymbolGem`
`symbolCreationSessionId`	`symbolGemSessionId`

The methods `reclaimGemSessionId` and `reclaimGemSessionIds` have been added.

Convenience method to start GC Gems

A method has been added:

```
System class >> ensureGcRunning
```

which starts both Admin and Reclaim Gem, if they are not running, and waits for up to 60 seconds for them to complete startup, otherwise returning an error.

Deprecated Reclaim Methods

Since there is no longer a link between extents and Reclaim Gem, the following methods are deprecated:

```
System class >> currentGcReclaimSessionsByExtent
System class >> numberOfExtentsWithoutGC
System class >> numberOfExtentRangesWithoutGC
Repository >> createExtent:withMaxSize:startNewReclaimGem:
```

Reclaim sessions can no longer be run on remote hosts. The following related methods are deprecated:

```
System class >> startReclaimGemForExtentRange:to:onHost:
System class >> startReclaimGemForExtentRange:to:onHost:
                     stoneHost:
```

Additional Multi-threaded Repository operations

MarkGcCandidates (MGC)

The markForCollection operation, and many other repository operations, were made multi-threaded in previous releases.

In this release, the markGcCandidates (MGC) operation is now multi-threaded, with behavior similar to other multi-threaded operations.

The following methods are affected:

```
Repository >> markGcCandidatesFromFile:
Repository >> markGcCandidatesFromFile:forceOnError:
Repository >> markGcCandidatesReadOnlyFromFile:
```

pagesWithPercentFree:

The method `Repository >> pagesWithPercentFree:` is now multi-threaded. This method is performed moderately aggressively.

The new method `fastPagesWithPercentFree:` performs the multi-threaded scan aggressively.

Pause reclaim on low free space

A GcUser parameter has been added that allows reclaim to pause when the repository becomes low in free space. When space becomes available again, reclaim will resume.

By default, a limit is computed that is 0.1% of the repository size, or 5MB, whichever is larger.

`UserGlobals at: #reclaimMinFreeSpaceMb put: 0.`

Minimum repository free space, expressed in megabytes, which must be available in order for reclaims to proceed. Reclaims will be temporarily suspended if the repository free space drops below this threshold. The default value of 0 specifies a varying reclaimMinFreeSpaceMb that is computed as the current size of the repository divided by 1000, with a minimum value of 5 mega bytes. The default calculation is the same as that used by the stone for the STN_FREE_SPACE_THRESHOLD configuration parameter.

Minimum: 0
Maximum: 65536
Default: 0

Changes in GcUser parameters

The following parameters are no longer used, and are removed from GcUser's settings:
#reclaimDeadShadowPageThreshold
#dataPageBufferSize
#deferReclaimFreeSpaceThreshold
#reclaimSleepTime

The following parameters have been renamed to begin with a lowercase 's':
#SweepWsUnionMaxThreads
#SweepWsUnionPageBufferSize
#SweepWsUnionPercentCpuActiveLimit

The following new parameters have been added:
#reclaimMinFreeSpaceMb; described under "Pause reclaim on low free space" on page 29.
#saveWriteSetUnionToFile; described under "Access to write set union" on page 35.

postReclaimAll removed

The method `Repository>>postReclaimAll` is no longer needed and has been removed.

9. Enhancements to Repository maintenance scans

Performance improvements when entire Repository in SPC

Repository operations that place heavy demand on pages normally manage pages to reduce page "churn" on the shared page cache. This handling is unnecessary for systems that are configured with a shared page cache large enough that the entire repository can fit into the cache.

A new configuration parameter, GEM_REPOSITORY_IN_MEMORY, has been added to improve performance of operations that scan the entire repository, when the entire repository is in memory. This can be set at runtime using #GemRepositoryInMemory. This impacts the following methods:

`findReferencePathToObject:` (and related methods)
`findAllReferencePathsToObjects:` (and related methods)
`pagesWithPercentFree:` (and related methods)

If this configuration setting is FALSE (the default), the old algorithm is used and there is no change in performance from previous releases.

for details on this parameters and the affected methods, see "GEM_REPOSITORY_IN_MEMORY" on page 72.

findReferencePath* now errors on invalid objects

If the argument to `Repository>>findReferencePathToObject:` and related methods was a special or an uncommitted object, it previously returned nil. Now, error 2023 or 2027 will be triggered.

Finding multiple reference paths to an object

The methods findReferencePathToObject:* return after finding a single reference to the object, which may require the method to be run multiple times to locate all references.

Methods have been added to find all reference paths in a single execution.

```
Repository>>findAllReferencePathsToObject:
Repository>>findAllReferencePathsToObjects:
Repository>>findAllReferencePathsToObjects:limitObjArray:
   printToLog:
Repository>>findAllReferencePathsToObjects:printToLog:
```

These methods return all reference paths to the given object or objects, rather than stopping after the first reference path is found. Otherwise, the behavior is similar to findReferencePath* methods, with the return value providing an array of reference paths rather than one reference path.

See comments in the method Repository>>findAllReferencePathToObjs:limitObjArray:printToLog: for details.

listReferences: methods now error on if argument contains duplicates

The Repository methods listReferences:, fastListReferences:, listReferences:withLimit: withMaxThreads:, and listReferencesInMemory:, accept as argument an array of objects to search for. Now, if there are duplicate elements in that array, an ArgumentError is raised.

Listing objects that reference an instance of a specific class

The ability to search the repository for all instances that references an instance of a particular class has been added. For example, it is now possible to find all objects that reference a SmallFloat, which would allow the references to these objects to be updated to refer to a SmallDouble.

The following methods have been added:

`Repository >> listReferencesToInstancesOfClasses:` *anArrayOfClasses*
`toDirectory:` *aString*
`Repository >> fastListReferencesToInstancesOfClasses:` *anArrayOfClasses*
`toDirectory:` *aString*

First scan the repository for the instances of classes in anArray, then rescan to produce the sets of references to these instances.

The results of the scan are written to bitmap files in the specified directory. For each class in the array of classes, a bitmap file is created with the filename <className>-<classOop>-references.bm. These files may be loaded into hidden sets using methods in System class to analyze the results.

The value returned from this method is an array of triplets. For each class in the array of classes, the result array contains:

<aClass>, <numberOfInstancesFound> <numberOfReferencesFound>

In addition to regular classes, the special classes Boolean, Character, SmallInteger, SmallDouble, and UndefinedObject may be used. For these classes, the second value in the triplet for these classes is always zero.

Listing instances no longer limited to 2000 classes per repository pass

Methods that scan the entire repository to count instances previously would scan the repository once for each set of up to 2000 classes. The following methods should all now accept arguments longer than 2000 elements:

```
countInstances:
countInstances:withMaxThreads:maxCpuUsage:
countInstances:withMaxThreads:maxCpuUsage:
fastCountInstances:
fastListInstances:limit:
fastListInstancesInPageOrder:toFile:
fastListReferences:
fastListObjectsInObjectSecurityPolicyToHiddenSet:
anObjectSecurityPolicyId
fastListObjectsInObjectSecurityPolicies:toDirectory:
fastListObjectsInObjectSecurityPolicies:withLimit:
listInstances:toDirectory:
listInstances:limit:toDirectory:withMaxThreads:maxCpuUsage:
listInstancesInMemory:
listInstances:
listInstances:limit:
listInstances:limit:toDirectory:withMaxThreads:maxCpuUsage:
   memoryOnly:
listInstancesToHiddenSet:withMaxThreads:maxCpuUsage:
listInstancesInPageOrder:toFile:withMaxThreads:maxCpuUsage:
listInstancesInPageOrder: anArrayOfClasses toFile:
listReferences:
listReferences:withLimit:
listReferences:withLimit:withMaxThreads:maxCpuUsage:
listReferencesInMemory:
listObjectsInObjectSecurityPolicyToHiddenSet:
listObjectsInObjectSecurityPolicies:toDirectory:
listObjectsInObjectSecurityPolicies:withLimit:
```

objectAudit results output to CSV file

The following method has been added. This will record the results of any failed audit to a disk file in comma separated text.

`Repository >> objectAuditToCsvFile: aFileName`

Similar to the basic objectAudit except that if a valid file name is provided for the csvFile (comma separated values) argument then a line is written to the file for each error encountered. If no errors are found, the file is deleted.

The lines contain the following information:

1. ErrorKind - the name associated with each of the errors defined above.
2. ObjectId - the objectId for the object with an error.
3. ClassName - the name of the object's class.
4. Offset - the offset (or other integer value, e.g. size)
5. Reference - the reference that does not exist.

Methods added to combine repository profiling and listInstances

The following methods have been added to allow a single repository-wide scan to both perform a listInstances and execute profileRepository.

`GsObjectInventory >> profileRepositoryAndListInstancesInPageOrder: toFile:`
This method combines the functions of the following methods in a single scan of the repository:
  GsObjectInventory>>profileRepository
  Repository>>listInstancesInPageOrder:toFile:

Returns an Array containing 2 elements:
  [1] - An new instance of the receiver containing the profiling result.
  [2] - An Integer indicating the number of object identifiers written to the file.

`GsObjectInventory >> profileRepositoryAndSkipHiddenClassesAndListInstancesInPageOrder:toFile:`
Combines the functions of the following methods in a single scan of the repository:
  GsObjectInventory>>profileRepositoryAndSkipHiddenClasses
  Repository>>listInstancesInPageOrder:toFile:

Returns an Array containing 2 elements:
  [1] - An new instance of the receiver containing the profiling result.
  [2] - An Integer indicating the number of object identifiers written to the file.

findObjectsConnectedTo: renamed and public

Previously, the method findObjsConnectedTo: was private (in the Private method category). It is replaced by a public method with a new name.

`Repository >> findObjectsConnectedTo:` *anArray*
  Return an array of all objects reachable from the objects in the given array. Objects implicitly referenced from the header like the class are not included in the sweep.

The method `Repository >> findObjsConnectedTo:` is deprecated.

10. Changes to shared and session collections

Enhancements to persistent shared counters

Increased number of persistent shared counters

The number of persistent shared counters has been increased from 128 to 1536.

Query for number of persistent shared counters

A method has been added to query for the number of persistent shared counters

```
System class >> numberOfPersistentSharedCounters
```

SharedCounter now has public interface

SharedCounters provide updates for non-persistent, session-specific counters, via protocol in System class. These can be recorded in statmonitor data using the -m argument, and viewed in VSD under "AppStats". Previously, access was only available with private methods. Now, SharedCounters now have a public API.

The shared counter public method names match those of the existing private methods, with underscore removed, for compatibility. These method names differ from the methods to access and set Persistent Shared Counters.

The following are the added public methods:

```
System class >> sharedCounter:
System class >> sharedCounter:decrementBy:
System class >> sharedCounter:decrementBy:withFloor:
System class >> sharedCounter:incrementBy:
System class >> sharedCounter:setValue:
System class >> sharedCounterFetchValuesFrom:to:
System class >> numSharedCounters
```

Increased range support for shared counters

Shared counters now permit Integers in the range -2**63 to 2**63

SessionState now has public access

Session state is an internal array that holds non-persistent data that is not transactional; it is not affected by commit or abort.

Previously, direct access to session state was officially reserved for internal use. The class SessionTemps provided access to a dictionary stored in session state, allowing dictionary-based access to application session temporary data. SessionTemps is the recommended way to store and access session temporary application data.

For some usages, and for convenience for legacy use of the private _sessionState methods, a public interface to session state has been added. As the number of GemStone reserved array slots changes, this avoids conflict with customer use.

The following methods have been added to allow customer to access session state directly. These methods can be used with indexes 1 and higher; an argument of 1 provides access to the first available customer slot. This allows GemStone to reserve a variable number of slots before the first customer available slot.

```
System class >> sessionStateAt:
System class >> sessionStateAt:put:
System class >> sessionStateSize
```

The methods _sessionStateAt: and _sessionStateAt:put: are deprecated but have no change in behavior. You must use an index of 20 or above with _sessionStateAt: and _sessionStateAt:put:, to avoid conflict with GemStone use of session state.

Access to write set union

The Admin GC Gem can now output the write set union bitmap to a binary bitmap file before sweeping the union. To activate, use the new GcUser parameter #saveWriteSetUnionToFile. This is false by default.

When true, it writes to a file named following the pattern for the Admin GC Gem log, appended by "wsu_N.bm", where N is 0, or a higher number if a file with that name already exists. This file can be loaded and accessed using hidden set protocol.

`UserGlobals at: #saveWriteSetUnionToFile put: false`

If true, causes the Admin GC gem to write out the write set union bitmap to a binary file before sweeping the union. The file will be placed in the same directory as the Admin GC gem log file.

11. Enhancements to Hot Standby

Failover timestamps

When performing a planned failover to a hot standby, failover timestamps have been added so that it can be determined when all transactions have been transmitted and restored into the standby.

The following methods have been added

`Repository >> lastFailoverTime`

When in restore from tranlogs, this returns the DateAndTime of the most recent failOver timestamp in a transaction logs, or nil if none have been processed since restore started. If not in restore from tranlogs, this returns the DateAndTime of the most recent suspendCommitsForFailover, or nil if none have been executed.

`Repository >> suspendCommitsForFailover`
`System class >> suspendCommitsForFailover`

This method is intended for use when a hot standby master system is shutting down in order for a planned failover to the slave system.

This method suspends commits and performs a checkpoint. When the checkpoint is complete, it returns the DateAndTime of the failover timestamp.

Commits are disabled until resumeCommits is executed; while commits are suspended, no sessions can commit and mark/sweep and reclaim activity will not execute. This state persists across stone shutdown, but is not applied by tranlog replay. Restoring a backup, or startstone -R, also causes commits to be resumed.

When a transaction record of a checkpoint with a failover timestamp is transmitted to a slave system, this will cause the slave system to stop continuous restore; this ensures the slave has all records from the master, and no further work may be done on the master that could be lost.

However, if the standby was started up later than the transaction record containing a failover timestamp, it is not considered to be a failover scenario and the standby does not stop continuous restore.

`Repository >> resumeCommits`
`System class >> resumeCommits`

Resumes commits which were suspended by a call to `suspendCommitsForFailover`. Configured GcGems are restarted.

Repository >> restoreStatusInfo extra return fields

The method Repository >> restoreStatusInfo now returns additional fields with failover timestamp information:

13: a SmallInteger, 0 or the posix time in seconds of the failover timestampfor which continuous restore was stopped.

14: a SmallInteger, 0 or the posix time in seconds of the time returned from the last execution of suspendCommitsForFailover.

stopLogSender/Receiver no longer allow -A

stoplogsender and stoplogreciver no longer allow the -A argument; only 'localhost' is valid. These utilities can now only be run on the same machine as the logsender or receiver process to be stopped. The stop operations are done using kill -TERM. These changes provide greater security for the given operations.

Connection between logsender and logreceiver now can use SSL

The connection between a logsender and a logreceiver in a hot standby can now be configured to use SSL. This is done using additional options provided on the start/stop logsender/receiver utilities. Credentials must be used for both the logsender and logreceiver in order for them to connect. The SSL standard TLS v1.2 is used.

startlogsender, stoplogsender, startlogreceiver, and stoplogreceiver have the following additional parameters to specify SSL:

`-C` *fileName*	certificate in PEM format that will be sent to the peer upon request.
`-J` *fileName*	certificate authority (CA) file in PEM format to use for peer certificate verification.
`-K` *fileName*	private key in PEM format for the certificate (-C option) .
`-Q` *string*	private key passphrase. Required if the -K option is used and the private key is encrypted.
`-S`	enable SSL mode. Must be specified to use any other SSL options and must also be specified when starting the peer process.
`-V`	Disable verification of the peer's certificate.

SSL credentials (certificates, etc) are also required to stop a logsender or logreceiver operating in SSL mode.

For example, to setup a logsender/logreceiver with certificate verification fully enabled:

```
startlogsender -A localhost -P 57785 -T $GEMSTONE/data
   -l $GEMSTONE/data/logsender.log -S
   -C $GEMSTONE/examples/openssl/certs/server.pem
   -K $GEMSTONE/examples/openssl/certs/server.pem
   -J $GEMSTONE/examples/openssl/certs/serverCA.pem
```

```
startlogreceiver -A localhost -P 57785 -T /tmp/tranlogs_received
   -l $GEMSTONE/data/logreceiver.log -S
   -C $GEMSTONE/examples/openssl/certs/server.pem
   -K $GEMSTONE/examples/openssl/certs/server.pem
   -J $GEMSTONE/examples/openssl/certs/serverCA.pem -p 57786
```

Self signed certificates

Note that by default, self-signed certificates will be rejected, because the certificate cannot be verified by a known certificate authority. To use self-signed certificates, the signer must be added to the CA list in the CA file (-J flag), or certificate verification must be disabled with the -V flag. Using -V effectively tells OpenSSL to ignore certificate errors. In this mode, communications between the logsender and receiver are encrypted, but the identities of the logsender and/or logreceiver have not been verified.

12. New indexing API and new indexing features

This release includes a major enhancement and update to the code supporting GemStone indexes. While there are many internal changes and additions to the API, the existing API remains the same; existing indexes and indexed queries will continue to work as previously.

An entirely new API to create and manage indexes and to create and execute queries has been added, and many new indexing features have been added that are available using this new API. Some of these new features are available using the previous syntax as well.

The internal representations are the same for index types that are available in both new and old interfaces. Excluding the new features that are exclusive to the new API, you may create indexes using either the old or new APIs and perform queries on these indexes using either the old or new APIs.

See page 90 for index/query bugs fixed in this release.

New index creation API

Version 3.2 adds a new API to create indexes by using specifications that can be stored and reused. To create an index using the new API, you create an instance of GsIndexSpec with the statement you wish to index and any options for that index, and use that to create the index on a collection.

The GsIndexSpec can be recreated from an indexed collection using the method `UnorderedCollection >> indexSpec`. This can be used to recreate indexes on this or another collection.

The protocol to creating index specifications uses methods similar to the old index creation methods.

```
GsIndexSpec >> equalityIndex:lastElementClass:
GsIndexSpec >> equalityIndex:lastElementClass:options:
GsIndexSpec >> identityIndex:
GsIndexSpec >> identityIndex:options:
GsIndexSpec >> unicodeIndex:
GsIndexSpec >> unicodeIndex:collator:
GsIndexSpec >> unicodeIndex:collator:options:
```

For example:

```
GsIndexSpec new
   equalityIndex: 'each.name'
   lastElementClass: String
```

GsIndexSpec controls other configuration options for indexes, which are stored as instances of GsIndexOptions. Specific GsIndexOptions are returned by class methods and options are combined using + and removed using -.

```
GsIndexOptions class >> reducedConflict
GsIndexOptions class >> requirePathTerms
```

For example,

```
aGsIndexSpec options: (GsIndexOptions reducedConflict +
                                GsIndexOptions requirePathTerms).
```

To create the index, send `createIndexesOn:` to the GsIndexSpec.

For example, to define and create an RcEqualityIndex on customer name:

```
GsIndexSpec new
   equalityIndex: 'each.name' lastElementClass: String;
   options: GsIndexOptions reducedConflict;
   createIndexesOn: myCustomers
```

New index query API

In addition to a new API for creating indexes, v3.2 adds a new API to create and execute queries, with greatly increased flexibility in composing and manipulating the query and the formula that supports that query.

To create an indexed query, first create an instance of GsQuery, which can be done from a string, from an existing selectBlock, or from a GsQueryFormula that was extracted from another query. Query creation methods include:

```
GsQuery class >> fromString:
GsQuery class >> fromString:on:
GsQuery class >> fromString:on:options:
GsQuery class >> fromString:options:
GsQuery class >> fromSelectBlock:
GsQuery class >> fromSelectBlock:on:
GsQuery class >> fromFormula:
GsQuery class >> fromFormula:on:
GsQuery class >> fromFormula:on:options:
GsQuery class >> fromFormula:options:
```

The query may contain variables, which must be bound to appropriate objects before the query can be executed. The query must also be bound to a collection before execution.

```
GsQuery fromString: 'each.age <= 18'

GsQuery fromSelectBlock: {:each | each.firstName = 'Dale'}

GsQuery fromString: 'each.age < max' on: employeeCollection

'18 <= each.age < 50' asQuery

'min <= each.age < max' asQueryOn: employeeCollection
```

Binding variables, such `min` and `max` in the example above, is done using `bind:to:`, and binding to a collection is done using `on:` or `asQueryOn:`.

Once the query is fully bound, UnorderedCollection protocol may be sent to it, and it will perform the query and return the result. Sending #queryResult returns the results; you may also use messages such as #asArray, #includes:, etc. See the image for available methods.

For example:

```
aQuery := GsQuery fromString: 'min <= each.age < max'.
aQuery
   bind: 'min' to: 18;
   bind: 'max' to: 50;
```

```
        on: aCollection;
        queryResult
```

You may also stream over the results of a query using `readStream` and similar protocol.

By default, queries are always optimized before execution. Optimization creates a new GsQueryFormula that is logically equivalent to the original, but in the most efficient form for execution. If the original formula is simple, there may be no change to the formula by optimization. Optimization is not required, and can be disabled; however, some queries may perform much more slowly without optimization. SelectBlock queries, as in previous releases, are always optimized for execution.

Query syntax supports or and not

Only GsIndexSpec/GsQuery

The new API supports using or (|)and `not` in the query syntax. The query optimize step performs the appropriate transformation to the query.

Set-valued indexes

Only GsIndexSpec/GsQuery

Indexes over collections (Set valued indexes), which were available in 32-bit GemStone/S but not previously in GemStone/S 64 Bit, have been restored in this release. The syntax is the same as in 32-bit; an asterisk (*) is used to represent a collection within the path. Only collections that themselves can be indexed (subclasses of UnorderedCollections) may be indexed. Set-valued index creation and indexed queries are only available with the new GsIndexSpec/GsQuery API.

Note that certain kinds of multiple-predicate set-valued queries are disallowed, due to the semantics of set-valued queries using &. Such queries returned incorrect results in the 32-bit product, and are disallowed in the GemStone/S 64 Bit implementation. See bug 43764 for more details.

Enumerated indexed paths—indexes on multiple instance variables

Only GsIndexSpec/GsQuery

You may now create a single index that include elements in multiple instance variables. This index or query is specified using | to separate the instance variable names.

For example, to create an index that includes both instance variables firstName and nickName for a customer, use `'each.firstName|nickName'`.

As with set-valued indexes, certain kinds of multiple-predicate enumerated queries are disallowed.

Selector indexes—preview feature

Only GsIndexSpec/GsQuery

It is now possible to create an index on a selector that is not an instance variable, but a method send; however, indexes created using this feature do not NOT have automatic index updates; you must manage any changes in the targets of the index manually.

Selector based queries use the # to indicate a message send.

For example, if fullName is a method that concatenates instance variables firstName and lastName, and these values do not change, then `'each.#fullName'` would define a query on the concatenated names. However, if you update either `firstName` or `lastName` for an instance, and do not update the indexed structure directly, an indexed query invoking `#fullName` will be incorrect.

This feature is preview status in this release, and is not fully supported. In particular, infrastructure to assist in tracking and updating indexes is not yet available.

Indexes on Unicode strings

Both interfaces

Creating Unicode indexes

Unicode strings were introduced in v3.1 to allow language-specific collation. Comparison of Unicode strings requires a collator (an instance of IcuCollator associated with an instance of IcuLocale), which must not change. In previous releases, indexes on Unicode strings could not be permanently associated with a collator, and were not supported.

To create an index on a final element that is a Unicode string, or where the final elements may sometimes include a Unicode string, specify the index using the new GsIndexSpec protocol:

```
GsIndexSpec >> unicodeIndex: path
GsIndexSpec >> unicodeIndex: path collator: anIcuCollator
GsIndexSpec >> unicodeIndex: path collator: anIcuCollator
    options: aGsIndexOptions
```

If you do not explicitly specify an instance of IcuCollator, the current default collator is used. The specified, or the instance of the current default collator, will be permanently associated with the index and used for all indexed query comparisons.

Creating Unicode indexes can also be done using the legacy API, e.g.

```
UnorderedCollection >> createEqualityOn: path
    withLastElementClass: lastElementClass
```

Due to the way Unicode strings fit into the CharacterCollection hierarchy, the semantics of specifying the *lastElementClass* are different for CharacterCollection classes (it is unchanged for all other uses).

When a Unicode string class (Unicode7, Unicode 16, or Unicode32), is specified as a *lastElementClass*, a Unicode index is created, using the current default collator. This collator will be used for all indexed queries that use this index. In addition to allowing instances of Unicode7, Unicode 16, and Unicode32, regardless of class specified, instances of any subclasses of CharacterCollection are allowed.

When a *lastElementClass* of String or CharacterCollection is specified, this specifically disallows instances of Unicode7, Unicode16, and Unicode32, although otherwise the hierarchical meaning of the *lastElementClass* applies.

Queries on Unicode-indexed collections

Queries are performed using the collator that was specified during index creation. In the case of indexes created using the legacy API, or without specifying a particular collator, this is the default collator at the time the index is created.

Note that the collator affects the results of comparisons such as < and >. If the current default collator on the system is not the same as the collator associated with the index, it is possible for the results of indexed and non-indexed queries to produce different results for the same query on the same collection.

Mixing traditional and Unicode strings

Note that you should no longer consider mixing traditional and Unicode Strings in a collection. This is now subject to ArgumentError exceptions if = is used to compare two strings that are not of the same type; you cannot compare a traditional and Unicode string, and vice versa. Unicode comparison mode changes these rules. See page 44 for details.

Indexes on non-homogenous collections

Only GsIndexSpec/GsQuery

Previously, each element on the path for indexed path terms was always required to implement the instance variable in that path. This is still true by default, and when using the old API.

You may now configure the index to allow instance variable to be missing for elements by including `GsIndexOptions class >> optionalPathTerms` when defining the GsIndexSpec.

DateTime, DateAndTime, and Unicode strings now cached in index structures

Both interfaces

Btree nodes include full or partial caches of the values of a number of basic data objects, which allows comparisons to be made without fetching the object itself. In v.3.2, instances of DateTime and DateAndTime, and Unicode strings, also have values cached within btree nodes. This improves indexed query execution on instances of these classes.

Instances of the following classes are cached in the btree:

Symbol	SmallInteger	Boolean
String	LargeInteger	Character
DoubleByteSymbol	ScaledDecimal	Time
DoubleByteString	Fraction	Date
QuadByteString	SmallDouble	DateTime
QuadByteSymbol	Float	DateAndTime
Unicode7	SmallFloat	
Unicode16	DecimalFloat	
Unicode32		

Deprecated Indexing methods

Since constraints are not supported, index creation methods that require a lastElementClass and do not provide the explicit argument are deprecated.

```
UnorderedCollection >> createRcEqualityIndexOn:
UnorderedCollection >> createEqualityIndexOn:
UnorderedCollection >> getLastElementConstraintOnPath:
```

Faster removal of all indexes

When dropping all indexes in the repository, you may now do this more efficiently using the new method `IndexManager >> removeAllTracking`. This method not only removes indexes, but also removes all modification tracking (clears all DependencyLists and reinitializes SharedDependecyList). This is faster and more efficient than removeAllIndexes. However, if you are using modification tracking for another purpose, such as with GemConnect, this method should not be used, since that tracking would also be removed.

IndexManager autoCommit responsive to commit record backlog

When IndexManager is configured to autoCommit, it will now commit on a FinishTransaction signal. FinishTransaction is sent to sessions in transaction in cases where a session not in transaction would get a sigAbort.

Change in auditIndexes results on success

When UnorderedCollection>>auditIndexes finds no index problems, it responds with either 'No indexes are present' or 'Indexes are OK'.

With set valued indexes, the collections within the indexed paths (those with the * path terms), will also have logical indexes, although they themselves are not indexed. For these cases, there are two further possible additional results: 'No indexes are present, but the receiver participates in one or more indexes' and 'Indexes are OK and the receiver participates in one or more indexes.'.

13. Changes in String classes and Character

GemStone/S 64 Bit v3.2 further integrates the existing Character and String functionality with the enhanced Unicode handling provided by the ICU libraries. In this release, the collation for traditional strings – instances of String, DoubleByteString, and QuadByteString – is unchanged, but a number other behaviors now use the ICU libraries.

In addition to the changes listed here, Unicode strings can now be used in indexes. See “New indexing API and new indexing features” on page 38 for changes in indexing. Also, topaz fileout and input can handle a larger range of Characters in strings using UTF-8 encoding, and can create literal strings as unicode strings as well as traditional strings. See “Topaz handling of characters with codepoints over 128” on page 75 for enhancements to topaz.

Equality Comparison and mixing Traditional and Unicode Strings

Comparing traditional and Unicode Strings using methods such as `>` and `<` produced inconsistently ordered results in previous releases, since the types of strings use fundamentally different comparison rules. Now, it has been disallowed to compare a traditional string or a symbol with a Unicode string without a collator; attempting to sort a collection containing both traditional strings or symbols and Unicode strings will result in an error.

In addition, you may not compare traditional strings and Unicode strings for equality using `=`. This means that collections and other data structures containing a mix of traditional and Unicode strings may now encounter errors when searching and performing other operations that use `=`, such as includes:.

Comparing Unicode and Traditional Strings

Comparing a traditional string to a unicode string now always uses a collator. For traditional strings that do not use the default collator, this can be explicitly specified using `compareTo:collator:`. Passing in nil as the `collator:` argument causes compare to use IcuCollator default.

All kinds of string, Utf8, and Character now understand `compareTo:collator:` and `compareTo:collator:useMinSize:`.

Unicode Comparison Mode—preview features

To allow traditional and Unicode strings to be compared, Unicode Comparison Mode has been introduced. This mode can be configured by setting the Globals variable #StringConfiguration to Unicode16; after committing, the change takes effect for subsequent logins, by adding reimplemented comparison operators to the transient method dictionary to override String comparison methods.

This change can only be made by SystemUser.

In Unicode Comparison Mode, traditional strings and symbols behave the same way as Unicode strings in comparisons, using the default IcuCollator to perform comparisons where a collator is not explicitly specified.

Since Unicode Comparison Mode affects equality of traditional strings and symbols, as well as ordering, there is a risk of breaking lookup and incorrect results in existing data structures; affected data structures should be rebuilt. The risks depend on the specific

IcuCollator, and the specific strings in your application. Working with GemTalk Engineering is recommended, and careful testing, if you wish to experiment with Unicode Comparison Mode.

Table 1.1 Interaction of string kinds and comparison modes

String Configuration	Application does not use any Unicode Strings	Application uses Unicode strings
Legacy String Comparison Mode (Globals at: #StringConfiguration) = String	Traditional strings and symbols compare using character-based legacy rules. Collation of strings and symbols containing characters over 255 may be incorrect (without Character Data Tables installed, which is deprecated).	Cannot order or compare traditional strings and symbols with Unicode strings.
Unicode Comparison Mode (Globals at: #StringConfiguration) = Unicode16	Traditional strings and symbols compare using Unicode string-based rules.	Can mix traditional and Unicode strings; both kinds of string, and symbols, use unicode string-based comparison rules.
Changing String Configuration	Changes the way traditional strings and symbols define ordering, and the way traditional strings define equality. Affected data structures must be located and rebuilt after this change. Applications must develop validation to ensure that lookup and other code is not impacted.	Changes the way traditional strings and symbols define ordering, and the way traditional strings define equality. Affected data structures must be located and rebuilt after this change. Applications must develop validation to ensure that lookup and other code is not impacted.

To restore legacy String Comparison mode, as SystemUser, set #StringConfiguration to String.

When #StringConfiguration is set to Unicode16, it affects the defaults for topaz fileformat and set sourcestringclass settings; see page 75 for details.

Other Changes in String classes

CharacterCollection now Abstract

CharacterCollection has been effectively, but not officially, an abstract class in previous releases. Now, it is abstract; instances of CharacterCollection itself cannot be created, only instances of its subclasses.

Unicode strings now convert to traditional strings, using asString

The method `asString` has been added to Unicode string classes, to allow instances of Unicode strings to be converted to the equivalent traditional string.

asUppercase, asLowercase, containsLowercase use ICU libraries for all strings

For instances of traditional strings - String, DoubleByteString, and QuadByteString -- `asUppercase`, `asLowercase`, and `containsLowercase` previously used the default internal character tables or installed character data tables for each Character. Now, these functions are performed using the ICU libraries, which will produce correct results for any Characters with codePoints over 255, as well as producing results that are correct according to specific language rules.

The legacy behavior is available using added methods `asUppercaseOld`, `asLowercaseOld`, and `containsLowercaseOld`.

New methods in both Unicode and Traditional strings

`asFoldcase`
: Both traditional and unicode strings now understand asFoldcase, using the ICU libraries. This method converts to a case-normalized form suitable for comparisons. This is usually similar to the lowercase, but follows language specific rules to convert characters as needed.

`asTitlecase`
: Both traditional and unicode strings now understand `asTitlecase`, using the ICU libraries. Conversion to titlecase is language-specific, but generally results in the first letter uppercase followed by the balance in lowercase.

`asUppercaseForLocale:`
: Convert to uppercase using the specified IcuLocale.

`asLowercaseForLocale:`
: Convert to lowercase using the specified IcuLocale.

`asTitlecaseForLocale:`
: Convert to title case using the specified IcuLocale.

Symbol hash in new vs. upgraded repositories

Symbol and QuadByteSymbol has an improved, identity based hash in v3.2, for new 3.2 repositories only. DoubleByteStrings already used this hash in previous releases.

To preserve lookup in existing repositories, repositories that are upgraded from previous versions retain the old hash for Symbol and QuadByteSymbol, inherited from the superclass.

Character Range Changes

Characters in range 16rD800-16rDFFF disallowed

The Unicode range of codePoints from 16rD800-16rDFFF is reserved for encoding leading/trailing surrogate pairs for UTF-16 encoding. These can never be legal Unicode characters, do it is now an error to attempt to create a Character in this range.

Character range upper limit now 16r10FFFF

The upper limit of the range of Characters previously was 16r7FFFFFFF. This allowed Characters that were not valid Unicode and/or could not be replicated to GBS. The upper limit on the Character codePoint is now the Unicode limit, which is 16r10FFFF (1114111).

The new method `Character class >> maximumCodePoint` returns the limit.

Character Data Tables deprecated

Character Data Tables are fully deprecated in this release. However, changes in collation defined by loading legacy or Unicode Character Data Tables in previous releases continue to be supported. This avoids the need to rebuild indexes when upgrading to version 3.2.

For customers that require collation of Strings that contain Characters with codepoints over 255, it is recommended that you use Unicode strings, rather than traditional Strings and DoubleByteStrings with loaded Character Data Tables. Unicode strings provide flexible, language-specific provide collation based on the context of the Strings, rather than Character by Character comparisons.

Note that in addition to collation, Character Data tables permitted changes to the categories for specific Characters (such as lowercase letter or digit), and case mappings (such as lower to uppercase). Customizations to category and case mappings, other than per the Unicode standard, are not supported.

The Character compare methods: >, <, and so on, do not have changes in behavior in this release, provided Unicode Comparison Mode is not enabled.

Unicode Character methods

Character methods to change case, and to test for case and related Character properties were previously handled by character tables; either the default internal tables, or installed character data tables. Now, many of these functions are performed using the ICU libraries. A number of methods have been changed, and new methods have been added that provide additional functionality using the ICU libraries. Some of these are documented here; other methods are also available. Refer to the image for additional protocol.

These changes are to individual Characters. In most cases involving sequences of Characters, better results can be achieved using the String or Unicode string protocol, which can take into account the entire string context and the relevant language.

Changes in existing methods

The following existing methods have been changed to use the ICU libraries. The previous implementation was unreliable for Character codepoints over 255, unless larger Character Data Tables were installed. The earlier behavior is still available, using a method with the suffix `Old`.

```
asUppercase          asLowercase
isUppercase          isLowercase
isTitlecase
digitValue           digitValueInRadix:
isDigit              isNumeric
isLetter             isAlphaNumeric
isEquivalent:        equalsNoCase:
```

The following existing methods have been also changed to use the ICU libraries, due to unreliable behavior for Character with codepoints over 255, but the previous behavior is not provided.

```
asTitlecase
isSeparator
```

Added methods

The following new methods have been added, using the ICU libraries.

`asFoldcase`
: Foldcase is a case designed to use where case should be disregarded, such as some kinds of comparisons.

`numericValue`
: Returns an Integer, Fraction, or Float for any Character with a general category of numeric, such as ½ and digits in non-western languages; or nil for non-numeric Characters.

`mirror`
: The mirror of a character applies to mirrored pairs such as ${ $}, and returns the opposite of the pair.`isUppercase`
Existing method has been changed to use the ICU libraries.

`isGraphic`
: Returns true if the receiver is a graphic character; a printable character, excluding separators.

`isHexDigit`
: Returns true for digits and the letters $A-$F.

`isPrintable`
: Returns true for all non-control characters, including spaces.

`isPunctuation`
: Returns true for Characters with the Unicode general category #P.

`hasMirror`
: Returns true if the receiver is part of a mirrored pair, e.g. ${ $} and $< $>.

Unicode general category

The method `unicodeCategory` has been added, which returns the Unicode category of the Character. This is the equivalent of the legacy `_categoryAsSymbol`. Note that while the category symbols are the same, the category numbers for symbols for legacy character data do not correspond to the ICU Unicode category numbers for those symbols.

Other Character Changes

Character >> -

The method #- has been added as a Character instance method; this is part of the new petitParser code.

Changes in IcuCollator

IcuCollator default for case-insensitive compare

To allow case-insensitive compare to work correctly, a second IcuCollator is now available using `IcuCollator class >> defaultCaseInsensitive`. This is initialized as needed. and updated when `IcuCollator class >> default:` is invoked.

IcuCollator added methods

The following methods have been added to IcuCollator:

`IcuCollator class >> forLocaleNamed:`
This convenience methods has been added to get the IcuCollator for a particular IcuLocale in a single call.

`IcuCollator >> =`
Compare IcuCollators for equivalence.

14. Changes in Numbers

Literal floats with + in exponents.

GemStone traditionally uses the e+N syntax to indicate a non-negative exponent of value N, e.g. '3.0000000000000000E+01'. This is not consistent with VW or ANSI, which would parse such a string considering the + to be the plus operator. This creates a source of confusion, since literals such as 3e+1, which would evaluate in GemStone to 30.0, would instead have a value of 4.0.

In v3.2, the parsing of code containing literal floats has changed. To avoid the risk of silently changing the value of embedded literals, literals that contain the ’e’ and ’+’ without whitespace separating them will now raise an error. You will need to manually remove the + from the exponent, for values that are intending a positive exponent, or add whitespace to follow the VW and ANSI behavior of exponent followed by addition operator.

Exponents that use a negative exponent, or have no sign on the exponent, are unaffected.

`Float >> asString` output has been changed to conform to this; the + is no longer emitted.

`Float fromString:` continues to accept exponents with a preceding +, since this is not an ambiguous usage.

Changes for ANSI compatibility

Changes in display for printStringRadix:*

`Integer >> printStringRadix:`
Previously, printing using printStringRadix: included the radix of the Integer, e.g. 16rFF. Now, the radix is omitted, so the result would be FF. To include the radix, use printStringRadix:showRadix:.

`Integer >> printStringRadix:showRadix:`
Previously, this displayed any negative signs after the radix, before the value; for example, 16r-FF. Now, any negative signs are first, so this value would now print -16rFF.

Other changes to conform to ANSI

`Integer >> lcm:`
Previously, this would return negative values if exactly one of the operands was negative; now, this method always returns the positive value.

`Integer >> bitAt:put:`
This now accepts put: arguments other than 0 and 1; the low order bit of the argument is used. Note that as of 3.0, the offset for bitAt:put: and bitAt: are 1-based rather than 0-based.

`Integer, LargeInteger, SmallInteger >> highBit`
This is now 1-based, not 0-base, for consistency with ANSI and other Integer bit methods.

`Number >> to:by:`
This now reports an error on zero `by:` argument, which is behavior specified by ANSI.

`Number >> asFloatE`, `asFloatD`, and `asFloatQ`
Return an object of SmallDouble or Float, depending on the range. GemStone does not have different implementations for FloatE, FloatD, and FloatQ.

Number class >> fromStream: added

A method has been added, Number class >> fromStream:, that is able to read any kind of GemStone numeric literal from a stream.

Change in Float emax/emin

The constants returned by the Float class methods emax and emin were incorrect, and have changed per bug #43107 (see page 92). Previously these returned 308/-308, now 1024/-1021.

Methods to get components of a Float

The following public methods have been added, to extract the components of a floating point:

```
Float >> signBit
Float >> exponent
Float >> mantissa
```

Method to get maximum 32-bit signed SmallInteger

The method `SmallInteger >> maximum32bitInteger` has been added.

15. Changes in Collections

Printing for recursive dictionaries and other collections

Printing dictionaries or other collections that contain references to the collection itself or to associations in that dictionary, previously caused a variety of printing problems. Generally, this problem existed as recursion as far as the printing character limit, but some types of dictionaries may skip printing the association containing the recursion.

Collections now print using new functionality, using nonRecursivePrintString and printNonRecursiveOn:. These are invoked by printString and printOn:, so printing behavior will be different in 3.2 for existing classes.

If you implement printing behavior for custom collection classes, you may override the method printNonRecursiveRepresentationOn:recursionSet:, to print your collection, sending printOn:recursionSet: to elements.

isAssociation added and used in Dictionary methods

The method isAssociation has been added to Object and Association, and Dictionary methods that previously tested for `isKindOf: Association` now invoke isAssociation instead. This provides greater flexibility in creating custom dictionary classes and elements.

WriteStream print: added

The WriteStream classes (WriteStream, WriteStreamLegacy and WriteStreamPortable) now understand the print: method. The print: method is a convenience method for that sends printOn: to the argument. E.g, the following two statements are equivalent:

```
myObj printOn: aStream
aStream print: myObj
```

Added general Collection methods any and removeAllPresent:

`Collection >> any`

Return an arbitrary element of the receiver. Error if the receiver is empty.

`Collection >> removeAllPresent:`

Removes from the receiver one occurrence of each element of aCollection that is also an element of the receiver. Differs from removeAll: in that, if some elements of aCollection are not present in the receiver, no error is generated.

This method was previously available for UnorderedCollections, but not SequentiableCollections.

Changes in Dictionary >> sort* methods

The Collection hierarchy includes methods sortAscending, sortAscending, sortDescending, sortDescending:, and sortWith:. Traditionally these GemStone-specific methods, when sent to a dictionary, have sorted the associations in the dictionary by key. However, the ANSI standard considers the elements of a Dictionary to be the values, and does not consider associations.

In a previous 2.x release, sortAscending and sortDescending were inadvertently changed to follow a more ANSI-like behavior, and return the sorted values. sortAscending:

sortDescending:, and sortWith: did not get this update, and retained the older behavior of returning the Associations, sorted by key.

For consistency with asSortedCollection and for closer adherence to an ANSI definition of dictionary behavior, although these are not ANSI methods, the behavior of all such sorting methods now considers the elements as being the values in the dictionary. The collection returned by these methods will include the values only, not associations, and be sorted by value, not key.

Changes made for ANSI compatibility

Methods with changes in behavior to conform to ANSI

`SequenceableCollection >> copyReplaceFrom:to:with:`

Previously, the arguments could be out of range to implement append behavior, and it did not allow the ANSI insert functionality.

Now, the to: argument is permitted to be exactly one greater than the from: argument, and the from: argument may be one larger than the size of the collection. If the to: argument is one greater than the from: argument, the replacement objects are inserted between the from: and to: indexes, and no elements of the receiver are replaced.

It is now an error if the to: argument is larger than the receiver's size, if the from: argument is less than 1, or if the to: argument is more than one less than the from: argument.

`CharacterCollection >> subStrings:`

Previously, this required the argument to be a single Character. ANSI defines that this method take a string argument. Now, both characters and strings are accepted,

This change is implemented via the following added methods, which can be accessed directly:

`subStringsDelimitedBy:` *aCharacter*
`subStringsDelimitedByAny:` *separators*

OrderedCollection new: error on negative argument

When attempting to create an instance of OrderedCollection using new: with a negative argument, previously this did not error, and returned an empty OrderedCollection. Now, per ANSI, this will throw an error.

Methods added for ANSI compatibility

`Collection >> do:separatedBy:`

Evaluates the one-argument block using each element of the receiver in order. aBlock is evaluated between successive elements. Returns the receiver.

`SequenceableCollection >> from:to:keysAndValuesDo:`

For each index in the specified range, the 2-argument block is evaluated with the index as the first argument and the element at that index as the second argument.

`SequenceableCollection >> copyReplacing:withObject:`

Returns a copy of the receiver in which all occurrences of objects equal to oldObject have been replaced by newObject. Identical to existing non-ANSI method copyReplacing:with:.

Existing methods implemented for additional classes

The following methods were previously implemented for a subset of collection classes, and have now been extended to other collection classes for ANSI compatibility:

`SequenceableCollection >> at:ifAbsent:`
This was previously implemented only for dictionaries.

`SequenceableCollection >> keysAndValuesDo:`
This was previously implemented only for dictionaries. When used with a sequenceable collection, the key is the index and the value is the element at that index.

CharacterCollection added Magnitude protocol

The following methods have been added to CharacterCollection, with behavior comparable to methods on Magnitude:

```
between:and:
max:
min:
```

SortedCollection added SequenceableCollection protocol

Previously, SortedCollection disallowed or did not implement a number of existing SequenceableCollection methods required by ANSI. The following methods now work correctly for SortedCollection; other methods that invoke these methods may also now work.

```
at:
with:with:with:with:
,
copyReplaceAll:with:
copyReplaceFrom:to:with:
copyWith:
```

Random byte generation methods added

The following methods uses the OpenSSL function RAND_bytes() to generate cryptographically strong pseudo-random bytes.

`ByteArray >> addRandomBytes:` *count* `startingAt:` *offset*
Add *count* random bytes to the receiver starting at *offset*. The receiver will grow if necessary, and any existing data in the range of offset to (*offset* + *count* - 1) is overwritten.

`ByteArray class >> withRandomBytes:` *count*
Return a new instance of the receiver of size *count* containing randomly generated data.

ReadStream nextElements:into: added

The ReadStream classes (ReadStream, ReadStreamLegacy and ReadStreamPortable) now understand the nextElements:into: method. This method is used by PassiveObject.

16. Changes in fileout

Code refactored

The methods supporting fileout in Behavior and in ClassOrganizer have been refactored. Fileout functionality in ClassOrganaizer now invokes the Behavior methods; this particular improves SymbolDictionary fileout format.

The output is functionally equivalent to the previous format, other than changes that fix bugs in the previous fileouts. See page 89 for fileout bugs fixed in this release.

Changes in formatting of image fileout

There are changes in location and content of comments within the fileout, and the ordering of the line with the class category fileout.

While the line "set compile_env: 0" is still included in fileouts, it is no longer printed as part of every method.

Changes in topaz fileout

The fileout form produced by the topaz **fileout** command has several changes:

- Can be configured to handle extended characters by writing out in UTF-8;
- New commands are now included in the output to set the string class and file format;
- Now includes the class comment and class category.

See “Topaz fileout” on page 76 for details on fileout in topaz with v3.2.

Image fileout of code with values over 127

With version 3.2, topaz filein will accept both UTF-8 encoded input as well as plain text, with plain text limited to Characters with codePoints under 255. To avoid incorrect interpretation of any text containing Characters with codePoints over 127, the **`fileformat`** topaz command must be set appropriately for the file contents; **`UTF8`** if the contents are encoded in UTF-8, or **`8BIT`** for plain text.

The `fileformat` line is added automatically to the output file when filing out using the **`topaz fileout`** command.

When filing out using image methods, you must manually add the appropriate text to the file, and manually convert the fileout body to UTF-8. By doing this, you can file out and file in code containing the full range of codePoints. For example:

```
file := GsFile openWrite: 'MyClass.out'.
file nextPutAll: 'fileformat UTF8'; lf.
file nextPutAsUtf8: (MyClass fileOutClass).
```

Note that GBS does not add the fileformat command to fileouts, nor does GBS use the **`fileformat`** command to control interpretation of filein, although this command does not error in GBS. GBS fileouts rely on client Smalltalk file protocols to write the specific file format. You should manually verify the required settings if you plan to file out from GBS and file in using topaz, or vice versa, if you have any Characters with codePoints over 127. Code filed out encoded as UTF-8 will create incorrect characters if filed in as **`8BIT`**, and vice versa.

17. Other Changes

Stack frames now include receiver and implementor classes, omit envId

Previously, printing a stack frame in topaz included the name of the class that implemented the method for the frame. Now, it also includes the class of the receiver. The class of the implementor follows the receiver in parenthesis.

Also, the environment ID is no longer printed if executing in environment 0, which the environment used by Smalltalk.

For example, a stack frame that previous printed:

```
AbstractException >> _signalWith:      (envId 0) @5 line 25
```

now prints:

```
ZeroDivide (AbstractException) >> _signalWith:   @5 line 25
```

These display changes apply to stack frame displays in general; including stacks displayed in the GBS debugger, or in other tools.

Migration now allowed across class histories

Previous versions of GemStone limited instance migration to migration between instances sharing the same classHistory. While this is the normal case, it is sometimes useful to migrate to equivalent but unrelated classes. Migration is no longer limited to classes with a shared classHistory.

"self" now resolves to nil

In topaz and other code evaluation in which "self" was previously unresolved, it will now resolve to nil.

Added #DbfOrigin

A new global has been added to Globals, #DbfOrigin, providing an integer representing the version of GemStone in which this repository originated. It is not affected by upgrade.

Pragmas allowed in primitive methods

Pragmas are now allowed in methods containing a <primitive: NN> , <protected> , and <unprotected> tokens.

18. Other Added Methods

System changed and added methods

descriptionOfSession: new element

The array returned by System class >> descriptionOfSession: includes an additional result, at index 21:

21, SmallInteger, processId of the remote GCI client process, or -1 if the session has no remote GCI client

Added methods

The following methods have been added:

`System class >> commandLineArguments`
Returns an Array of Strings, the command line arguments to the gem or topaz -l process for this session. Each invocation of this method returns a new instance of the result.

`System class >> gemLogPath`
Returns a String, the path of the directory in which the session's gem log file is being written. Returns an empty String in a topaz -l process. Each invocation of this method returns a new instance of String.

`System class >> gemProcessId`
Returns a SmallInteger, the processId from Unix getpid() of this session's gem or topaz -l process .

Methods added to Object, understood by all classes

speciesForPrint

To ensure the correct class of String is returned, printing methods now invoke `speciesForPrint`.

asUnicodeString

The method `Object >> asUnicodeString` has been added. This is similar to asString, but the class of the return is a Unicode rather than a traditional string.

describe1K

This method has been added to limit the error message size when reporting errors on large objects.

GsSocket added methods

Efficient writes of multiple buffers

If writing a number of separate strings to a socket, performing sequential write operations with no intervening read involves a delay for each TCP roundtrip. To avoid creating a new object containing the entire text and not incur the delay, a method has been added to write multiple byte objects in a single call.

`GsSocket >> writev:` *numBuffers* `specs:` *specArray*

Use the writev system call to write the specified number of buffers. specArray is an array of size numBuffers*3, in which each 3-element sequence describes a buffer:

* A byte object containing bytes to be written
* The 1-based index within the buffer of the first byte to be written
* The number of bytes to be written.

numBuffers must be <= 20.

This method will block until the write completes, or until a socket error occurs. Returns receiver if write completes successfully, otherwise signals an Error. Signals an ArgumentError if an element of specArray is invalid. Signals a SocketError if the underlying writev() fails.

Accept with timeout

`GsSocket >> acceptTimeoutMs:`

This new method is similar to the existing method GsSocket >> accept, but allows a timeout to be specified.

GsFile added methods

`GsFile >> ,`

Write the contents of the argument to the receiver's file at the current position.

`GsFile >> maxSize`

For compatibility with Stream; returns SmallInteger maximumValue.

`GsFile >> nextPutAsUtf8:` aString

Writes the contents of the given String, MultiByteString or Utf8 to the receiver's file using UTF8 encoding. If this file is later read in using GsFile, use standard GsFile protocol to read the file, and send `decodeFromUtf8` to the contents.

TimeZone added methods

`TimeZone class >> availableZones`

Return a sorted Array of Strings representing the TimeZones in the GemStone distribution of the Olson timezone database.

`TimeZone class >> named:`

Return a n instance of TimeZone with the specified name, from the GemStone distribution of the Olson timezone database.

GciInterface added methods

The following methods have been added to GciInterface. For details, see the method comments in the image.

```
GciInterface >> storeTravExecute:flags:environmentId:execute:
   alteredObjects:traverseInto:
```

```
GciInterface >> storeTravPerform:flags:perform:selector:args:
   alteredObjects:traverseInto:
```

Class added method

`Class >> allSuperclasses`

Return a collection of all superclasses of the receiver, excluding the receiver. Associated methods have also been added to ClassOrganizer.

Block added methods

The following method has been added for ANSI compatibility:

`BlockClosure >> ifCurtailed:`

Evaluate the receiver and return its result. If abnormal termination of the receiver occurs, the argument terminationBlock is evaluated. The value returned from the evaluation of terminationBlock is discarded.

Methods to get count of block arguments

The method `argumentCount` as been added to the following classes; this method returns the number of arguments needed to evaluate the receiver.

```
BlockClosure >> argumentCount
ExecBlock >> argumentCount
SelectBlock >> argumentCount
```

The new `argumentCount` methods invoke the same primitive as the existing methods `ExecBlock >> numArgs` and `ExecBlock >> numberArgs`.

- `ExecBlock >> numArgs` is retained as generally-used GemStone legacy protocol.
- `ExecBlock >> numberArgs` is deprecated.

GemStone/S 64 Bit v3.2 - New Features

19. Logging of session logins and logouts

A feature has been added to allow automatic recording of each login and logout from the stone (as well as other operations that are related, such as stone startup and shutdown).

This feature is disabled by default. To enable, set the new configuration parameter **STN_LOGIN_LOG_ENABLED** to TRUE in the configuration file used by the stone, prior to stone startup. See page 73 for details on this configuration parameter.

By default, all sessions that login to the stone will have logins and logouts logged, when enabled in the stone. You can disable logging for specific users by using the new method `UserProfile >> disableLoginLogging`. After this is executed, that UserProfile will not have logins or logouts recorded in the log file.

Logins and logouts are recorded to a text file named *stoneName*`_login.log`, in the same directory as the stone log.

Each log entry is on a line, with the following fields:

```
TimestampString TimeStampSeconds EventKind UserName SessionId
ProcessId RealUserID EffectiveUserID HostName GemIPAddress
ClientIPAddress NumCommits
```

For example:

```
"10/03/2013 16:42:48.720" 1380843768 STARTUP Stone 0 15270 631 631
kata.gemtalksystems.com 204.45.122.94 0.0.0.0 0
"10/03/2013 16:43:13.017" 1380843793 LOGIN DataCurator 3 15317 631
631 kata.gemtalksystems.com 204.45.122.94 10.94.141.15 0
"10/03/2013 16:43:23.488" 1380843803 SHUTDOWN Stone 0 15270 631
631 kata.gemtalksystems.com 204.45.122.94 0.0.0.0 0
```

Added methods

The following new methods allow you to manage login/logout logging status for specific UserProfiles, which overrides the status in the stone.

Note that setting this status for a particular UserProfile has no effect if login/logout logging is not enabled in the stone. However, once the logging status is set for a UserProfile, that will remain set for that UserProfile. If the stone later does have logging enabled, that UserProfile’s status will be used to determine logging.

`UserProfile >> exemptFromLoginLogging:` *aBoolean*
Enable or disable the printing of login/logout events for the receiver to the stone's login log file. Has no effect if the stone does not have login logging enabled. Requires write access to the DataCurator segment and the #OtherPassword privilege.

`UserProfile >> disableLoginLogging`
Disables the printing of login/logout events for the receiver to the stone's login log file. Requires write access to the DataCurator segment and the #OtherPassword privilege.

`UserProfile >> enableLoginLogging`
Enables the printing of login/logout events for the receiver to the stone's login log file,

if login logging is enabled in the stone. Requires write access to the DataCurator segment and the #OtherPassword privilege.

`UserProfile >> hasLoginLogging`
Answer true if the receiver is configured to write login and logout events to the stone's login log file, false otherwise. This value does not take into account if the stone has login logging enabled.

Added Cache Statistics

The cache statistics **LoginLogFlushes**, **LoginLogThreadOperations**, and **NumInLoginLogQueuehave** been added. See "Added Cache Statistics" on page 67.

20. Access to AES encryption/decryption

Advanced Encryption Standard (AES) encryption/decryption is performed using the OpenSSL open source package and the AES specification, available at:

`http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

All encryptions/descriptions are in cipher block chaining (CBC) mode; see the AES specification document for further details.

Encryption and decryption API methods are provided for 128-bit/16-byte keys, 192-bit/24-byte keys, and 256-bit/32-byte keys. The key is a ByteArray of the appropriate size. The salt must be a 128-bit/16-byte ByteArray.

The following methods are provided on ByteArray and CharacterCollection, for performing the encryption and decryption:

`aesEncryptWith128BitKey:` *aKey* `salt:` *aSalt* `into:` *aByteObjOrNil*
Encrypts the receiver using 128 bit AES encryption and places the result into *aByteObjOrNil*. If *aByteObjOrNil* is nil, the results will be places in a new instance of the class of the receiver. Otherwise, *aByteObjOrNil* must be a non-invariant kind of byte object; the results will be placed in that, starting at offset 1, and it will be resized if necessary.

`aesEncryptWith192BitKey:` *aKey* `salt:` *aSalt* `into:` *aByteObjOrNil*
Encrypts the receiver using 192 bit AES encryption and places the result into *aByteObjOrNil*. If *aByteObjOrNil* is nil, the results will be places in a new instance of the class of the receiver. Otherwise, *aByteObjOrNil* must be a non-invariant kind of byte object; the results will be placed in that, starting at offset 1, and it will be resized if necessary.

`aesEncryptWith256BitKey:` *aKey* `salt:` *aSalt* `into:` *aByteObjOrNil*
Encrypts the receiver using 256 bit AES encryption and places the result into *aByteObjOrNil*. If *aByteObjOrNil* is nil, the results will be places in a new instance of the class of the receiver. Otherwise, *aByteObjOrNil* must be a non-invariant kind of byte object; the results will be placed in that, starting at offset 1, and it will be resized if necessary.

`aesEncryptWith128BitKey:` *aKey* `salt:` *aSalt*
Encrypts the receiver using 128 bit AES encryption and places the result into a new instance of the class of the receiver.

`aesEncryptWith192BitKey:` *aKey* `salt:` *aSalt*
Encrypts the receiver using 192 bit AES encryption and places the result into a new instance of the class of the receiver.

`aesEncryptWith256BitKey:` *aKey* `salt:` *aSalt*
Encrypts the receiver using 256 bit AES encryption and places the result into a new instance of the class of the receiver.

`aesDecryptWith128BitKey:` *aKey* `salt:` *aSalt*
Decrypts the receiver using 128 bit AES decryption and places the result into a new instance of the class of the receiver.

`aesDecryptWith192BitKey:` *aKey* `salt:` *aSalt*
Decrypts the receiver using 192 bit AES decryption and places the result into a new instance of the class of the receiver.

`aesDecryptWith256BitKey:` *aKey* `salt:` *aSalt*
Decrypts the receiver using 256 bit AES decryption and places the result into a new instance of the class of the receiver.

`esDecryptWith128BitKey:` *aKey* `salt:` *aSalt* `into:` *aByteObjOrNil*
Decrypts the receiver using 128 bit AES decryption and places the result into *aByteObjOrNil*. If *aByteObjOrNil* is nil, the results will be places in a new instance of the class of the receiver. Otherwise, *aByteObjOrNil* must be a non-invariant kind of byte object; the results will be places in that, starting at offset 1, and will be resized if necessary.

`aesDecryptWith192BitKey:` *aKey* `salt:` *aSalt* `into:` *aByteObjOrNil*
Decrypts the receiver using 192 bit AES decryption and places the result into *aByteObjOrNil*. If *aByteObjOrNil* is nil, the results will be places in a new instance of the class of the receiver. Otherwise, *aByteObjOrNil* must be a non-invariant kind of byte object; the results will be places in that, starting at offset 1, and will be resized if necessary.

`aesDecryptWith256BitKey:` *aKey* `salt:` *aSalt* `into:` *aByteObjOrNil*
Decrypts the receiver using 256 bit AES decryption and places the result into *aByteObjOrNil*. If *aByteObjOrNil* is nil, the results will be places in a new instance of the class of the receiver. Otherwise, *aByteObjOrNil* must be a non-invariant kind of byte object; the results will be places in that, starting at offset 1, and will be resized if necessary.

Random byte generation

The new method ByteArray class >> withRandomBytes: simplifies creating keys and salts. This uses the OpenSSL function RAND_bytes() to generate cryptographically strong pseudo-random bytes. See page 53.

Example

An example of using AES encryption/decryption can be found in this method:

```
CharacterCollection>>encryptionExample
```

21. External Sessions

GemStone/S 64 Bit v3.2 incorporates a number of new classes to facilitate spawning and managing external sessions.

GsNetworkResourceString

GemStone uses Network Resource Strings (NRS) for interprocess communication; in particular, logins require that the stone and gem resources be identified using NRS strings. NRS syntax is GemStone-specific. The new class GsNetworkResourceString provides a way to construct NRS strings programatically, and to request default gem and stone NRS strings for common configurations.

The following class methods construct basic NRS strings, using the GemStone default values (such as gs64ldi, gs64stone, and localhost) for any values that are not provided as arguments:

```
stoneNRS
stoneNRSForStoneName: aStoneName
stoneNRSForStoneName: aStoneName onHost: stoneHostName
gemNRS
gemNRSForNetLDI: netldiNameOrPort
gemNRSForNetLDI: netldiNameOrPort onHost: gemHostName
gemNRSForNetLDI: netldiNameOrPort onHost: gemHostName
    gemService: customGemService
```

The following instance variables can be used to set further values on the NRS string. You may apply these to a NRS created using `GsNetworkResourceString new` or to one created using the above methods.

`log`	log file name for gem
`temporaryObjectCacheSize`	cache size for gem
`dir`	log directory for gem
`authorization`	host unix id and password
`netldi`	netldi name or port number
`node`	host name
`body`	stone name or gem service name

Sending `asString` provides the formatted NRS string.

GsExternalSession and related classes

External sessions allow you to execute Smalltalk code in separate Gems, which may run on different servers and log in as different users to different repositories. This allows you do to things such as partitioning work among multiple gems or managing separate repositories.

Operations to create and communicate with the external sessions use the Foreign Function Interface (FFI) to access the GCI libraries; except on AIX, which has limitations in FFI support. On AIX, external sessions use the primitives provided to support the GciInterface class, and use instances of GsLegacyExternalSession.

The new class GciLibrary provides interface methods for all the GemBuilder for C functions, as generated from gci.hf. Note that this library exposes all functions provided by the GCI interface, both public and private.

Much of the basic functionality of GsExternalSession was previously available via the GciInterface class. GsExternalSession and its associated classes provide a more simple and intuitive interface, and provide a number of helpful additional features. However, GsExternalSession does not provide traversal, and this interface is not intended for extensive direct transmittal of objects between the two Gems.

To use GsExternalSession, you create an instance, specifying the stone and gem NRS and the GemStone username and password. You may then login and execute code. Code is passed to the external session as strings, or as blocks (which are converted to strings for transmission to the remote session). This code may be executed synchronously or asynchronously. Return values should be specials (SmallInteger, Character, etc.), or strings (kinds of CharacterCollection or ByteArray. Returning other objects should be avoided; they are returned as OOPs in the remote gem, which may or may not exist in the calling gem.

The new classes GciError and GciLegacyError have been added to represent errors during remote execution. If the code being executed on the remote session encounters an exception, this is raised as a GciError in the calling session (or GciLegacyError, if running with GsLegacyExternalSession). Since remote debugging is not possible with this interface, the stack of the error is included with the error description. This does not include compilation errors, which use CompileError.

For example, to execute countInstances: for String in a separate session on the same repository

```
| ses res |
ses := GsExternalSession new.
ses
   stoneNRS:
      (GsNetworkResourceString defaultStoneNRSFromCurrent);
   gemNRS:
      (GsNetworkResourceString gemNRSForNetLDI: 'gs64ldi');
   username: 'DataCurator';
   password: 'swordfish'.
ses login.
res := ses executeString:
   '(SystemRepository countInstances: { String }) first'.
ses logout.
res
```

22. PetitParser ported to GemStone

Petitparser has been ported to GemStone.

PetitParser is an open-source parsing framework that makes it easy to define parsers with Smalltalk code and to dynamically reuse, compose, transform and extend grammars. The resulting grammars can be modified on the fly.

A number of classes have been added, with the prefix PP. In addition, PetitParser related methods have been added to appropriate classes in the image.

23. GsHostProcess for asynchronous performOnServer:

System >> performOnServer: can execute arbitrary OS code on the server, but only operates synchronously; Smalltalk blocks until complete. It provides no access to stdin and returns data from both stdout and stderr.

To provide more flexible access to run processes on the host, a new class is provided, GsHostProcess.

To use this, use the class method `fork:`, passing the command line you wish to execute. This will return immediately with an instance of GsHostProcess with sockets on stdin, stdout, and stderr. You can use socket protocol to read from or write to these sockets.

Note that pathname resolution is not provided. You must fully qualify executable paths.

For example:

```
| hostprocess |
hostprocess := GsHostProcess fork: '/bin/ls'.
hostprocess stdout read: 1024
```

GemStone/S 64 Bit v3.2 - Utility and Environment Changes

24. Changes in Exceptions and handling

Compiler warnings on shadowed block and temporary variables

When the compiler encounters issues with shadowed temporary variables, it now triggers a complier warning. Per ANSI, block argument and temporary variables are permitted to have the same name as instance variables, method temporary variables, or block arguments or temporaries declared in an outer scope; references resolve to the block declaration, not the outer scope.

Similar compiler warning were reported in previous releases for unused temporary variables.

Warnings are written for:

- unused method or block temporary variables (this is existing behavior)
- block temporary variable shadowing a method temporary, instance variable, or block argument
- block argument shadowing a method temporary, instance variable, or outer scope block argument

You may log these (and other Warnings) to the gem log or another file by modifying Warning>>defaultAction.

Changes to error messages

The following errors have updated text:

RepositoryError #2734, from multithreaded restore.

#rtErrSpecialOrNotCommitted #2027

#rtErrGetGcLockFailed #2501

OutOfRange

Login failure errors include more information

Previously, when login failed, it returned GS_ERR_LOGIN_DENIAL, error 4051, which reported invalid username or password. Now, this may report different error conditions, such as a failure to initialize SSL.

AlmostOutOfMemory signal now enabled by default, but has no action

AlmostOutOfMemory, if enabled, is signaled when the gem is close to running out of temporary object memory. This allows appropriate action to avoid the gem running out of memory and dying. Previously, this had to be enabled manually in each image; in v3.2, it is enabled by default. The default action on signal is to do nothing.

FFI errors in processing C headers

When an error is encountered while processing C header files, it now signals an Error rather than UserDefinedError, and the description include the file and line number.

Removed Errors

The following GCI errors have been removed:

Legacy Number	GCI error name	Image name
2224	GCI_ERR_BREAK_CANCELED_MSG	#gciErrBreakCanceledMsg
2225	LGC_ERR_SYNC_1	#lgcErrSync1
2226	LGC_ERR_SYNC_2	#lgcErrSync2
2227	LGC_ERR_BYTE_PACKET_TOO_LONG	#lgcErrBytePacketTooLong
2228	LGC_ERR_ARG_SIZE_INCONSISTENT	#lgcErrArgSizeInconsistent
2229	LGC_ERR_OOP_PACKET_TOO_LONG	#lgcErrOopPacketTooLong
2232	LGC_ERR_EXPECTED_END	#lgcErrExpectedEnd
2233	LGC_ERR_PACKET_KIND_UNKNOWN	
2234	LGC_ERR_EXPECTED_CMD	#lgcErrExpectedCmd

The following GCI errors have been added:

Legacy Number	GCI error name	Image name
2516	LGC_ERR_TRAV_BUFF_READ_ERR	#lgcErrTravRead
3002	ABORT_ERR_CommitsSuspended	#abortErrCommitsSuspended
4151	ERR_FATAL_NATIVE_CODE	

25. Cache Statistics Changes

Added Cache Statistics

ClientAborts (Pgsvr)
Number of times the client gem of this page server aborted a transaction.

ClientCommits (Pgsvr)
Number of times the client gem of this page server committed a transaction.

LoginLogFlushes (Stone)
Number of times the stone's login log thread has flushed writes to the login log file.

LoginLogThreadOperations (Stone)
Number of operations performed by the stone's login log thread

NumInLoginLogQueue (Stone)
Number of sessions queued to have entries written to the stone's login log file.

PageMgrPolls (Stone)
Number of times pagemanager thread has polled

PageMgrSleepMs (Stone)
Number of milliseconds the pagemanager thread has slept

PageMgrSleepState (Stone)
Number of times pagemanager thread has slept

PageMgrThreadWakeups (Stone)
Number of times pagemanager thread has woken up

Changed Cache statistics

The following cache statistics have been renamed and the units have changed from K bytes to bytes.

CodeCacheSizeKBytes replaced by **CodeCacheSizeBytes**
MeSpaceAllocatedKBytes replaced by **MeSpaceAllocatedBytes**
MeSpaceUsedKBytes replaced by **MeSpaceUsedBytes**
NewGenSizeKBytes replaced by **NewGenSizeBytes**
OldGenSizeKBytes replaced by **OldGenSizeBytes**
OmKBytesFlushed replaced by **OmBytesFlushed**
PermGenSizeKBytes replaced by **PermGenSizeBytes**
PomGenSizeKBytes replaced by **PomGenSizeBytes**

Removed Cache statistics

The following statistic has been removed:

OldGenPreGcSizeKBytes

26. Utilities Changes

Several utilities add -v option to get version

The utilities:

```
copydbf              statmonitor
startlogsender       stoplogsender
startnetldi          stopnetldi
startlogreciver      stoplogreciver
startstone           topaz
```

now return the version when passed the -v argument. For example,

```
UNIX> startstone -v
startstone 3.2.0 BUILD: 64bit-32386
```

The gslist utility, which has previously existing behavior for the -v option, does not follow this pattern. Gslist returns the version using the -V option.

copydbf

There have been minor improvements in the wording and capitalization of output messages, and copydbf output is now printed to stdout rather than stderr.

gslist changes

gslist -x added options line for some processes

The complete output provided for gslist -x did not include the process startup arguments list in all cases. The options: line was added with these arguments for the stone, logsender and logreceiver processes.

gslist option for parseable output

The timestamp printed for process start in gslist is not entirely parseable; for example, it does not include the year. To override the simple default timestamp, you may set a new environment variable, GS_GSLIST_TIME_FORMAT. This can be set to a UNIX-style date format string, and gslist will use that to display the timestamp. If this variable is not set, the timestamp is unchanged. (#42956)

gslist -v now includes restoreStatus

gslist output has been enhanced so the output with -v includes a report of stones in restore status. For example:

```
restoreLogs 3.2.0     gsadmin   Nov 23 17:51 Stone       gs64stone
```

pstack prints smalltalk stack

The pstack utility now will also print the Smalltalk stack to the process's gem log.

startnetldi no longer accepts port range argument

the -p option for startnetldi has been removed, since the netldi no longer creates separate sockets for gem login.

startstone restricts stone name to legal characters

Characters in the stone name are now restricted to letters, digits and underscore, dash, and period, which is the POSIX standard.

Stone logging timestamp interval reduced

To avoid clutter in the stone log, not all messages are printed with individual timestamps. The delay between timestamps has been reduced from 15 seconds to 1 second.

Hot standby utility argument changes

startlogsender, startlogreceiver, stoplogsender, and stoplogreceiver now all have additional arguments for SSL; and stoplogsender and stoplogreceiver no longer accept the -A argument. These changes are described under "Enhancements to Hot Standby" on page 36.

Change in environment variable that control automatic log deletion

Several GemStone system processes, such as GcGems, automatically delete their log files on shutdown. Previously, this automatic log deletion could be prevented by commenting out the reference to GEMSTONE_CHILD_LOG in the scripts that started these gems.

The environment variable to do this has changed. Now, to ensure that a GemStone system gem does not delete it's log file on exit, set the new environment variable GEMSTONE_KEEP_LOG, by uncommenting the appropriate lines in the appropriate script.

27. Configuration Parameter Changes

Ability to pre-grow extents to non-maximum sizes

Previously, if you pre-grew your extents on stone startup, you were limited to pre-growing to the maximum size limit of the extents. Any further space later required beyond that size would cause a fatal out of space condition.

Now, DBF_PRE_GROW can be specified as a collection of Integers as well as TRUE/FALSE. When a set of Integers is specified, the corresponding extent is grown to that size on startup, but may continue to grow later up to the maximum specified in DBF_EXTENT_SIZES (if any).

DBF_PRE_GROW

Pre-grow DBF extent sizes. When this is set to TRUE and there are extents for which a size is set in DBF_EXTENT_SIZES, each extent with a size in DBF_EXTENT_SIZES will be pregrown to that size.

The value of DBF_PRE_GROW may also be a list of integer sizes. Extents with an integer size specified in DBF_PRE_GROW will be pregrown to this size if needed. It is an error if a (non-blank) DBF_PRE_GROW size is larger than a (non-blank) value for an extent in DBF_EXTENT_SIZES. Extents without an entry in DBF_PRE_GROW will not be pregrown.

The sizes are in units of Megabytes (1 Megabyte = 1048576 bytes), minimum value 1, maximum value 33554432.

Elements of DBF_PRE_GROW may be blank to specify pregrow sizes for some but not all extents, such as:
DBF_PRE_GROW = 1000, , 1000;

Default: FALSE

Specify parameter values in KB, MB, or GB

Some configuration parameters can now be specified using KB, MB, or GB, as well as is done in previous releases by a plain number, which is specified in the default units. For example, now if you wish to set a 100GB shared cache, any of the following can be used:

```
SHR_PAGE_CACHE_SIZE_KB = 100000000;
SHR_PAGE_CACHE_SIZE_KB = 100000000KB;
SHR_PAGE_CACHE_SIZE_KB = 100000MB;
SHR_PAGE_CACHE_SIZE_KB = 100GB;
```

Numbers specified using MB, KB, or GB are absolute settings, not in reference to the default units. While the KB suffix on some parameters may be somewhat misleading when using other units, for convenience for existing systems, the names have not changed.

The parameters that this applies to are:

DBF_PRE_GROW
DBF_EXTENT_SIZES
GEM_PRIVATE_PAGE_CACHE_KB
GEM_TEMPOBJ_CACHE_SIZE

GEM_TEMPOBJ_MESPACE_SIZE
GEM_TEMPOBJ_POMGEN_SIZE
SHR_PAGE_CACHE_SIZE_KB
STN_FREE_SPACE_THRESHOLD
STN_PRIVATE_PAGE_CACHE_KB
STN_TRAN_LOG_LIMIT
STN_TRAN_LOG_SIZES

Default for Temporary Object Memory increased to 50000

The default for GEM_TEMPOBJ_CACHE_SIZE was previously 10000; in v3.2, the default is 50000.

Maximum Process Count changes

To avoid problems with reaching the limit on the maximum number of processes able to attach to a cache, there have been changes in the configuration parameters.

The maximum number of Reclaim GcGem sessions for a given extent is now set by a separate configuration parameter, STN_MAX_GC_RECLAIM_SESSIONS. This defaults to the number of extents specified in DBF_EXTENT_NAMES. You specify the number of Reclaim GcGems to start up automatically as usual, using STN_NUM_GC_RECLAIM_SESSIONS. The setting for STN_MAX_GC_RECLAIM_SESSIONS limits the number you can specify the start at startup, and the number of Reclaim GcGems you can start up programmatically.

Slightly more cache space is required for higher values of SHR_PAGE_CACHE_NUM_PROCS. We recommend leaving SHR_PAGE_CACHE_NUM_PROCS at the default, which allows the system to calculate the number of permitted processes. The computation is now:

```
STN_MAX_SESSIONS (for user sessions)
    + 5 (for stone, pcmon, pagemanager, symbolgem and admingem)
    + STN_MAX_GC_RECLAIM_SESSIONS (for reclaim gem sessions)
    + SHR_NUM_FREE_FRAME_SERVERS (for free frame page servers)
    + STN_NUM_LOCAL_AIO_SERVERS (for AIO page servers)
    + 1 (for statmonitor)
    + 1 (for hot standby gem)
    + 1 (to allow stopstone to login)
```

Change in specification for GEM_NATIVE_CODE_ENABLED

The parameter GEM_NATIVE_CODE_ENABLED can now be set to 0 (equivalent to FALSE), 1 (equivalent to TRUE), or 2.

Setting this parameter to 0 or FALSE disables native code. Setting to 1 or TRUE enables native code. Setting it to 2 enables native code with inlining of some SmallInteger math primitives.

The default is 2.

This can also be passed to the gem using the -N command line argument.

Minimum Shared Page Cache size increased to 16000

The minimum setting for SHR_PAGE_CACHE_SIZE minimum is now 16000 (which specifies a cache somewhat larger than 16 MB).

Added Configuration Parameters

GEM_REPOSITORY_IN_MEMORY

Determines the performance behavior of the gem for certain operations that scan the entire repository. If set to TRUE, the gem assumes most or all of the data pages in the repository have been previously loaded into the shared page cache. If set to FALSE, the gem assumes most or all of the data pages in the repository are not in the shared page cache and must be read from disk.

This setting affects performance only. All operations affected by this setting will succeed and produce the same results.

Repository instance methods affected by this setting are:

`findReferencePathToObject:` (and related methods)
`findAllReferencePathsToObjects:` (and related methods)
`pagesWithPercentFree:` (and related methods)

Runtime equivalent: #GemRepositoryInMemory
Default: FALSE

SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_POLICY

Specifies whether large memory pages will be used when creating the shared page cache. Enabling large memory pages can result in significant performance gains when large shared page caches are used. The improvement is due to a reduction in translation lookaside buffer (TLB) cache misses. The TLB is an internal structure used by the operating system to manage memory address translation.

Large memory page support is an operating system and hardware dependent feature. Currently, GemStone supports large memory pages on AIX and Linux only. This configuration option is silently ignored on all other platforms.

Three policies are available on supported operating systems:

0 - Disabled: No large memory page support.
1 - Advisory: Large memory pages are requested when the shared page cache is created. If the operating system denies the request, a warning is printed in the SPC monitor log file and the cache is started without large memory pages. NOT RECOMMENDED ON AIX!
2 - Mandatory: Large memory pages are requested when the shared page cache is created. If the operating system denies the request, an error is printed in the SPC monitor log file and the shared page cache fails to start.

Both Linux and AIX require operating system kernel changes in order to enable large memory pages. “Improved support for SPC large memory pages” on page 20.

Default: 0
Minimum: 0
Maximum: 2

STN_ALLOW_NFS_EXTENTS

If TRUE, stone will startup using extents and tranlogs which are on NFS-mounted filesystems. This is less reliable and less performant than locally mounted filesystems, or filesystems on storage arrays which appear as local mounts. This variable cannot be changed at runtime.

Default: FALSE

STN_LOGIN_LOG_ENABLED

Enable the logging of all session login and logout events to s separate log file owned by the stone. The file will be named *stoneName*_login.log and will be placed in the same directory as the stone log.

When this feature is enabled, logins and logouts are recorded for all sessions by default. Logging may by disabled for a UserProfile by sending the #disableLoginLogging message to a UserProfile instance and committing the transaction.

The login log file is a text file that contains one line per event. Fields within a line are separated by spaces; the Timestamp String is quoted. The fields logged in each line are:

- Timestamp String - time in human-readable form
- Timestamp Seconds - seconds from the epoch (January 1, 1970, 00:00 UTC)
- Event Kind - one of STARTUP, SHUTDOWN, LOGIN, LOGIN_FAIL, LOGOUT, or COMMIT_RESTORE.
- UserName - the UserProfile's userId, or "Stone" for the stone process.
- SessionId
- ProcessId
- Real UNIX user ID - numeric value; always 0 on Windows.
- Effective UNIX user ID - numeric value; always 0 on Windows.
- Host Name - node name where the gem process is running.
- Gem IP Address - IP Address of the gem.
- Client IP Address - IP Address of the gem's client.
- NumCommits - number of commits performed by the session.

STARTUP and SHUTDOWN records are written to indicate when the stone was started and stopped and do not indicate a session login or logout.

Login failures are written for non-exempt sessions that fail a login attempt, usually due to specifying a bad password.

Default: FALSE

STN_MAX_GC_RECLAIM_SESSIONS

The maximum number of page reclaim garbage collector sessions which are expected to be used on the system. When the default is specified, the actual value used is the number of extents defined in DBF_EXTENT_NAMES configuration.

Default: 0
Minimum: 0
Maximum: 256

STN_MAX_LOGIN_LOCK_SPIN_COUNT

Maximum number of times a session will attempt to write lock its user security data object at login time before raising a fatal error and failing the login. The session will sleep for 100 milliseconds between retries.

Enabling certain UserProfile security features (password aging, etc) causes each session to update its user security data object at login time and commit. A write lock must be acquired on this object to guarantee the commit succeeds.

Each UserProfile has a unique user security data object. Lock retries may be required when 2 sessions attempt to login with the same user ID at nearly the same instant. Simultaneous logins that user different user IDs never require lock retries.

Repositories that do not enable UserProfile security features are not affected by this parameter because the write-lock and commit described above are not required at login time.

Runtime equivalent: #StnMaxLoginLockSpinCount
Default: 100
Minimum: 1
Maximum: 36000

28. Topaz changes

Topaz echo of input from stdin

When topaz takes input other than from a terminal, the echo to terminal has been improved. Now, input command text, not just topaz prompts and command results, are displayed. This is consistent with what is sent to file by **output push**.

Topaz handling of characters with codepoints over 128

Topaz handling of text containing codepoints over 128 has been substantially improved in this release. These changes affect fileout (using the topaz **`fileout`** command), filein, and literal strings created via code execution at the topaz command line.

Previously, Topaz did not directly support fileout, filein, or literal strings that contained Characters with codepoints over 128.

The following new or changed commands support extended characters.

fileformat

`fileformat` is a new command in topaz. It requires an argument, and accepts either **`8bit`** or **`UTF8`**.

`utf8` sets the file format to UTF-8. Code that is filed out using **`fileout`** is encoded in UTF-8, and files read using **`input`** are interpreted as being UTF-8 and are decoded accordingly.

`8bit` sets the file format to 8-bit. Code that is filed out using **`fileout`** is not encoded. For compatibility with previous releases, this is the default. Fileout of code containing Characters with codePoints over 255 will error.

Input from stdin that is connected to a tty (i.e. interactive stdin) is always interpreted as UTF-8.

When a **`fileformat`** command is encountered in input files, the setting is used for the rest of that input and any nested files, but restored to the previous setting when the input file is at the end.

set sourcestringclass

The existing command **`set sourcestringclass`** has been updated. It requires an argument, and accepts **`String`** or **`Unicode16`**.

`String` means that literal strings created during topaz commands such as **`run`** and **`input`** (filein) are instantiated as instances of String, DoubleByteString, or QuadByteString. For compatibility with previous releases, this is the default.

`Unicode16` means that literal strings created during topaz commands such as **`run`** and **`input`** are instantiated as instances of Unicode7, Unicode16, or Unicode32.

If the repository has set the #StringConfiguration reference in Globals to Unicode16, on login, if an explicit command to set either setting has not been done, topaz will set file format to UTF8 and sourcestringclass to Unicode16. Note that this mode is not supported for existing repositories, as it changes the behavior of = and other comparison operators, and may break existing collections.

Topaz fileout

Commands reflecting the setting for **`fileformat`** and **`set sourcestringclass`** are now included in code fileouts made using the topaz **`fileout`** command.

Note that this does not affect fileout using Smalltalk image methods, in Behavior and ClassOrganizer, and GBS fileout; fileout other than using the topaz command is unaffected by this change. See “Changes in fileout” on page 54 for image fileout changes and handling of UTF-8.

The **`fileout toFile:`** command has an additional optional keyword, **`format:`**, which can be used to specify **`8Bit`** or **`UTF8`** and override the current topaz setting.

GBS has been enhanced to accept the fileformat command, but the command has no effect on fileins using GBS tools.

The fileout produced by topaz **`fileout`** now also include the class comment and class category.

Topaz new command options

In addition to the command changes described that related to extended Characters, a number of Topaz commands have new options.

`limit lev1bytes` *anInteger*
When the topaz **level** is set to 1 or greater, this limit controls how many bytes to display of instVar values and frame temporaries. If **lev1bytes** is set to zero, then the value of "limit bytes" is used for instVar values and frame temporaries.

`display stacktemps`
enables the display of stack frames to include un-named evaluation temps which have been allocated by bytecodes within the method.

`omit stacktemps`
Disables effect of **display stacktemps**.

`lookup`
now accept paste of the method text as displayed from `where` and other commands.

`step thru`
Advances execution to the next step point in the current frame, or its caller, or the next step point in a block for which current frame's method is the home method.

topaz .hlp file may be included with executable

Previously, the topaz .hlp file was required to be in $GEMSTONE/sys. Now, on Solaris and Linux only, it will also search the directory containing the executable.

29. VSD updated and productized

There have been many changes and improvements in VSD.

Packing and distribution

VSD now is now packaged as an independent product, with its own documentation, as well as being included in the GemStone/S 64 Bit product distributions.

The statistics information provided for VSD is now distributed in a separate file, allowing the same VSD executables and vsd TCL code to display information for different versions of GemStone.

VSD is now also included in the Windows Client distribution; both 32-bit and 64-bit versions of VSD are distributed.

New Documentation

VSD Documentation is no longer included in the *System Administration Guide*; expanded and improved documentation is provided in the new *VSD User's Guide*. This documentation includes the statistics reference that was formerly in the *System Administration Guide*.

Updated libraries

The TCL/TK libraries that VSD is build on have been updated to a new version. This fixes a number of bugs or unexpected behaviors in VSD.

VSD for Windows has been recompiled for Windows 7; this version is not supported on earlier versions of Windows.

Windows startup changes

VSD on windows is no longer started from a .bat file. You now execute the executable vsd.exe, directly. This also avoids the application's need to open a command prompt window.

The platform-independent TCL file, with the name vsd on UNIX platforms, is named vsd.tcl on Windows.

Improvements and new features:

- All panes with scrollbars now can use mouse scrolling
- On Windows, alt-F4 now closes window
- Command line now accepts the -h argument
- Some of the preinstalled templates have been cleaned up. Note that these changes will only be visible if you delete your existing .vsdtemplates file, which would also delete any custom templates.
- The dialog that allows you to choose a statistics file to load, now saves the dialog dimensions in the .vsdrc file, on UNIX platforms, so the same dimensions will be used the next time a file dialog is opened.

30. GCI changes

New thread-safe GCI interface

In addition to the existing GCI interface as documented in the GemBuilder for C manual, there is a new, parallel, thread-safe set of GCI calls.

This does not affect existing GCI applications.

Thread safe libraries are distributed on all platforms with the names:

```
libgcits-3.2.0-64.so or .dll
libgcits-3.2.0-32.so or .dll
```

Documentation for the thread-safe GCI interface is found in the include file:

```
$GEMSTONE/include/gcits.hf
```

Checksums for GCI to detect changes

To detect any changes in the structs of the GCI interface between versions, the repository now calculates the Md5 checksum for the standard GCI and the thread-safe GCI. T his is stored in the image:

```
Globals at: #GciStructsMd5
Globals at: #GciTsStructsMd5
```

The distribution $GEMSTONE/doc directory includes files containing the report on the stucts:

```
gciStructs.txt
gcitsStructs.txt
```

GCI applications can use the checksum to verify that the version of the GCI is compatible.

Added GCI functions

The following GCI functions have been added. For more information, see the *GemBuilder for C* manual.

`(int64)` **`GciFetchUtf8Bytes_`**`( OopType` *aString*`, int64` *startIndex*`, ByteType` *theBytes*`[], int64` *numBytes*`, OopType *`*utf8String*`, int` *flags* `)`

This function encodes an instance of a kind of String, MultiByteString, or Utf8 into UTF-8. The encoded bytes are placed in the buffer *theBytes*, and the OOP of the encoded objects is placed at *utf8String*. The OOP of *utf8String* is also put in the ExportSet, and must be manually removed using **GciReleaseOops** after fetching all bytes.

If all characters in aString are < 128, or if the class of aString is Utf8, then the the behavior is the same as **GciFetchBytes_**. No encoding is done; the bytes of *aString* are place in *theBytes*; *utf8String* is the same as *aString*, and is not added to the ExportSet.

`(OopType)` **`GciNewUtf8String`**`(const char* utf8data, BoolType convertToUnicode)`

Returns a new instance of Utf8, Unicode7, Unicode16, or Unicode32, with the value that the UTF-8 encoded *unicodeCString* points to. *unicodeCString* is a pointer to a null-terminated UTF-8 encoded character string. If *convertToUnicode* is 0, this function returns an instance of Utf8; if it is 1, it returns an instance of Unicode7, Unicode16, or Unicode32, the minimal character size required to represent *unicodeCString*.

`(OopType)` **`GciNewUtf8String_`**`(const char*` *unicodeCString*`, size_t` *nBytes*`, BoolType` *convertToUnicode*`)`

Returns a new instance of Utf8, Unicode7, Unicode16, or Unicode32, with the value that the UTF-8 encoded *unicodeCString* points to. *unicodeCString* is a pointer to a UTF-8 encoded character string of length *nBytes*. If *convertToUnicode* is 0, this function returns an instance of Utf8; if it is 1, it returns an instance of Unicode7, Unicode16, or Unicode32, the minimal character size required to represent *unicodeCString*.

`(GciNbProgressEType)` **`GciNbEndPoll`**`( int64 *`*result*`, int` *timeoutMs* `);`

Similar to GciNbEnd_, but waits for a result for up to *timeoutMs* before returning. The result in **result* is 8 bytes and is correct on big endian machines.

`(int)` **`GciPollSocketForRead`**`( int` *socketFd*`, int` *timeoutMs*`,);`

Wait for *timeoutMs* milliseconds for the specified socket to be read-ready or to have an error. This function returns 0 if timed out, 1 if socket is ready for read, and an int < 0 if an error occurred. The result in this cases is the negated errno value.

Other GCI function changes

Breaks in User Actions

GciHardBreak and GciSoftBreak have no effect if called from within a user action.

Execute accepts Utf8

GciExecute, GciExecute_, GciExecuteFromContext, GciExecuteFromContext_ accept a Utf8 as well as a String.

GciIntAppName_ has additional argument

This function now includes a argument to specify native code

Linux Compile and Link Information

Complier version

g++ 4.4.3

Debugger version

GNU gdb (GDB) 7.1

Compiling a user action or GCI application

```
g++ -fmessage-length=0 -fcheck-new -O3 -ggdb -m64 -pipe -pthread
   -fPIC -fno-strict-aliasing -fno-exceptions -I$GEMSTONE/include
   -x c++ -c userCode.c -o userCode.o
```

The following warn flags are recommended for compilation:

```
   -Wformat -Wtrigraphs -Wcomment -Wsystem-headers -Wtrigraphs
   -Wno-aggregate-return -Wswitch -Wshadow -Wunused-value
   -Wunused-variable -Wunused-label -Wno-unused-function
   -Wchar-subscripts -Wmissing-braces -Wmultichar -Wparentheses
   -Wsign-compare -Wsign-promo -Wwrite-strings -Wreturn-type
   -Wuninitialized
```

Linking a user action library

```
g++ -shared -Wl,-Bdynamic,-hlibuserAct.so userCode.o
   $GEMSTONE/lib/gciualib.o -o libuserAct.so -m64 -lpthread -lcrypt
   -ldl -lc -lm -lrt -lpam -lpam_misc -Wl,-z,muldefs
   -Wl,--warn-unresolved-symbols
```

Linking a GCI application

```
g++ userCode.o $GEMSTONE/lib/gcirtlobj.o
   -Wl,--warn-unresolved-symbols -m64 -lpthread -lcrypt -ldl -lc
   -lm -lrt -lpam -lpam_misc -Wl,-z,muldefs -o userAppl
```

Solaris on SPARC Compile and Link Information

Complier version

CC: Sun C++ 5.8 Patch 121017-05 2006/08/30

Debugger version

Sun Dbx Debugger 7.5 Patch 121023-07 2010/09/22

Compiling a user action or GCI application

```
CC -xO4 -xcode=pic32 -xarch=v9 -mt -xchip=ultra2
   -I$GEMSTONE/include -features=no%except
   -features=no%anachronisms -c userCode.c -o userCode.o
```

Linking a user action library

```
CC -xarch=v9 -G -Bsymbolic -h libuserAct.so -i userCode.o
   $GEMSTONE/lib/gciualib.o -o libuserAct.so -Bdynamic -lc
   -lpthread -ldl -lrt -lsocket -lnsl -lm -lpam -lCrun -znodefs
```

Linking a GCI application

```
CC -xildoff -xarch=v9 -i userCode.o $GEMSTONE/lib/gcirtlobj.o
   -z nodefs -Bdynamic -lc -lpthread -ldl -lrt -lsocket -lnsl
   -lm -lpam -lCrun -o userAppl
```

Solaris on x86 Compile and Link Information

Complier version

CC: Sun C++ 5.10 SunOS_i386 128229-09 2010/06/24

Debugger version

Sun DBX Debugger 7.7 SunOS_i386 2009/06/03

Compiling a user action or GCI application

```
CC -xO4 -m64 -xarch=generic -Kpic -mt -I$GEMSTONE/include
   -features=no%except -c userCode.c -o userCode.o
```

Linking a user action library

```
CC -m64 -xarch=generic -G -Bsymbolic -h libuserAct.so -i userCode.o
   $GEMSTONE/lib/gciualib.o -o libuserAct.so -Bdynamic -lc -lpthread
   -ldl -lrt -lsocket -lnsl -lm -lpam -lCrun -z nodefs
```

Linking a GCI application

```
CC -xildoff -m64 -xarch=generic -i userCode.o $GEMSTONE/lib/gcirt-
   lobj.o -z nodefs  -Bdynamic -lc -lpthread
   -ldl -lrt -lsocket -lnsl -lm -lpam -lCrun -o userAppl
```

AIX Compile and Link Information

Complier version

IBM XL C/C++ for AIX, V11.1

Debugger version

dbx

Compiling a user action or GCI application

```
xlC_r -O3 -qstrict -qalias=noansi -q64 -+ -qpic
   -qthreaded -qarch=pwr6 -qtune=balanced -D_LARGEFILE64_SOURCE
   -qminimaltoc -qlist=offset -qmaxmem=-1 -qsuppress=1500-
   010:1500-029:1540-1103:1540-2907:1540-0804:1540-1281:1540-1090
   -qnoeh -I$GEMSTONE/include -c userCode.c -o userCode.o
```

Note that there is no space in the `-qsuppress` arguments that are continued on the following line.

Linking a user action library

```
xlC_r -G -Wl,-bdatapsize:64K -Wl,-btextpsize:64K
   -Wl,-bstackpsize:64K -q64 userCode.o $GEMSTONE/lib/gciualib.o
   -o libuserAct.so -e GciUserActionLibraryMain -L/usr/vacpp/lib
   -lpthreads -lc_r -lC_r -lm -ldl -lbsd -lpam -Wl,-berok
```

Linking a GCI application

```
xlC_r -Wl,-bdatapsize:64K -Wl,-btextpsize:64K
   -Wl,-bstackpsize:64K -q64 userCode.o $GEMSTONE/lib/gcirtlobj.o
   -Wl,-berok -L/usr/vacpp/lib -lpthreads -lc_r -lC_r -lm -ldl
   -lbsd -lpam -Wl,-brtllib -o userAppl
```

DARWIN Compile and Link Information

Complier version

i686-apple-darwin10-g++-4.2.1 (GCC) 4.2.1 (Apple Inc. build 5664)

Debugger version

GNU gdb 6.3.50-20050815 (Apple version gdb-1469) (Wed May 5 04:36:56 UTC 2010)

Compiling a user action or GCI application

```
g++ -DOBJ_gct -fmessage-length=0 -fcheck-new -O3 -ggdb -m64 -pipe
   -fPIC -fno-strict-aliasing  -D_LARGEFILE64_SOURCE
   -I$GEMSTONE/include -x c++ -c userCode.c -o userCode.o
```

The following warn flags are recommended for compilation:

```
   -Wformat -Wtrigraphs -Wcomment -Wsystem-headers -Wtrigraphs
   -Wno-aggregate-return -Wswitch -Wshadow -Wunused-value
   -Wunused-variable -Wunused-label -Wno-unused-function
   -Wchar-subscripts -Wconversion -Wmissing-braces -Wmultichar
   -Wparentheses -Wsign-compare -Wsign-promo -Wwrite-strings
   -Wreturn-type
```

Linking a user action library

```
g++ -dynamiclib userCode.o $GEMSTONE/lib/gciualib.o
   -o libuserAct.dylib -m64 -lpthread -ldl -lc -lm -lpam -undefined
   dynamic_lookup
```

Linking a GCI application

```
g++ userCode.o $GEMSTONE/lib/gcirtlobj.o -undefined dynamic_lookup
   -m64 -lpthread -ldl -lc -lm -lpam -o userAppl
```

Windows Compile and Link Information

Complier/Debugger version

Microsoft Visual Studio 2010 Version 10.0.30319.1 RTMRel

Microsoft Visual C++ 2010   01021-532-2002102-70611

Compiling a GCI application

```
cl /W3 /Zi /MD /O2 /Oy- -DNDEBUG /TP /nologo /D_LP64 /D_AMD64_
   /D_CONSOLE /D_DLL /DWIN32_LEAN_AND_MEAN
   /D_CRT_SECURE_NO_WARNINGS /DNATIVE
   /I 'VisualStudioInstallPath\atlmfc\include'
   /I 'VisualStudioInstallPath\VC\include'
   /I 'C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Include'
   /I '%GEMSTONE%\include' -c userCode.c -FouserCode.obj
```

Linking a GCI application

```
link /LIBPATH:"VisualStudioInstallPath\VC\lib\amd64"
   /LIBPATH:"C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Lib\x64"
   /OPT:REF /INCREMENTAL:NO /MAP /nologo /MANIFEST
   /MANIFESTFILE:userAppl.exe.manifest
   /MANIFESTUAC:"level='asInvoker'" userCode.obj
   %GEMSTONE%\lib\gcirpc.lib ws2_32.lib netapi32.lib advapi32.lib
   comdlg32.lib user32.lib gdi32.lib kernel32.lib winspool.lib
   /out:userAppl.exe
```

Chapter
2

Bugs Fixed in GemStone/S 64 Bit 3.2

The following bugs have been fixed in GemStone/S 64 Bit v.3.2:

"Heartbleed" OpenSSL security bug

The OpenSSL Heartbleed bug (http://heartbleed.com/) was present in versions of the shared libraries that are used in version 3.1 and later. Version 3.2 includes an updated version of the openSSL libraries with the fix for this bug. (#44080)

Reclaim paused during multi-threaded scans

3.x releases before 3.1.0.5 were subject to a bug in the coordination between reclaim and the multi-threaded scan operations, such as backup and markForCollection. This bug could resulted in the multi-threader operation crashing. To avoid this risk, in 3.1.0.5, reclaim is suspended during multi-threaded operations. The underlying bug is fixed in this release, and reclaim does not pause. (#43261)

Hang with swap space full on AIX

When swap space is full, AIX sends a SIGDANGER signal. The Stone did not correctly handle this signal and would not recover from the hang, when the low swap space was resolved. (#44060)

NotTranloggedGlobals not handled correctly in restore from logs

Creation of non tranlogged objects is not recorded in the tranlogs, which could cause problems during restore if objects were garbage collected. (#42702, #43113). This feature was disabled in version 3.1.0.3 and later to avoid risk, and is re-enabled and fixed in v3.2.

Unable to read tranlog with record-level compression

The slave stone encountered errors while restoring tranlogs in a hotstandby system, in cases where the tranlogs contained many records for byte objects containing compressed data. The tranlogs written by the logreceiver, or by copydbf -c, use record-level compression; this bug affects reading of the record-level compressed format from the tranlogs. (#44131)

Login issues

LDAP authentication failed on first attempt, then succeeds

The method System >> validatePasswordUsingLdapServers:baseDn:filterDn: userId: password: may fail on the first attempt but succeed on all subsequent attempts, even if the password is correct every time, due to an uninitialized handle. (#43627)

Login failure after 100 attempts to lock UserSecurityData

When user security is enabled for a user profile or repository-wide, information must be written to a user profile's UserSecurityData on login. With many simultaneous logins, a particular login may be unable to lock the object for some time. Previously, the login would fail after a fixed 100 attempts; now, this limit is configured using the new configuration parameter STN_MAX_LOGIN_LOCK_SPIN_COUNT. See page 74 for details on this parameter. (#42862)

Upgrade and Migration issues

Conversion fails to upgrade AllUsers if disallowedUserPasswords is corrupted

If a repository returns a non-Boolean from AllUsers disallowUsedPasswords, the conversion process fails to upgrade AllUsers. A corrupted disallowUsedPasswords setting may not cause problems in execution, only in upgrade.

The code performing conversion is unchanged; a step has been added to the conversion process in the Installation Guide, to check for a corrupted disallowUserPassword result and correct if necessary. (#43820)

Image upgrade resets #sleepTimeBetweenReclaimMs to 0

On image upgrade, the configured value for the GcUser configuration parameter #sleepTimeBetweenReclaimMs (in the GcUser's UserGlobals) is reset to 0. (#43521)

recompileAllMethods did not update needsRecompileFor30 status if no instance methods

After the upgrade from 2.x to 3.x, you may execute needsRecompileFor30 to determine if a class needs to be recompiled. Sending recompileAllMethods then recompiles the methods in the class and updates the status. However, if the class did not have instance methods, recompileAllMethods did not update the status. (#42718)

Internal use of deprecated methods made deprecation detection problematic

GemStone introduced deprecation mechanisms, including errors and warnings when deprecated methods were invoked. Internal calls within Gemstone to deprecated methods made this feature very cumbersome to use in practice. (#42531)

See page 30 for details on changes in deprecation in v3.2.

Invariance lost during migration

Objects that are invariant became variant when migrated to a new class version. (#43311)

Upgrade does not clear default definitions from CPreprocessor

Upgrade did not clear the DefaultDefinitions classVar on CPreprocessor. (#43213)

Versioning class lost setting for class category

Any value set in a classes' category instance variable was lost when that class was versioned. (#42967)

Collection bugs

RcQueue>>changeMaxSessionId: may cause problems

The code implementing changeMaxSessionId: did not correctly initialize the new sections of the RcQueue, and could potentially result in later errors. (#42719)

RcIdentityBag did not error on removeAll:

Unlike other collection class instances, sending removeAll: to an RcIdentityBag with an argument for which some elements did not exist in the bag succeeded rather than erroring with LookupError 2015.

RcIdentityBag >> removeAllPresent: returned receiver not argument

Other implementors of removeAllPresent: returned the argument, while RcIdentityBag >> removeAllPresent: returned the receiver. For consistency, this method now also returns the argument. (#43539)

Large IdentityDictionaries may be corrupted during rebuild

Instances of IdentityDictionary must be rebuilt periodically as elements are added. There was a code path in rebuild that could cause values in key-value pairs to be set to Associations containing the key and value. This occurred in large IdentityDictionaries, and the specific conditions depend on how elements hash into the dictionary. (#43515)

Incomplete locking in Dictionary and IdentityDictionary

When a Dictionary or IdentityDictionary was locked, it was still possible to modify the values of existing key-value pairs. Locking now prevents modification of the dictionary as it does with other types of dictionaries. (#42383)

Dictionary copy creates copy with shared associations

When an object is copied, changes to that object should not have the side effect of modifying the original object, although copy does not make copies of the instance variable values themselves. For dictionaries that are implemented using associations, including Dictionary and IdentityDictionary, the result did not create a copy of the associations, so changes in values at given keys were applied to both collections. (#42760)

Bag equality false positives with multiple entries

Bag equality comparisons did not account for non-unique collection elements and could return false positives. (#43414)

Set, Bag, RcQueue >> deepCopy results empty or invalid

The inherited deepCopy implementation did not allow for the internal structure of these classes, results in empty or invalid results. (#43181)

Some Dictionary subclasses did not correctly implement objectSecurityPolicy:

The implementation of objectSecurityPolicy: was suitable for Dictionary, but not for other types of dictionary. (#42608)

Infinite loop comparing SymbolDictionaries with same name and contents

If two instance of SymbolDictionary, which have the same contents and the same name, are checked for equivalence using =, the comparison would get stuck in an infinite loop. (#42281)

RcKeyValueDictionary select: returned RcKeyValueDictionary

RcKeyValueDictionary >> speciesForSelect returned RcKeyValueDictionary, which resulted in select: and similar methods returning instances of RcKeyValueDictionary. To avoid the overhead of creating complete reduced-conflict structures, most Rc classes return the simpler non-RC equivalent. Now, RcKeyValueDictionarys speciesForSelect is KeyValueDictionary. (#43407)

SymbolKeyValueDictionaries did not coece arguments for all methods to Symbol

Some methods in SymbolKeyValueDictionary did not coerce arguments to Symbols. Updated methods include at:ifAbsent:, at:otherwise: , removeKey:ifAbsent:, and removeKey:otherwise: (#42716)

String related issues

Large byte object logical size may be corrupted by setting ObjectSecurityPolicy

When setting the security policy using objectSecurityPolicy:, for a large object (an object that requires more than a page of storage space, and thus uses internal tree structures to implement), it may have left the internal root node with an incorrect logical size. This would affect ByteArrays and Strings over about 16K in length. (#43640)

Case-insensitive search is case-sensitive with Unicode strings

Methods that perform case-insenstive search, such as includesString: and findStringNoCase:startingAt:, performed a case-sensitive, rather than case insensitive search on unicode receivers. (#43104, #43773)

Unicode string behavior failed with locale en_US_POSIX

The locale en_US_POSIX is intended for binary data, not language sensitive data, and if the operating system locale resolved to this locale, Unicode string comparisons did not work correctly. Now, if this locale is implied by the OS locale, IcuCollator>>default returns en_US instead. (#43104)

Unicode String (comma) Double/QuadByteString errorred

Using the comma operator to append a Double or QuadByteString to a Unicode string resulted in an error. (#43749)

Passivate/activate of MultiByteStrings/Symbols to file produced corrupted string on little-endian platforms

When an instance of DoubleByteString , DoubleByteSymbol, QuadByteString, or QuadByteSymbol was passivated to a file on a little-endian platform such as Linux/x86, and reactivated, the byte order was reversed, producing invalid results. (#42800)

Bugs involving Streams

FileStream >> nextPut: incorrect error return

FileStream >> nextPut: should return the argument, or raise an error. If there was an error in the underlying GsFile operation, this method would incorrectly return true or nil. (#43106)

FileStream>>atEnd could have returned nil

FileStream>>atEnd should return either true or false, but if the underlying GsFile had an error, it would return nil. Now, an error will cause the method to return true. (#43762)

PassiveObjects could not be used with Portable Streams

In version 3.x, either Legacy or Portable Stream hierarchies could be installed as the Positionable Streams in an image. One primary difference is a 1-based vs. 0-based offset. This offset differences caused failures using PassiveObject with the Portable stream hierarchy. (#42596)

Now, PassiveObjects can be used with any kind of Stream, and with instances of GsFile.

Fileout issues

The code supporting Fileout has been refactored, and there are minor changes in the output. See “Changes in String classes and Character” on page 44 for details.

Fileout requires CodeModification privilege

Fileout unnecessarily required #CodeModification privilege. (#43128)

Fileout of SymbolDictionary did not include class comments

When a SymbolDictionary was filed out, the fileout did not include class comments for classes in that SymbolDictionary. (#42653)

ClassOrganizer fileout form versioned classes

Fileouts using ClassOrganizer, rather than the fileout methods in Behavior, filed out in a format that forced new versions of the classes to be created. This was a particular issue for the 2.x to 3.x upgrade conversion. (#42317)

Indexing Bugs

Indexing has had substantial work and internal cleanup in this version, as part of adding the new API and new features (see “New indexing API and new indexing features” on page 38. It is likely that previously undocumented bugs have been fixed as part of these changes, and there are likely to be minor changes in behavior under some circumstances that are not specifically noted.

Errors with multiple equality indexes with different last element class

Previously, it was allowed to create two equality indexes on the same path with different classes specified as lastElementClass. This resulted in failures later when performing index updates. Multiple indexes with different lastElementClass are now disallowed. (#43085)

Multiple-predicate conjoined queries on set-valued/enumerated indexes

The internal implementation of set-valued and enumerated path term indexes does not properly support multiple predicates conjoined using the and operator (&). Such predicates with set-valued queries were unreliable in the 32-bit implementation, and have been disallowed in the new implementation in GemStone/S 64 Bit. This is recorded as bug #43764.

In general, attempting to perform a multi-predicate conjoined query on a set-valued or enumerated index will fail with an error. However, due to the effects of the new not operator and the or operator |, and the possibility of formula changes due to optimization, some such queries may work. Error messages now provide the details.

Removing identity index could remove equality index on special

When the lastElementClass for an index is a special, such as Boolean, the creation of an equality index creates the indexing structures that would implement an identity index. Removing that identity index incorrectly removed the equality index. (#43786).

Now, you can only remove indexes that you have explicitly created. However, indexed identity queries on specials that have an equality index will still be performed using the implicit index.

Indexes on invariant objects

When an index is created on an invariant object, which does not get a dependency list, some operations such as removing incomplete indexes and creating a second index may error. (#42643)

DependencyList not in SharedDependencyList

It was possible for DependencyLists to not be in the SharedDepedencyLists structure, and this was not caught by index audit. There was no error conditions resulting from this; the consequence was extra instances of DependencyList. This condition is now caught by audit, and repaired. (#41868)

removeAllIncompleteIndexes may error, preventing index removal

When an incompletely created index had been left with particular internal indexing structure variables that were nil or of size zero, removeAllIncompleteIndexes resulted in an error. #(43658)

Removing an entry from an index when collision buckets at full size may result in corruption

Indexes use an internal structure, an RcIndexDictionary, containing instance of RcIndexBucketWithCache. There is a bug that is exposed when the collision buckets are fully packed (1380 entries), such that removing an element may corrupt the RcIndexBucketWithCache, and thus the index. (#43801)

IndexManager did not autoCommit during DependencyList removal

When removing an index, the IndexManager autoCommit normally allows periodic commits. This was not occurring during execution while removing DependencyLists. With a very large number of DependencyLists, this could result in excessively large dirty sets. (#43647)

Indexing temporary collections containing persistent objects can lead to object audit errors

If a temporary collection containing persistent objects has an indexed created, it can lead to inconsistent state. Now, all indexes and indexing infrastructure on temporary collections is removed on abort. (#43703)

Bugs primarily affecting GBS

In GBS, could not evaluate block with an explicit return

Evaluating a block from GBS that contained an explicit return from the block returned an error, unable to find home context. (#42837)

Stepping into 3.x blocks did not work

Stepping through 3.x code did not step into blocks correctly. (#42940)

On first step into server context, cannot evaluate temporary variables

In the GBS debugger, before any step within a server context, GemStone was not able to resolve temporary variables in the method. (#42543, #42546)

The fix includes a new bytecode in the compiled server method, so there is an additional step point. Since this is added when the method is compiled, there is no change in behavior with existing compiled server methods. New or recompiled methods will include the step point, and will not have this problem.

Server contexts missing in GBS debugger during user action or GsFile server errors

If an error occurred during a UserAction (including GsFile operations, which are implemented as UserActions), the server context were not included in the debugger stack.This bug did not affect topaz, which included all stack frames. (#38477, #43672)

Numerics issues

Float emax/emin incorrect

The constants returned by the Float class methods emax and emin were incorrect; they returned the maximum expressed as a base 10 value, while the standard requires the base 2 value. The previous return values of 308/-308 are now 1024/-1021. (#43107)

Incorrect printing for MinusQuietNaN and MinusSignalingNaN

When these objects were printed, they displayed incorrectly as PlusQuietNaN and PlusSignalingNaN. (#43772)

Duration >> printOn: printed fractional seconds incorrectly

Duration did not correctly print fractional numbers of seconds. (#42705)

Float, SmallFloat -0.0 asString does not include sign

Float or SmallFloat negative zero, -0.0, printed without the sign, as '0.0000000E+00' and '0.0000000000000000E+00'. (#43815).
Note that in v3.2, the printed float exponent will not include the "+".

Non-SmallDouble 0.0 negated is not negative

Sending negated to the SmallFloat or Float 0.0 produced a non-negative SmallDouble. Now, the sign of the result is correct. (#43816).

Activating a passivated SmallFloat results in a SmallDouble

If a SmallFloat is passivated and reactivated, the recreated valus is a SmallDouble. (#44048)

countInstances: incorrectly generated .bm files

Executing `SystemRepository >> countInstances:` incorrectly also created .bm files in the current directory (#42677)

ProfMonitor results incomplete

Samples of the call stack taken by ProfMonitor only covered the top half of the stack; statistical summaries did not account for method calls in the lower half of the stack. (#43418)

Memory leak in Admin Gem

The Admin GcGem had a small memory leak. (#43529)

GemBuilder for C failed to signal user-defined ANSI exception

The GemBuilder for C function GciRaiseException() has the input argument GciErrStype, which includes a exceptionObj field; however, this was not used when signalling an exception. Now, this field may be set to an instance of a ANSI Exception. When this is done, that object will be signalled on exception, and the error number in that object will set the error number in the resulting error struct. (#42567)

Some Exception classes did not include legacy exception number

The following Exception classes inherited the error number from the superclass, and thus could not be mapped properly on the client:

```
CompileWarning
Deprecated
InterSessionSignal
TestFailure
ResuableTestFailure
```

Server contexts missing in debugger during user action or GsFile server errors

If an error occurs in a user action, or by extension a GsFile method which uses user actions, an error will be raised. However, the stack in the debugger will be missing server contexts. (#43672)

Method category changes silently failed without permission

If the userProfile adding a method category to a class, or making other similar changes in the method categories, did not have permission to modify the class, the operation failed but did not report an error. (#43424). Now, a permission error is reported.

The affected methods were:

```
Behavior>>addCategory:environmentId:
Behavior>>moveMethod:toCategory:environmentId:
Behavior>>renameCategory:to:environmentId:
Behavior>>renameOrMergeCategory:to:environmentId:
Behavior>>removeCategory:environmentId:
```

This issue affected method category operations in GBS.

compileMissingAccessorMethods did not work on Class instance variables

Previously, this method could not be used on Class instances to compile class instance variables. (#43584)

Also, the methods compileMissingAccessorMethods and compileAccessingMethodsFor: have been changed, so the methods that are generated by execution do not include the generic comment.

Code after a Delay in ensure: block never executed

The code that handles an ensure block was terminating the GsProcess, so code after a Delay in an ensure block was never reached to execute. (#42681)

Tranlog analysis cluttered with user:NULL

User information is written to the tranlogs when a user logs in; after that point, only a session id is used. Since tranlog analysis scripts such as printlogs operate on a specific set of tranlogs, it is likely that for some transactions within the set of tranlogs, no user details are available.

Previously, transactions by sessions that started prior to the tranlog range resulted in output with user: NULL. Such entries are no longer included.

Note that when printlogs performs filtering on specific user information, it cannot match transactions within sessions that logged in prior to the beginning of the specified tranlog range with that user. (#43185)

Incorrect result set for myCacheStatistics

This method incorrectly returned the full set of cache statistics corresponding to the descriptions returned by System class >>cacheStatisticsDescription.

It now correctly returns the smaller set of cache statistics corresponding to the descriptions returned by System class >> cacheStatisticsDescriptionForGem.

ClassOrganizer did not correctly handle subclasses of nil

Creating subclasses that are subclasses of nil caused methods in ClassOrganizer to error or recurse. (#13156)

DateTime asJson stack overflow

Instances of DateTime did not respond correctly to the methods to create their json representation. (#42411)

DateAndTime bugs

DateAndTime fromString: did not read fraction seconds correctly

The output of DateAndTime printString includes fractional seconds. This output was not read in correctly by DateAndTime fromString:. (#42920)

DateAndTimeANSI >> = failed for subclasses other than DateAndTime

Equality comparisons failed for instances of subclasses of DateAndTimeANSI other than DateAndTime. This was the only GemStone subclass, and affects application specific subclasses of DateAndTimeANSI. (#42704)

DateAndTime timeZoneName, abbreviation

The methods DateAndTime >> timeZoneName and DateAndTime >> timeZoneAbbreviation errors on non-UTC instances. (#43388)

VSD dropdown list in wrong place with multiple monitors

When running with two monitors, if the main monitor is on the right, dropdown lists for windows located on the secondary monitor were in the wrong location. (#37808)

Topaz duplicate prompts when stdout redirected

When output from topaz stdout was redirected, it would write duplicate prompts. (#43197)

Startstone failure with shrpcmon invalid ftok parameter

On AIX, startstone could have failed occasionally fail due to a problem with the startup of the shrpcmonitor process. (#43269)

Process hang in shmdt() on cache detach

On POWER7 hardware, especially with hyperthreading, processes may hang for several seconds in shmdt() on cache detach. To avoid this system call, set

```
export GS_DISABLE_SHMDT=1
```

If not in quiet mode, a message similar to the following is printed to stdout at cache disconnect:

```
[Info]: Skipping shared memory segment detach system call
shmdt()
```

Client hot hangs on network connect issue

When connected remotely, GCI applications (such as GBS) could hot hang due to network interruptions. (#42456)

Risk of SEGV in Repository >> _findOopsOnPages:

Due to a wrong variable declaration, this method could SEGV. (#43150)

Long queue lock sleep times resulted in contention

The amount of time that the spin lock code sleeps on queue locks has been reduced and made platform specific, to reduce possible contention on queue locks. (#42978)