# GemStone/S 64 Bit™ Release Notes

## Version 3.7

September 2023

GEMTALK™ SYSTEMS

**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 3.7 release. Read these release notes carefully before you begin installation, upgrade, 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 3.7.

## 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 LLC, previously GemStone Systems, Inc. and a division of VMware, Inc.

## Technical Support

### Support Website

**gemtalksystems.com**

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

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

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

We recommend checking this site on a regular basis for the latest updates.

### Help Requests

GemTalk Technical Support is limited to customers with current support contracts. Requests for technical assistance may be submitted online (including by email), or by telephone. We recommend you use telephone contact only for urgent requests that require immediate evaluation, such as a production system down. The support website is the preferred way to contact Technical Support.

**Website: techsupport.gemtalksystems.com**

**Email: techsupport@gemtalksystems.com**

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

Please include the following, in addition to a description of the issue:

- The versions of GemStone/S 64 Bit and of all related GemTalk products, and of any other related products, such as client Smalltalk products, and the operating system and version you are using.
- Exact error message received, if any, including log files and statmonitor data 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.

# Table of Contents

# Chapter 5. Bug Fixes

Chapter

1

# Release Notes for 3.7

## Overview

GemStone/S 64 Bit™ 3.7 is a new version of the GemStone/S 64 Bit object server. Version 3.7 includes a number of important new features, including compiler optimizations, changes to process scheduling and debugging, new FileSystem support, multithreaded instance migration, customer special classes, and other new and enhanced features and bug fixes.

These Release Notes include changes between the previous version of GemStone/S 64 Bit, v3.6.6, and v3.7. If you are upgrading from a version prior to 3.6.6, review the release notes for each intermediate release to see the full set of changes.

For details about installing GemStone/S 64 Bit 3.7 or upgrading from earlier versions of GemStone/S 64 Bit, see the *GemStone/S 64 Bit Installation Guide* for v3.7 for your platform.

### New keyfiles required

The keyfiles for v3.6.x and earlier cannot be used with v3.7; new keyfiles are required for this release. To obtain a new keyfile for GemStone/S v3.7, write to keyfiles@gemtalksystems.com. In your request, include your license information, platform and any updates to contact information.

Please contact GemTalk Technical Support if you have issues or questions.

## Supported Platforms

### Platforms for Version 3.7

GemStone/S 64 Bit version 3.7 is supported on the following platforms:

- Red Hat-compatible Linux 7.9, 8.7, and 9.2, and Ubuntu 20.04 and 22.04, on x86; Ubuntu 20.04 on ARM (Ubuntu on ARM is supported for development only) GemStone is tested on a mixture of Red Hat, CentOS, Rocky and Alma; these are all considered fully certified platforms. Any reference to Red Hat applies to any Red Hat-compatible distribution.
- AIX 7.1 and 7.2
- macOS 12.6.8 (Monterey) and 11.7.9 (Big Sur) on x86; macOS 13.4.1 (Ventura) and 11.7.8 (Big Sur) on Apple silicon (ARM)

Note that GemStone/S 64 Bit v3.7 was built using compilation features that are not available on older CPUs (more than about 10 years old); v3.7 will not run on these CPUs, regardless of the Linux OS version. See page 21 for more details.

Distributions for Solaris/x86 and Solaris/SPARC are no longer available.

For more information and detailed requirements for each supported platforms, please refer to the *GemStone/S 64 Bit v3.7 Installation Guide* for that platform.

X509-Secured GemStone feature is tested and supported on Linux platforms only.

### GemBuilder for Smalltalk (GBS) Versions

The following versions of GBS are supported with GemStone/S 64 Bit version 3.7:

#### GBS/VW version 8.7

| **VisualWorks 9.1.1 32-bit and 64-bit** |
|---|
| • Windows 10<br>• RedHat ES 7.9, 8.7, and 9.2; Ubuntu 20.04 and 22.04 |

#### GBS/VA version 5.4.7

| **VAST Platform 11.0.1** | **VAST Platform 10.0.2** | **VA Smalltalk 8.6.3** |
|---|---|---|
| • Windows Server 2016 and Windows 10 | • Windows Server 2016 and Windows 10 | • Windows Server 2016 and Windows 10 |

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

## VSD Version

The GemStone/S 64 Bit v3.7 distribution includes VSD version 5.6.1. The previous version of GemStone/S 64 Bit, v3.6.6, included VSD v5.6.

VSD version 5.6.1 includes improved handling for appended files. For details on the changes, see the Release Notes for VSD v5.6.1.

With GemStone/S 64 Bit v3.7, **statmonitor** uses an additional optimization file to improve lz4 compression; this file is used by VSD when reading an lz4-compressed statmonitor data file. Further details on this change can be found on page 47.

As a result, lz4-compressed statmonitor files from v3.7 cannot be read by versions of VSD earlier than v5.6. VSD 5.6.1 can read statmonitor files generated in older versions of GemStone/S 64, 32-bit GemStone, and GBS, as well as those generated by GemStone/S 64 Bit v3.7.

VSD 5.6.1 is included with the GemStone distribution, and can also be downloaded as a separate product from https://gemtalksystems.com/vsd/

## GemBuilder for Java (GBJ) Version

The most recent version of GBJ, v3.1.3, can be used with GemStone/S 64 Bit v3.7.

GBJ 3.1.3 has not been tested on AIX with v3.7, although it is expected to work, and the distribution on AIX does not include the GBJ shared library. Contact GemTalk Technical Support if you require GBJ on AIX.

## GemConnect Version

The most recent version of GemConnect, v2.4, can be used with GemStone/S 64 Bit v3.7.

## Rowan

The GemStone/S v3.7 distribution includes Rowan v 2.5.

## Sparkle

Sparkle, the Pharo IDE for GemStone, provides GemStone development tools in the Pharo client Smalltalk environment. GemStone/S 64 Bit v3.7 is required with Sparkle.

Sparkle is under active development, and there are ongoing changes in GemStone server code to support Sparkle.

See the *Installation guide for Sparkle* in the Documentation subdirectory of the project, under https://github.com/GemTalk/Sparkle/.

## Upgrade

Upgrade is supported from all 3.4.x, 3.5.x, and 3.6.x versions. Upgrade from 3.3.x is not disallowed, but is not tested or supported. To upgrade from earlier versions, upgrade to a supported upgrade origin version, and then upgrade to 3.7.

Note that **JsonParser** has changed definition in v3.7, and **GsHostProcess** changed definition in 3.6.5. If you have subclasses of these classes, or methods that reference either class directly or by name, you may need to recompile methods and/or edit source code, to ensure these methods work as intended. See the *Installation Guide for v3.7* for details.

Note that several obsolete or deprecated configuration parameters have been removed. These will now error when read, rather than being ignored. You may need to update legacy configuration files if you have not done so previously. See “Removed configuration parameters” on page 45.

For large systems with many symbols, in which upgrade performance is critical, we are now recommending to run `System clusterAllSymbols` in the old environment, prior to upgrade.

### isNil/notNil

GemStone/S 64 Bit v3.7 includes new configurable optimized selectors, described on page 21; these optimized selectors include `isNil`, `notNil`, and `yourself`. If your application includes method overrides to implement `isNil` or `notNil`, you will need to either modify your application, or disable the affected optimized selectors. Instructions for this are in the *Programming Guide* and are summarized on page 21.

## Seaside and GsDevKit

The code for webSocket, which is optional but used in Seaside/GLASS, includes classes that reimplement `isNil`. As a result of the `isNil/notNil` issue, the `isNil` selector is **not** optimized in the distribution seaside extent, nor when bootstrapping GLASS into extent0.dbf. See https://github.com/GsDevKit/GsDevKit_home/issues/331 for more information.

The **upgradeSeasideImage** has been updated to disable optimization of `isNil` by default. The **-E** argument has been added to **upgradeSeasideImage**, which bypasses the step of disabling optimization (and thus results in `isNil` remaining optimized). This argument can be used if your application does not override or load code with a custom `isNil` implementation.

## Documentation Changes

Documentation has been revised for this release, with modifications to incorporate new and changed features, as well as corrections and improvements.

In addition to the maintenance changes, and addition of new information and features, the following improvements have been made:

- The *Topaz User's Guide* has an additional chapter describing scripting, including existing shell script information and new superDoit scripts.
- The *System Administration Guide* section on Hot Standbys has been updated, and the Appendix describing Configuration File handling has been rewritten for completeness and clarity.
- The *Programming Guide* has a new chapter for File handling, including GsFile as well as File System; and the FFI chapter has been extensively revised.
- A new Installation Guide specific for Linux RPM installations has been added, the *Installation Guide for Linux RPM*.

A number of recently added features are documented in Supplemental Documentation; see awskeymanagement, monitorwithprometheus, upgradeViaHotStandby, and the new AzureKeyVault.

The *GemStone/S 64 Bit X509-Secured GemStone System Administration Guide* has not been updated for this release.

## 1.1 Library and Distribution changes

### Updated Library Versions

The version of OpenSSL has been updated to 3.0.10. This is a major version update from 3.6.x.

The version of OpenLDAP has been updated to 2.6.4.

The version of libssh has been updated to 0.10.5.

The version of Kerberos has been updated to 1.20.1.

The version of lz4 has been updated to 1.9.4.

The version of prometheus has been updated to 1.0.1.

The version of aws-sdk-cpp has been updated to 1.9.362.

The version of zoneinfo has been updated to 2022d.

The version of zlib has been updated to 1.2.12.

#### New libraries

A new library is now included, double-conversion, v3.2.1.

On Linux/x86 only, the following libraries have been added:
azure-sdk-cpp 1.9.0
curl 8.1.1

### RPM distributions available

RPM (Red Hat Package Manager) distributions are now available.

RPM installations create a shared read-only distribution, in which it is not appropriate to use the `$GEMSTONE/data` directory for configuration files, extents, transaction logs, and other repository-specific files. Likewise, using the default settings in $GEMSTONE/data/system.conf is not appropriate.

When using an RPM installation, you must explicitly specify a configuration file location, and the configuration file must specify an application-specific directory in which the extents and transaction logs will be located.

Two features have been added to improve support basic GemStone RPM installations:

- The new **-E** *configFile* option has been added to many of the utilities; this is similar to **-e**, but with a simplified search path. In addition, an environment variable $GEMSTONE_DATADIR is defined as the directory in which the **-E** configuration file is located.
- The distribution now includes an additional template configuration file, `$GEMSTONE/bin/gemstone_data.conf`. This template defines the default extent and transaction log locations using $GEMSTONE_DATADIR, rather than $GEMSTONE.

#### RPM Installation process

After installing GemStone via RPM, a simple GemStone environment can be setup by:

- Create a directory for all GemStone files, *gemstoneApplicationDir.*
- Copy `$GEMSTONE/bin/extent0.dbf` and `$GEMSTONE/bin/gemstone_data.conf` to this directory, and give these files write permission.
- Execute **startstone**, including **-E** *gemstoneApplicationDir*`/gemstone_data.conf.`

See the new *Installation Guide for Linux RPM* for more detail.

### clientLibs distributions

Along with the GemStone distribution, GemStone now includes a clientlibs directory tree. This provides a way to flexibly handling multiple versions of client library files. The structure of this directory tree is:

```
/clientlibs
        versionNum
                64bit
                        specific library files
                32bit
                        specific library files
```

When a new version of the server is released, the *newVersionNum* tree from that release can be added to the `clientlibs` directory without perturbing other uses.

### $GEMSTONE/projects moved

Earlier 3.6.x releases included the `projects` directory under `$GEMSTONE/upgrade/`. This directory includes rowan-format source code for projects such as superDoit.

This directory has been moved up and is now a base directory, `$GEMSTONE/projects/`.

### Further changes in permissions within the distribution

The GemStone distribution included subdirectories and files with incorrect permissions; primarily non-executable files with execute permission; and some cases of subdirectories with owner write. These permissions have been corrected.

`$GEMSTONE/data` and `$GEMSTONE/ualib` continue to have owner and group write permission.

## 1.2 OS Configuration and Installation Changes

### Using setcap for large pages and OOM protection on Linux

In previous releases, the *Installation Guide* has specified operating system configuration steps to enable use of large memory pages and for protecting processes from the OOM killer, using setcap to give the executables specific Linux capabilities. This is not optimal for security, and has a side effect of disallowing stack traces, since gdb did not have privileges to attach to the process, and prevented statmonitor from reading certain memory statistics. (#50580)

Alternate ways to configure large memory pages and OOM protection have been determined, and the way GemStone uses capabilities has been modified for v3.7. The previously existing techniques work, but for security and usability reasons are no longer recommended.

### Large Memory Pages

Previously, it was recommended to give the Shared Page Cache Monitor the cap_ipc_lock capability.

Now, you can avoid giving this capability, by instead selecting and configuring a huge pages group. This is the procedure:

Select or create a group that includes the Linux user that will be starting up the shared page cache, and all users who will be accessing the shared page cache.

Then create the file `/etc/sysctl.d/64-gemstone-local.conf`, to set this group's numeric id to the huge pages group. Add a line to the file:

```
vm.hugetlb_shm_group = numericGidOfGroup
```

You can determine the gid of a group from the name using:

```
os$ getent group groupName
```

This will take effect on reboot; you can apply immediately by executing:

```
os$ sysctl --system
```

To verify the current value, cat `proc/sys/vm/hugetlb_shm_group`, which will show the group id. To enable large pages transiently without rebooting, you can set the gid of the group directly in `/proc/sys/vm/hugetlb_shm_group`. This will be reset on reboot if not configured as by `/etc/sysctl.d/64-gemstone-local.conf`.

For full details, see the updated instructions in the *Installation Guide for v3.7.*

### OOM Killer protection

Previously, a number of executables were given the cap_sys_resource capability, to allow them to modify their oom_score_adjust; critical processes reduced their scores, and less critical processes, such as gems, increased their score.

Now, only the Stone itself, and the pgsvrmain for remote configurations, need the capability. The only setcap executions needed are:

```
os$ setcap cap_sys_resource=pe $GEMSTONE/sys/stoned
os$ setcap cap_sys_resource=pe $GEMSTONE/sys/pgsvrmain
```

On startup, these processes use this capability to set their oom_score_adjust, then release the capability. Other critical processes are spawned by the Stone and inherit the Stone's oom_score_adjust, or for remote caches, are spawned by the pgsvrmain and likewise inherit oom_score_adjust. The spawned processes that inherit an oom_score_adjust do not report the protection in their log files. The scores for Gems can be safely left at the default.

The GemStone code changes to release the capability, avoid the previous issues with gdb and statmonitor, including for processes that have explicit setcap.

### Large Memory Pages and Locking SPC in memory

If you configure your Shared Page Cache to use large memory pages, the cache is inherently then locked into memory. There is no need to configure the administrative user to allow locking using SHR_PAGE_CACHE_LOCKED.

## Example scripts for setting up GemStone in systemd on Linux

Under `$GEMSTONE/examples/admin/`, the following directory and files have been added:

```
/systemd
   netldi.service
   gemstone.env
   netldi.env
   gemstone.service
```

These scripts include the details needed to set up the Stone and NetLDI to run as services managed by systemd.

Note that, whether using systemd or not to run GemStone, `/etc/systemd/logind.conf` should include:

```
RemoveIPC=no
```

to avoid Stone shutdown when the ssh or gui session that started the Stone logs out, and the Stone's shared semaphore array is deleted.

## Raw partitions on Character devices no longer supported

The RAW driver for direct I/O has been deprecated for many years (superseded by the O_DIRECT flag for a block device). With Linux kernel 5.14, the RAW driver was dropped; this or later versions are used by Ubuntu 22.04 and RedHat 9.x.

With v3.7, raw partitions on character devices are no longer supported for GemStone dbf files, on older as well as recent kernels. v3.7 supports raw partitions on block devices.

## Reverse DNS lookups required for Single Sign-On (SSO)/Kerberos logins

In order to use SSO logins using Kerberos, it was, and is, required that you have reverse DNS lookup setup and operating correctly; see the informational bugnote for #50507.

The error reporting for this case has been improved, and SSO tracing enabled; see "Debug tracing for SSL, SSO, SSH now allowed in fast builds" on page 39.

## 1.3 Process and Debugger Optimizations

## New native code generation and compiler optimizations

The native code generator has been extensively improved for performance, and the compiler has been further optimized.

Some operations, such as integer arithmetic, have 2x-5x performance improvements, overall a 10% improvement can be expected.

Note that operations that are not CPU-bound, i.e. that are I/O bound waiting on disk reads or writes, will not observe performance improvements.

### Native code not reimplemented on AIX, Mac/Linux on ARM

In this release, AIX, Linux on ARM, and Darwin on Apple silicon (ARM) run in interpreted mode only.

Native code is available for Linux and Mac on x64_64.

### Older CPU machines cannot run v3.7

GemStone has been compiled using optimizations that require CPU features that are not available on older CPUs (more than about 10 years old).

The minimum required for 3.7:

- Intel CPUs with architecture Sandy Bridge or newer (excluding Atom CPUs)
- AMD CPUs with architecture Bulldozer version 1 or newer

v3.7 cannot run on older CPUs. Attempting to run any executable will result in an error such as:

```
[Error]: Program cannot start because the CPU is missing the
following feature(s):
[Error]: avx pclmul
[Error]: For more information, please contact GemTalk technical
support.
```

## Conditionally Optimized Selectors

The selectors **`isNil`**, **`notNil`**, and **`yourself`** are now optimized by default. The optimization is configurable, and can be removed.

The new method `GsNMethod class >> configurableOptimizedSelectors` returns the optimized selectors.

To remove the optimization, as SystemUser you must reset GsNMethods's class variable OptimizedSelectors to an Array containing the selectors that will be optimized (minus any you do not want optimized). Symbols other than the three specified will be ignored. You may use an expression such as `GsNMethod _classVars at: #OptimizedSelectors put:` *anArray*.

To update the bytecodes in compiled methods, run **upgradeImage** or otherwise recompile all GemStone kernel methods, and recompile all application methods.

## Process Scheduling and Debugger Support

GemStone/S 64 Bit v3.7 includes additional infrastructure and changes in process scheduling and management, breakpoints, and stepping from v3.6.x. These differences includes both additional methods, and behavior changes in specific methods.

### Scheduler now preemptive

The scheduling of a higher priority process previously required an explicit yield or wait in GemStone; now, GemStone does preemptive scheduling, similar to the behavior in Pharo and other Smalltalk dialects.

For example `ExecBlock>>forkAt:` no longer requires a subsequent yield, to allow a high priority GsProcess to start running. `GsProcess >> resume` and `GsProcess >> priority:` will automatically yield, if the receiver is or becomes higher priority than the current process.

Code with coordinating multiple processes may see differences in behavior in this version of GemStone.

## Breakpoint Handling and Debugger Support

The GsProcess, Behavior and GsNMethod API that supports breakpoints and debugging has been redesigned, to improve and facilitate debugger implementation. Additional exception codes have been added to distinguish signaling to the GCI vs. signaling to Smalltalk. These changes are likely to affect behavior in corner cases, but should be generally transparent to end users. If you have implemented a debugger or made extensive customizations, contact GemTalk Engineering for more information.

## Interrupting sockets

When a socket is waiting on activity (such as read or accept) in one GsProcess, while another GsProcess is running, previously the socket that is waiting could not get scheduled when the wait was complete and it was ready for further execution.

Now, a socket can be configured as interrupting. The following methods have been added:

```
GsSocket >> interrupting
GsSocket >> interrupting: aBoolean
```

After executing *aSocket* `interrupting: true`, then a GsProcess waiting for that socket to be ready for read or write in an instance method of GsSocket or a subclass of GsSocket will be scheduled to run when that socket is ready.

## Process and debugging bugs fixed

A number of bugs have been fixed in the underlying code. These issues may or may not have been in previous releases.

- Step over in recursive method stopped in a frame of the same selector, which may be incorrect (#49539)
- Process terminate may be ignored if unwind block continuously runs. (#49464)
- `GsProcess >> suspend` did not suspend a process waiting for a delay or on a semaphore (#49526)
- `GsProcess >> resume` resumed a process that was waiting on a delay but not suspended (#49802)
- Terminated GsProcess does not allow ensure block in progress to complete, risking stuck semaphore (#48271)
- After AlmostOutOfStack, attempting to execute further encountered error clearing stack (#47373)
- GsProcess terminate on a process waiting on a socket without a timeout was not terminated correctly. (#47196)

## Handling asynchronous events in a designated process

In a multi-process application, you may now configure a specific process to receive and handle these signals. The following methods have been added. For an example of how these can be used, see `GsSignalingSocket class >> _readNotificationExample`.

`GsSignalingSocket class >> newForAsyncExceptions:` *anArray*

Returns a kind of GsSignalingSocket which has been registered to receive data representing asynchronous Notifications. The socket returned represents one end of an underlying UNIX domain socket pair, the other end of that pair is written to by the C thread that is the source of the Notification.

*anArray* must contain one or more of the classes InterSessionSignal, ObjectsCommittedNotification, TransactionBacklog, and GcFinalizeNotification, to specify the type of Notification to receive. Use `GsSignalingSocket >> readNotification` to receive the Notifications.

Do not use `InterSessionSignal class >> enableSignalling`, `ObjectsCommittedNotification class >> enableSignalling`, or `TransactionBacklog class >> enableSignalling` in conjunction with this method.

Only one socket can be registered in a session to receive asynchronous Notifications. `GsSignalingSocket class >> disableAsyncExceptions` must be used to cancel a previous registration before the next execution of `newForAsyncExceptions:`.

`GsSignalingSocket >> readNotification`

Read a notification from the receiver, which must be the result of executing `GsSignalingSocket class >> newForAsyncExceptions:`.

`GsSignalingSocket class >> disableAsyncExceptions`

Cancel a registration for receiving asynchronous notifications.

## Added and renamed Process and related methods

### GsProcess "Step" methods renamed

The methods `GsProcess >> step*FromLevel:*` have been renamed to more accurately describe their function; the new names are `setStep*BreaksAtLevel:*`. The earlier methods are still present and usable, but deprecated.

### Signaling the GCI

The following methods have been added:

`AbstractException >> signalToGci`

Receiver is signaled and it will not be trappable by any exception handler. An Error will be returned to the GCI.

`AbstractException class >> signalToGci`

An instance of the concrete subclass is created and signaled; not be trappable by any exception handler. An Error will be returned to the GCI.

### Other added methods

```
GsProcess >> abandonUnwindAndTrimStackToLevel:
GsProcess >> breakpointLevel
GsProcess >> breakpointLevel:
GsProcess >> terminateTries:eachTimeoutMs:
GsProcess >> trimStackToLevel:
GsProcess >> unsafeTerminate
GsProcess >> unsafeTrimStackToLevel:
GsProcess class >> current
GsMethod >> setBreakAtStepPoint:breakpointLevel:
ProcessorScheduler class >> highestPriority
ProcessorScheduler class >> lowestPriority
```

## Removed Methods

The following related methods have been removed:

```
Breakpoint class >> trappable:
ExecBlock >> valueNowOrOnUnwindDo:
```

Other removed methods are listed under “Deprecated and Removed Classes and Methods” on page 82.

Chapter

# 2 Changes in Administration

This chapter describes changes and new features that apply to the administration of a GemStone Repository, and changes in the GemStone environment, configuration, and tools, including:

## 2.1 One-time Password

### One-time passwords allow login without further authentication

A new feature has been added, to allow a session to create a one-time, time-limited password for a specific UserProfile, which will allow a login as that user without further authentication. This allows, for example, writing code that uses GsExternalSession to login other sessions to divide tasks over multiple sessions, without needing to hard-code the GemStone password in code or query for it on the command line. The current session's authentication is "carried over" for a one-time authentication without re-entering a password. One time passwords can be used to login as the same user as the current session or another user configured on the session’s allowlist of UserProfiles.

Topaz and GsTsExternalSession/GsExternalSession include additional methods to support one-time passwords. With GemBuilder for C, you must specify the new flag GCI_ONETIME_PASSWORD with the login parameters using GciLoginEx() or GciLoginEx_(). GemBuilder for Smalltalk products do not yet support one-time password login.

### Privilege to create one-time passwords

Only userIds with the new privilege #CreateOnetimePassword can create one-time passwords. This is granted by default to SystemUser and DataCurator.

Temporary passwords cannot be created to login as SystemUser.

### One-time passwords for other UserProfiles

In addition to being able to login a second session as your own UserProfile, one-time passwords can also be created for other UserProfiles. Creating a one-time password for another UserProfile requires that the other UserProfile be on this session’s allowlist of UserProfiles.

Adding a UserProfile to the allowlist requires #OtherPassword privilege. Once the whitelist is updated, #OtherPassword is not needed to create a temporary password, only #CreateOnetimePassword is needed.

### Creating and Using a one-time password

Methods to create a one-time passwords are in GsCurrentSession. The one-time password is registered in the Stone, but not persisted (no commit is needed). The UserId and this one-time password can be used for a single login within the specified time window. After the successful login or the timeout has elapsed, the one-time password is no longer usable.

The GemStone GCI login functions GciLogin() and GciTsLogin() include a new flag, GCI_ONETIME_PASSWORD, that must be set in order to perform a login using a one-time password. This can be set using new methods in GsTsExternalSession and GsExternalSession, and a new command in topaz. The one-time password can be used to login using GsTsExternalSession or GsExternalSession (the expected use), and can also be used to login using topaz (linked or RPC). Other tools can be used provided that they support the GCI_ONETIME_PASSWORD flag.

**Example 2.1 Using GsTsExternalSession to perform a login as SubordinateUser**

The following example is executed as DataCurator, and presumes the existence of a GemStone UserProfile with the userId SubordinateUser.

First, DataCurator adds SubordinateUser to their allowlist:

```
System myUserProfile addOnetimePasswordUserId:
   'SubordinateUser'.
```

Then, a temporary password is created:

```
tempPW := GsCurrentSession currentSession
   createOnetimePasswordForUserId: 'SubordinateUser'
   validForSeconds: 30.
```

To use this password to login an external session:

```
sess := (GsTsExternalSession newDefault)
   username: 'SubordinateUser';
   onetimePassword: tempPW;
   yourself.
sess login.
```

### New topaz command

To use one-time passwords in topaz, a new command has been added.

**set onetimepassword** *onOrOff*
Determines if the password is a normal GemStone password or a special one-time password. When OFF, the default, the password field is interpreted as a normal GemStone password. When ON, the password field is interpreted as a onetime password.

## Tracking one-time passwords

### Additional fields in Login log

The login log feature allows you to enable logging of logins and logouts for a repository, using the STN_LOGIN_LOG_ENABLED configuration parameter.

The file generated when login logging is enabled now includes two additional fields for each entry:

ParentUserProfile - OOP of the UserProfile that created the token used for login or 20 (OOP_NIL) if no token was used.

ParentPid - process ID of the parent session that created the token used for login or -1 if no token was used.

One-time logins that fail to login due to a mismatch in UserId and one-time password, or an invalid or expired password are also reported.

### Additional fields in descriptionOfSession: result

The results of `System >> descriptionOfSession:` now includes additional fields (note that a new entry 25 has also been added; see page 38):

26. UserProfile which created the onetime password for the session, or nil if no onetime password was used.

27. Process ID of the gem that created the onetime password for the session, or -1 if no onetime password was used.

### Added Cache Statistics

The follow cache statistics have been added:

**OnetimePasswordsActive** (Stn)
Number of onetime passwords in the stone's hash map.

**OnetimePasswordsCreated** (Gem)
Number of onetime passwords this session has created.

**OnetimePasswordsExpired** (Stn)
Number of expired onetime passwords automatically removed by stone.

**OnetimePasswordsNotValidated** (Stn)
Number of failed onetime password validations.

**OnetimePasswordsTotal** (Stn)
Number of onetime passwords ever created by any session since the stone was started.

**OnetimePasswordsValidated** (Stn)
Number of successful onetime password validations.

**OnetimePasswordUsed** (Gem)
A boolean with value 1 if the session used a onetime password to login, otherwise 0.

## New privilege

The privilege #CreateOnetimePassword has been added. This privilege is granted to DataCurator and SystemUser, but existing and newly created users will not have this privilege.

## Added methods

`GsCurrentSession >> createOnetimePasswordForUserId:` *aUserId*
`validForSeconds:` *seconds*

`GsCurrentSession >> createOnetimePasswordForUserProfile:` *aUserProfile*
`validForSeconds:` *seconds*

`GsCurrentSession >> createOnetimePasswordValidForSeconds:` *seconds*

`GemStoneParameters >> onetimePassword: pword`

`GemStoneParameters >> onetimePasswordFlag`

`GemStoneParameters >> onetimePasswordLoginFlags`

`GemStoneParameters >> setLoginWithOnetimePassword`

`GsExternalSession >> onetimePassword:` *aString*

`GsTsExternalSession >> onetimePassword:` *aString*

`UserProfile >> addOnetimePasswordUserId: aUserId`

`UserProfile >> addOnetimePasswordUserProfile:` *aUserProfile*

`UserProfile >> addOnetimePasswordUserProfiles:` *anArray*

`UserProfile >> onetimePasswordUserProfiles`

`UserProfile >> removeAllOnetimePasswordUserProfiles`

`UserProfile >> removeOnetimePasswordUserId:` *aUserId*

`UserProfile >> removeOnetimePasswordUserProfile:` *aUserProfile*

`UserProfile >> removeOnetimePasswordUserProfiles:` *anArray*

## 2.2 Changes in Users and ObjectSecurityPolicies

### CreateOnetimePassword privilege

This new privilege allows creating a one-time password that allow another session to login; see “One-time Password” on page 25 for details.

### MigrateObjects privilege

This privilege was present in earlier versions, but not implemented. This privilege is required for the new optimized instance migration; see “Multithreaded Instance Migration” on page 60.

## 2.3 Changes in Backup and Restore

### Full control over compression levels

GemStone programmatic fullBackup and secureFullBackup allow you to compress backups using gzip (zlib) or lz4. Generally speaking, lz4 backups are faster but do not compress as much.

Both gzip and lz4 support different compression levels, with lower compression level values providing faster performance with less compression. gzip supports compression levels 1 to 9, while lz4 supports 0 to 12. For GemStone backups, the upper ranges provide little addition compression and take much longer.

#### Consistent default for Lz4 to level 0

In previous releases, the default for lz4-compressed fullBackup was level 3. Since the reason for using lz4 rather than gz is generally performance, this has been changed so the default is now level 0. This is the same compression level as the default for secure lz4 compressed backup.

#### Specifying level in existing methods

The existing methods:

```
Repository >> fullBackupTo: fileNames MBytes: mByteLimit
    compressKind: anInt bufSize: count
```

```
Repository >> secureFullBackupTo: fileNames MBytes: mByteLimit
    compressKind: anInt bufSize: count encryptKind: encKind ...
```

now accept for *anInt* not only 0, 1 (for gzip), and 2 (for lz4), but also additional values to indicate compression level. 101... 109 indicate various gzip compression levels, and 200--212 indicate the various lz4 compression levels.

#### New methods including level argument

New methods have been also added to allow you to specify the particular level of compression you wish to use. These methods are variants of existing methods with the added `compressLevel:` argument.

`Repository >> fullBackupGzCompressedTo:` *fileNames* `MBytes:` *mByteLimit*
`compressLevel:` *cLevel*

`Repository >> fullBackupLz4CompressedTo:` *fileNames* `MBytes:` *mByteLimit*
`compressLevel:` *cLevel*

`Repository >> secureGzFullBackupTo:` *fileNames* `MBytes:` *mByteLimit*
`compressLevel:` *cLevel* `bufSize:` *count* `encryptKind:` *encKind*
`publicKeyCerts:` *anArrayOrString* `signatureHashKind:` *hashKind*
`signingKey:` *signingKeyFn* `signingKeyPassphrase:` *aPassphrase*
`numThreads:` *numThreads*

`Repository >> secureLz4FullBackupTo:` *fileNames* `MBytes:` *mByteLimit*
`compressLevel:` *cLevel* `bufSize:` *count* `encryptKind:` *encKind*
`publicKeyCerts:` *anArrayOrString* `signatureHashKind:` *hashKind*
`signingKey:` *signingKeyFn* `signingKeyPassphrase:` *aPassphrase*
`numThreads:` *numThreads*

### Repository scan operations now allowed in solo sessions

A number of repository scan operations that were previously disallowed in solo sessions can now be used.

This includes Repository methods for `allInstances`*, `listInstances`*, `allReferences`*, `listReferences`*, and related methods; and GsObjectInventory scans.

### In restore mode, compiler now generates arguments and temporaries as Strings

When a repository is in restore mode, it cannot create new Symbols, since Symbols are canonicalized and require the SymbolGem to commit, which would create a state difference between the restoring repository and the repository it is restoring from. This created problems executing code that includes arguments or temporary variables with names that did not already exist in the repository, since variable names are created as symbols for efficiency.

Now, when the repository is in restore mode, temporary variable names in compiled code are created as Strings, which will not require the SymbolGem commit. (#49984)

## 2.4 Hot Standby Changes

### Upgrade via hot standby

A process that allows upgrade of a hot standby configuration without having to recreate the slave was introduced in recent 3.6.x releases.

A number of methods have been added to make this process somewhat more streamlined; these added methods are describe in the following sections.

The documentation for this process is now provided as a supplemental document, at this location: https://docs.gemtalksystems.com/current/upgradeViaHotStandby.

Note that v3.7 includes a symbol, #OptimizedSelectors, that impacts upgrade via hot standby. If using the upgrade via hot standby process to upgrade from v3.6.x to 3.7.x, you must first resolve this symbol in the 3.6.6 environment. See the documentation for details.

## suspendCommitsForFailover no longer supported

`Repository >> suspendCommitsForFailover` should no longer be used; instead, use the alternate method `Repository >> failOverToSlave`.

`suspendCommitsForFailover` allowed you to keep the master system running as a master, while you created a forked duplicate system on the slave node. After `suspendCommitsForFailover` to setup the former slave as a master, you have two masters that are not compatible, since they have separate, incompatible tranlog sequences. With `suspendCommitsForFailover`, to setup the former master as a slave of the new master, you had to restore a new backup of the former slave, new master, into the former master.

If you do want to continue running the master as an independent fork along with the system running on the former-slave, execute `commitRestore` on the former master after executing `failOverToSlave`.

## Clarification on hot Standby with failover and multiple slaves

As in previous releases, it is possible for multiple slave stones to be restoring from a single master stone. While only one logsender (the **primary logsender**) can attach to the master stone, the primary logsender may service up to 5 slave logreceivers. Additional logsenders can run on the master node; these read from the tranlogs, but are not attached to the Stone, so the master Stone is not aware of them, and they are not notified if, for example, another tranlog directory is added.

The first Slave system’s logreceiver to connect to the Master’s primary logsender is the one for which `failOverStatus` is reported. There is no other difference between multiple slaves, other than this reporting information. If the system is restarted with the slaves connected in a different order, the `failOverStatus` information may not describe the same slave as before the restart. When you have multiple slaves, it is recommended to use the new **gslist** options (described on page 33) rather than `failOverStatus`.

A `failOverToSlave` operation prepares all slaves to become the new master; whichever slave performs the `commitRestore`, and the application API connects to, will be the new master. The other slaves can seamlessly connect to this new master.

## Repository >> failOverStatus improved report

The `failOverStatus` method results now include more details about the status of the master Stone executing it, and the slave Stone being restored into (the first to attach to the logsender, if there is more than one slave). This includes the most recent tranlog record as well as the most recent checkpoint, and other critical information on the master repository. This allows you to verify the exact restore status of the slave and compare to the status of the current master.

For example:

```
This repository last tranlog commit file 6 record 206
  last checkpoint file 6 record 15
Other repository restored to commit file 6 record 77
  last restored checkpoint file 6 record 15
This repository commits allowed
```

Note that this may be misleading if you have multiple slaves. If you have more than one slave, you may check on the intended failover slave itself, or use **gslist -j -v** *logsender*, as described on page 33.

## continuousRestoreFromArchiveLogs* now has configurable timeout

These methods available in previous releases waited for up to 20 minutes for the logreceiver to start writing log records, before giving up and returning an error. It is possible that very slow hosts under certain circumstances could take this long to start writing logs.

The timeout has been decreased to a more reasonable 60 seconds. Additional methods have been added that include an explicit timeout:

```
Repository >> continuousRestoreFromArchiveLogs:timeout:
```

```
Repository >> continuousRestoreFromArchiveLogs:
   withDelay:timeout:
```

These methods are similar to the existing methods, but will timeout and error after the given number of seconds.

## Terminating logsender and logreceiver programmatically

You may now stop the primary logsender and the logreceiver using Smalltalk expressions. The following methods have been added.

Note that these methods are in System class and require SystemControl privilege, as with any method that terminates server processes.

`System class >> killLogReceiver`

Causes stone to send SIGTERM to the log receiver process. Returns true if the SIGTERM signal was sent, false if no signal was sent because no log receiver process was running or a SIGTERM signal had already been sent.

`System class >> killLogSender`

Causes stone to send SIGTERM to the log sender process. Returns true if the SIGTERM signal was sent, false if no signal was sent because no log sender process was running or a SIGTERM signal had already been sent.

## commitRestoreForFailoverAfterWaitingUpTo: now stops logreceiver

This method now will stop the logreceiver, if it is running, as well as stopping continuous restore. (#50512)

## Full details in JSON from gslist

logsenders and logreceivers write keys into the locks directory, which allows them to be queried by **gslist**. This provides a way to see the status of all logsenders or logreceivers.

**gslist** has a new features which allows you to query for information on a logsender and all the logreceivers attached to it; using **gslist -j -v** (both options are required).

For example, the following is the output on the Stone's node, for a system running on host benton, with a primary slave on host fossa and a secondary slave on host santiam, on the stone's node. The "`statusInMasterCache:true`" indicates that this is the primary slave, whose details are reported by `failOverStatus`.

```
os$ gslist -v -j logsender_57222
{"GemStoneServers":[
        {
        "Name":"logsender_57222",
        "Host":"benton",
        "HostId":"69621bb0476b1937",
        "Ip":"10.94.441.15",
        "Status":"OK",
        "Type":"Logsender",
        "Version":"3.7.0",
        "Creator":"lalmarod",
        "Started":"2023-04-12T18:13:29.000-08:00",
        "Pid": 15613,
        "Port": 57222,
        "Options":{
                "-d":null,
                "-P":"57222",
                "-A":"benton.gemtalksystems.com",
                "-l":"/benton/admin/logs/logsender57222.log",
                "-s":"bentonstone"
        },
        "LogName":"/benton/admin/logs/logsender57222.log",
        "Sysconf":null,
        "Execonf":null,
        "GEMSTONE":"/benton/admin/GS6437",
        "Exe":"/benton/admin/GS6437/sys/gem",
    "Logreceivers":[   {
        "StatusInMasterCache":"true",
        "PeerIp":"10.94.441.127",
        "PeerHost":"fossa.gemtalksystems.com",
        "ProcessId": 3637081,
        "ReplayedCommit":{"File": 4,"Record": 5235},
        "ReplayedCheckpoint":{"File": 4,"Record": 5235}
  },
        {
        "StatusInMasterCache":"false",
        "PeerIp":"10.94.441.104",
        "PeerHost":"santiam.gemtalksystems.com",
        "ProcessId": 791089,
        "ReplayedCommit":{"File": 4,"Record": 5235},
        "ReplayedCheckpoint":{"File": 4,"Record": 5235}
  },
]
```

## 2.5 Garbage Collection Changes

### markForCollection performance improvement for repositories with long object chains

When a repository has very long object chains, performance of `markForCollection` can become slow. The code was improved substantially in v3.6.5 (bug 50025). v3.7 includes additional improvements; thread sleep time has adjustments to moderately improve performance under these circumstances.

### ReclaimGem configuration parameters renamed

To accommodate shorter waits on modern hardware, several configuration parameters have been renamed and are specified in smaller units.

While application settings in earlier released are converted to the new units, it is strongly recommended to review these settings and adjust or reset them to default values. Old settings may limit system commit performance.

- **sleepTimeBetweenReclaimMs** is replaced by **sleepTimeBetweenReclaimUs**, with units of microseconds. The value for **sleepTimeBetweenReclaimMs** in the previous release has the units converted and applied to **sleepTimeBetweenReclaimUs**.
- **sleepTimeWithCrBacklogMs** is replaced by **sleepTimeWithCrBacklogUs**, with units of microseconds. The value for **sleepTimeWithCrBacklogMs** in the previous release has the units converted and applied to **sleepTimeWithCrBacklogUs**.
- **maxTransactionDuration** is replaced by **maxTransactionDurationUs**, with units of microseconds. **maxTransactionDuration** was previously in units of seconds and could be as high as 300 (5 minutes). The new parameter **maxTransactionDurationUs** has a maximum of 20000000 (20 seconds) and a default value of 100000 (0.1 second). A value defined in a previous release is converted and applied, but values over 20 will be limited to 20000000.

### Number of threads configured for AdminGem

The configuration file value for STN_NUM_GC_RECLAIM_SESSIONS is now used to ensure that the AdminGem uses a number of threads appropriate for the repository configuration to perform write set union sweep and epoch.

In 3.7, when an epoch or write set union sweep is run, the AdminGem uses the Stone's computed value for STN_NUM_GC_RECLAIM_SESSIONS. The larger of this value or the AdminGem's setting for #epochGcMaxThreads or #sweepWsUnionMaxThreads, is used to determine the number of AdminGem threads used to perform the operation. The number of threads used is printed in the AdminGem's log entry for the operation.

### Renamed method to configure number of reclaim threads

A new method has been added:

```
System class >> changeNumberOfReclaimThreads:
```

Which has identical behavior as the existing method `System class >> changeNumberOfReclaimGemSessions:`. The older method continues to be usable and is not deprecated. The new method is preferred; the name reflects the current reclaim implementation.

## Reclaim GcGem Logging

The ReclaimGem configuration option #verboseLogging now has multiple levels. Higher levels may produce a very large amount of output for debugging errors and evaluating performance, and are not recommended for general use.

For compatibility with earlier versions of GemStone, in which only booleans were expected, true and false are accepted: false = 0 and true = 1.

These are the definitions for the levels:

0 - only summary information every 5 minutes in slow, 15 minutes in a fast
1 - log sigAbort, gcGemAlert, gcHwPage, etc
2 - numDead from stone, numDead processed per thread
3 - summary: commitCount, stayInTrans, crPage, numLive, numDead, numReclPages
4 - abort, start phases
5 - conflict set
6 - num live and dead found per page
7 - summary of info added to ot slot
8 - deltas in rootPage
9 - push live or dead

The default ReclaimGem logging (with level 0) is now on one line, making the log easier to read. Note at this level, summaries of reclaim operations are printed every 15 minutes.

## Determining sessions that have not voted

The following method has been added:

```
System class >> notVotedSessionsReport
```

Returns a String describing sessions that have not voted since the last epochGc or markForCollection.

## 2.6 Changes in Configuration File Handling

GemStone has a number of ways that executable and system configuration files can be located and specified, and the ways **gemnetobject** scripts can be specified and located. Some details were missing or unclear in the documentation, and a number of inconsistent or incorrect behaviors have been adjusted.

The number of options and combinatorics could make it difficult to determine why a specific configuration file was selected and used. The startup output and log files now include further details on the environment variables and arguments that affect this.

To bypass the sometimes confusing defaults, the **-E** argument has been added to a number of utilities. This is the equivalent of **-e**, but disables the default search locations.

The documentation in the *System Administration Guide*, Appendix A, has been extensively updated to clarify the specification options, precedence, and default file names for executable and system configuration files.

### Added -E argument to simplify configuration file specification

**startstone**, **startnetldi**, **topaz**, and **gemnetobject** now include the **-E** argument, which specifies a configuration file (not a directory).

When the **-E** argument is specified, use of the GEMSTONE_EXE_CONF and GEMSTONE_SYS_CONF environment variables is disabled, and the process does not search for default-named files (such as system.conf, *hostname*.conf, *stonename*.conf, and gem.conf) in any of the usual default locations.

When using the **-E** argument, the internal environment variable GEMSTONE_DATA_DIR is set to the directory in which the **-E** argument file is located. This can be used along with the `$GEMSTONE/bin/gemstone_data.conf` configuration file to setup a GemStone installation. This location is also used for GemStone system log files.

### Configuration file with GEMSTONE_DATA_DIR

An additional template configuration file, `gemstone_data.conf`, is now provided in the distribution. This specifies a default location for the extents and transaction logs relative to a configuration file path, rather than using $GEMSTONE/data. This is intended to facilitate RPM installations, in which the GemStone distribution is installed in a shared, read-only location and thus using $GEMSTONE/data for application files is not appropriate.

`$GEMSTONE/bin/gemstone_data.conf` specifies the location for extents and transaction logs using the internal environment variable GEMSTONE_DATA_DIR. This variable is not set directly by the user. Rather, when using the **-E** argument to **startstone** or other utilities, GEMSTONE_DATA_DIR is defined as the directory in which this configuration file is located.

## 2.7 Other Changes related to Administrative Operations

### Cache Warming

#### Cache warming during upgrade

During the upgrade process, after the stone has started but before **upgradeImage** has completed, logins may fail with a bad Gem version error, which prevents cache warming from happening. Since cache warming may result in faster upgrade, it is now allowed for cachewarmers to login before **upgradeImage** has completed.

#### Deprecated cache warming configuration parameters removed

The deprecated configuration parameters STN_CACHE_WARMER and STN_CACHE_WARMER_SESSIONS have been removed in v3.7. To run cache warming automatically on stone startup, you must use the configuration parameter STN_CACHE_WARMER_ARGS.

### SuperDoit scripts using GemStone base extent

SuperDoit solo scripts by default use a Rowanized extent (`$GEMSTONE/bin/extent0.rowan.dbf`), although it was possible to configure a script to use the base gemstone extent (`$GEMSTONE/bin/extent0.dbf`). In most cases, script behavior was identical between the base image and Rowan.

Now, the solo script may invoke a new support script, `$GEMSTONE/bin/superdoit_baseimage_solo`, that uses the base image, to more easily configuration a script for situations in which a Rowan image is not suitable.

See $GEMSTONE/examples/superDoit for examples:

- `versionReport.gs_solo` has been added, using the base image.
- `template.gs_solo` has been added, as a template for scripts that use the base GemStone image.
- `template.solo` has been removed; replaced by `template.rw_solo`, which uses the Rowan image.

### Changes in abort behavior in solo session

A solo session (a GemStone login that does not connect to a Stone process) has no transactional behavior nor ability to make persistent changes. `System class >> commitTransaction` has always been disallowed in solo sessions.

When converting existing scripts to solo, calls to `System class >> abortTransaction` or `System class >> beginTransaction` that had been present within the scripts previously, could result in losing changes to persistent objects that cannot be committed in a solo session (such as adding classes to UserGlobals).

To make code conversion more reliable, `abortTransaction` now signals an error if invoked in a solo session where changes to persistent objects would be lost. This allows you to find and remove the abort.

A new method, `System class >> soloAbort`, can be used within a solo session to perform a logical abort.

`System class >> beginTransaction` is also disallowed, if in a solo session and changes to persistent objects would be lost.

### Changes in session information for GcLock

Up to five sessions can hold the GcLock. The method `System class >> sessionIdHoldingGcLock` and the statistic **SessionWithGcLock** reported only the last session to get a lock, and was cleared whenever a session released its lock, even if other sessions held a lock.

`sessionIdHoldingGcLock` remains, but is deprecated.

The following methods have been added:

`System class >> sessionsHoldingGcLock`
Returns an Array of sessionIds of sessions holding a garbage collection or repository scan lock.

`System class >> gcLocksCount`
Returns a SmallInteger which is the number of sessions holding a garbage collection or repository scan lock, or 0 if the lock is free.

`System class >> gcLocksReport`
Returns a string report on the GcLocks.

`System class >> descriptionOfSession:` now includes an additional field providing the session lock held by the given session:

25. gcLockKind. 0 or the type of gcLock or repository scan lock held by the session.

The associated cache statistics **SessionWithGcLock**, which reported the session holding the lock, is replaced by **SessionsWithGcLock**, which reports the number of sessions holding the lock. **GcLockKind** is now a per-session stat, describing the kind of lock that this session is holding.

## Methods added to determine recursive size of an object tree

The following methods have been added to support repository analysis.

`Object >> recursiveSize`
Returns an Array of the form { *size* . *numberOfObjects* } where *size* is approximate physical size on disk of an object and all of the objects referenced by it and *numberOfObjects* is the count of those objects. Excludes referenced objects in the symbolList, referenced Symbols and referenced special objects.

`Object >> recursiveSizeInMemory`
Returns an Array of the form { *size* . *numberOfObjects* } where *size* is approximate physical size on disk of an object and all of the objects referenced by it and *numberOfObjects* is the count of those objects. Excludes committed objects in memory unless those objects have uncommitted changes. Excludes referenced objects in the symbolList, referenced Symbols and referenced special objects.

`Object >> recursiveSizeInMemoryReport`
Return the result of `recursiveSizeInMemory` as a readable String.

`Object >> recursiveSizeReport`
Return the result of `recursiveSize` as a readable String.

### Fast variants of findAllReferences

The following methods have been added:

`Object >> fastAllReferences`
Same as `Object >> findAllReferences`, but uses more threads.

`Object >> fastPrivateReferences`
Same as `Object >> findAllReferences`, but uses more threads, and returns references from internal nodes of large objects and Nscs.

## 2.8 Changes in Environment Variables

### Debug tracing for SSL, SSO, SSH now allowed in fast builds

Previously, using a number of debugging environment variables to print tracing information to log files was disallowed on fast (normal) builds, for security reasons. Now, sensitive information is redacted, so use of these environment variable in regular fast builds is allowed.

The affected environment variables are:

GS_DEBUG_SSL_LOG_DIR
when set to a directory, enables logging of SSL calls to a file named GsSSLDebug_pid.log

GS_DEBUG_SSO_LOG_DIR
when set to a directory, enables logging of Kerberos/single sign-on calls to a file named GsSSODebug_pid.log

GS_DEBUG_LIBSSH_DIR
when set to a directory, enables logging of SSH calls to a file named GsSSHDebug_pid.log

### Tracing LDAP

The previous debugging trace information was supported by GS_DEBUG_LDAP, which wrote information to stdout. This variable remains, but a new environment variable has been added to support writing trace information to a file.

GS_DEBUG_LDAP_DIR
when set to a directory, enables logging of LDAP calls to a file named GsLDAPDebug_pid.log

If you enable GS_DEBUG_LDAP_DIR, GS_DEBUG_LDAP has no effect.

Note that GS_DEBUG_LDAP_DIR redacts sensitive information, but GS_DEBUG_LDAP does not.

## 2.9 Changes in Errors

### Added Errors

The following exceptions have been added:

RT_ERR_STEP_St 6023
Single-step breakpoint signalled to Smalltalk.

RT_ERR_CODE_BREAKPOINT_St 6024
Method breakpoint signalled to Smalltalk.

RT_ERR_STACK_BREAKPOINT_St 6025
Stack breakpoint signalled to Smalltalk.

### Added Error and Exception Classes

The **SshSocketError** class, associated with the new feature "Support for ssh and sftp using OpenSSL", starting on page 66, was added to support the error ERR_SshSocketError 2759. ERR_SshSocketError was present, but reserved, in previous releases.

The error **GcFinalizeNotification**, which is private to GemStone internals, has been added; it is used in finalization of Ephemerons.

#### FileSystem-related additions

The following errors have been added:

| | | |
|---|---|---|
| DirectoryDoesNotExist | ERR_DirectoryDoesNotExist | 2762 |
| DirectoryExists | ERR_DirectoryExists | 2763 |
| DirectoryIsNotEmpty | ERR_DirectoryIsNotEmpty | 2764 |
| FileAttributeNotSupported | ERR_FileAttributeNotSupported | 2765 |
| FileDoesNotExistException | ERR_FileDoesNotExistException | 2766 |
| FileException | ERR_FileException | 2767 |
| FileExists | ERR_FileExists | 2768 |
| FileSystemError | ERR_FileSystemError | 2769 |
| DirectoryRequired | ERR_DirectoryRequired | 2770 |
| FsEACCES | ERR_FsEACCES | 2771 |
| FsEBADF | ERR_FsEBADF | 2772 |
| FsEEXIST | ERR_FsEEXIST | 2773 |
| FsEINTR | ERR_FsEINTR | 2774 |
| FsEINVAL | ERR_FsEINVAL | 2775 |
| FsENOENT | ERR_FsENOENT | 2776 |
| FsENOMEM | ERR_FsENOMEM | 2777 |
| FsENOSPC | ERR_FsENOSPC | 2778 |
| FsENOTDIR | ERR_FsENOTDIR | 2779 |
| FsEROFS | ERR_FsEROFS | 2780 |
| FsFileDescriptorInvalid | ERR_FsFileDescriptorInvalid | 2781 |
| FileRequired | ERR_FileRequired | 2782 |
| FilePermissionDenied | ERR_FilePermissionDenied | 2783 |

| | | |
|---|---|---|
| FsUnixError | ERR_FsUnixError | 2784 |
| FileAlreadyExistsException | ERR_FileAlreadyExistsException | 2785 |
| FsError | ERR_FsError | 2786 |
| ZnError | ERR_ZnError | 2787 |
| ZnCharacterEncodingError | ERR_ZnCharacterEncodingError | 2788 |

## Removed Errors

The following errors have been removed:

RT_ERR_TERMINATE_PROCESS 6018

RT_ERR_TRACKED_OBJS_NEEDS_INIT 2381

## Cannot return now not resumable

The **CannotReturn** error returned true for `isResumable`, although this error could not actually be resumed. (#49677)

## OS-origin error messages changed

Previously, GemStone provided it's own strings to describe errors that orginate from the OS, for example, "ENOENT, The file or directory specified cannot be found".

Now, the wording provided by the OS (on Unix) is used, which may be different; for example, "No such file or directory".

## 2503 changed from FloatingPointError to NumericError

This error number is now associated with the class NumericError.

## AbstractException >> signalToGci

The following method has been added:

`AbstractException >> signalToGci`

Receiver is signaled and it will not be trappable by any exception handler. An Error will be return to the GCI.

## Print GCI trace records on protocol error

To aid in analysis of GCI protocol errors, recent GCI call history is now automatically printed to the gem log when a GCI protocol error occurs. GCI call trace records are maintained in a wraparound buffer, and the most recent 100 records will be printed to the log.

## MFC Warning now includes possibleDeadSymbols

The warning signaled by `markForCollection` now includes the possibleDeadSymbols count, as well as live and dead object counts. This change was present in 3.6.4, but not documented.

## 2.10 Changes in Utilities

### Many utilities faster on Linux

The **startstone**, **stopstone**, **stopnetldi**, and **waitstone** utilities have been tuned and optimized for modern hardware, and require much less overhead time than in previous releases. In practice, the amount of startstone performance improvement will depend on the specific **startstone** tasks for an application.

### Utilities startstone, topaz, NetLDI and gemnet* support -E argument

These utilities now support new rules for resolving configuration files, as described in section 2.6 on page 36.

Some corner cases of behavior in the non-E handling have been cleaned up.

The documentation in the *System Administration Guide* has been extensively revised and expanded.

### Custom gemnetobject locations

It is allowed to make a copy of any of the gemnetobject scripts and edit to add your desired behavior.

Previously, it was documented that the custom gemnetobject would be found in the home directory of the current user. However, the code was incorrectly looking in the startup directory of the NetLDI. This behavior has been removed.

To use a custom gemnetobject script, you may either:

- use the full path to the custom script in the NRS login parameters.
- edit `$GEMSTONE/sys/services.dat` to translate the custom script name to its location path and file.

Note that relative paths are not allowed.

### Changes in largememorypages

#### Argument handling changes

When using a configuration file as an argument to **largememorypages**, the configuration file must have large pages enabled. If SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_POLICY is not enabled in a **-z** or **-e** argument, the **largemememorypages** output includes cache configuration information, but no large page size recommendations.

In addition, the **-p** or SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_SIZE_MB in a **-z** or **-e** argument no longer defaults; you must now specified the page size on Linux and AIX.

#### Instructions updated

The recommendations for how to configure Linux to enable use of large memory pages, which are printed as output, has been updated to recommend using a group rather than cap_set_resource. See the *Installation Guide* for v3.7 for more details.

## secure_backup_extract added

The **secure_backup_extract** utility provides a way to verify the signature of a secure backup, that can be used on secure backup files from any earlier version (the secure backup feature was added in v3.4), without requiring a GemStone environment. This utility is available on little-endian hosts only (Linux and Mac).

Usage:

```
secure_backup_extract (sig | cert) secureBackupFile outputFile
where:
   sig - extract the digital signature and write to outputFile
   cert - extract the public signing key and write to
         outputFile
   secureBackupFile - a valid GemStone secure backup
   outputFile - location to write script results; this file
         must not already exist
```

**secure_backup_extract** extracts the public signing cert or the digital signature from a secure backup file, and writes it to a new file. While $GEMSTONE is not needed, you must have openssl and perl available.

This script produces the same result as **copydbf**, using **copydbf -X** to extract the public signing cert, or **copydbf -Y** to extract the digital signature. To verify the digital signature, compare the binary files using the unix utility `cmp`.

For example, if you have a secure backup file *backup.sdbf* that was signed using GemStone's example private cert `backup_sign_2_clientkey.pem`, either of the following will extract the public signing cert:

**secure_backup_extract** `cert` *backup.sdbf signingCert.secbkext*
**copydbf** `-X` *signingCert.copydbf backup.sdbf*

This can be compared with each other, or against the signing cert used to make the backup; for example:

`cmp -l` *signingCert.secbkext* `$GEMSTONE/examples/openssl/certs/backup_sign_2_clientcert.pem`

To extract the digital signature itself:

**secure_backup_extract** `sig` *backup.sdbf digitalSig.secbkext*
**copydbf** `-Y` *digitalSig.copydbf backup.sdbf*

which can be compared for validation:

`cmp -l` *digitalSig.secbkext digitalSig.copydbf*

## Topaz changes

### Default fileformat now utf8

Regardless of the Comparison Mode of the repository, topaz's default **fileformat** is now **utf8**. This ensures files will be written to disk in UTF-8 encoding.

Files generated from GemStone will include an explicit **fileformat** command. If you are filing in files with the old 8-bit encoding, ensure the headers include `fileformat 8bit`, or execute this command prior to filein.

## Improvements to debugging using DEBUGGEM

The topaz **debuggem** command allows you to attach topaz to an executing process for debugging. v3.7 has new features that make this easier.

### System waitForDebug

The added method `System class >> waitForDebug` enables GEM_LISTEN_FOR_DEBUG; it is not necessary to use the configuration parameter, nor to invoke `listenForDebugConnection`. It then immediately waits, so another session can attach. When using this method, the **debuggem** command is the same, but there is no need to perform the **stack set** in the debugging session, as was necessary when using **topazwaitfordebug.**

### System cancelWaitForDebug

Executing the added method `System class >> cancelWaitForDebug` in the executing session cancels the effect of the waiting part of the waitForDebug method. On logout, the executing session will resume running. This is not needed if using the new **detach** topaz command.

## Added Topaz commands to support DEBUGGEM

**KILL**
In the debugging topaz session (the session created by **debuggem**), execute a kill -TERM against the executing gem or topaz -l (the session being debugged).

**DETACH**
In the debugging topaz session (the session created by **debuggem**), executes `System cancelWaitForDebug` and logs out. The GsProcess being debugged will be continued after the logout, and `System class >> waitForDebug` will return to the caller.

**RESUME**
In the debugging topaz session (the session created by **debuggem**), executes `System cancelWaitForDebug` and logs out. The GsProcess being debugged will be continued after the logout, and `System class >> waitForDebug` will return to the caller.

## SET added option ONETIMEPASSWORD

A option has been added to support the one-time password feature (see page 25).

**set onetimepassword** *onOrOff*
Determines if the password is a normal GemStone password or a special one-time password. When OFF, the default, the password field is interpreted as a normal GemStone password.When ON, the password field is interpreted as a onetime password.

### LIMIT added option LEV2OOPS

A new option has been added to the **limit** command:

**limit lev2oops** *aNum*
When **level** is set to 2 or greater, or **obj2** is used, the **lev2oops** limit controls how many oops to display of instance variable values.

## 2.11 Configuration Parameter Changes

### Removed configuration parameters

The following obsolete and nonfunctional parameters have been removed:

DBF_SCRATCH_DIR

GEM_PRIVATE_PAGE_CACHE_KB

STN_PRIVATE_PAGE_CACHE_KB

GEM_RPCGCI_TIMEOUT

The following legacy parameters have also been removed; applications should use the parameter STN_CACHE_WARMER_ARGS for automatic cache warming on startup.

STN_CACHE_WARMER

STN_CACHE_WARMER_SESSIONS

### GEM_NATIVE_CODE_ENABLED

Previously, a setting of 0 disabled native code, 1 enabled native code, and 2 enabled native code with inlining some math primitives.

Now, there are only two settings: 0 to disable native code, and 2 to enable native code. A setting of 1 is reserved for future use, but is legal to use and interpreted as a 2 to enable native code.

### SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_SIZE_MB

The default setting for this parameter was previously 2 to specify 2MB large pages, although a value of 0 was intended to specify the linux default value. Now, with no setting or a setting of 0, the system determine a default by looking at:

- `/sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages`
- `/sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages`
- the value of Hugepagesize in `/proc/meminfo`

If there are 1GB pages configured, the value defaults to 1GB; otherwise, if there are 2MB pages configured, the value defaults to 2MB; otherwise, it will use the Linux default size in /proc/meminfo.

### SHR_PAGE_CACHE_SIZE_KB default increased

The default for SHR_PAGE_CACHE_SIZE_KB, when a specific value is not defined in a configuration file used by the Stone, has increased from 75MB to 200MB.

## STN_GEM_TIMEOUT

If the value of this parameter is 0, the login timeout used was previously 5 minutes, and is now 1 minute.

## STN_NUM_GC_RECLAIM_SESSIONS now applies to Admin Gem

The number of sessions used by the Admin GcGem now may be determined using the setting for reclaim sessions, which is calculated based on the setting for STN_MAX_GC_RECLAIM_SESSIONS. See “Number of threads configured for AdminGem” on page 34.

## 2.12 Changes in Statmonitor and Cache Statistics

The information written on cache statistics has been extensively revised and reorganized, and new Process Types added to reflect the different responsibilities of different GemStone processes or process threads. In particular, the general Pgsvr process type has been divided up into the different specific page servers (cache, free frame, AIO, etc.), and the PageManager has been divided from the Stone. A number of obsolete statistics have been removed, and new statistics have been added.

#### Performance improvement for Shared Page Cache Monitor

Statistics collection in the Shared Page Cache Monitor has been adjusted so expensive per-frame statistics are collected less frequently, not more than once per second, to reduce overhead for large systems.

### Improved lz4 compression and new VSD required

Compressing statmonitor data with lz4 now has improved compression (between 3% and 25%) by using a C dictionary; this dictionary is in the distribution as `$GEMSTONE/bin/lz4.statmon.dict.bin`, and is used by VSD to read lz4-compressed statmonitor data files in VSD v5.6 and later.

Statmonitor files generated by 3.7 and compressed using lz4 can only be read using VSD v5.6 or later. VSD 5.6 can read lz4-compressed statmonitor files generated by earlier versions of GemStone, and older versions of VSD can read statmonitor files generated by GemStone v3.7 that are not compressed or that are compressed using gz.

To uncompress an lz4-compressed statmonitor data file on the command line, you must use an additional -D $GEMSTONE/bin/lz4.statmon.dict.bin argument; for example:

```
os$ lz4 -D $GEMSTONE/bin/lz4.statmon.dict.bin stats.out.lz4
```

### Statmonitor -z -Z arguments now accept compression level

The **statmonitor -z** (gzip) and **-Z** (lz4) compression arguments now accept an optional argument to specify the compression level. Higher levels provide greater compression, at the expense of performance.

**-z** accepts 1...9; the default, if no level is included, is 6.

**-Z** accepts 1...12; the default, if no level is included, is 3.

### Statistics Types reorganized

Several Process Types have been split up. For statistics that applied to the previous general process type, these are now associated only with the relevant process types. For example., the PageMgr* stats that were previously of type Stn, since the PageManager is a thread in the Stone, are now specifically of type PageMgrThr.

#### Pgsvr split

The general type Pgsvr no longer exists. It is replaced by:

**StnAioThr** (AIO PageServer threads in Stone)
**FreeFrameThr** (Free Frame PageServer threads in Stone)
**CachePgsvr** (Cache PageServer)

**GemPgsvrMain**, **GemPgsvrThr** (Gem PageServer)

Statistics that were previously of type Pgsvr will now apply to one or more of these new types.

### Stone, Page Manager, and Stone's Restore Threads split

The PageManager has been a thread in the Stone for some number of releases, but statistics for PageManager were previously recorded under the same statistics type as the Stone. Now, the PageManager has its own Type, **PageMgrThr**.

Note that since the PageManager is a thread, not a process, host process stats are not collected for the PageManager. To see host process stats, you must look at the Stone's statistics.

The restore threads used during restoreFromBackup have also been made a new process type, **RestoreThr**.

### Gems split

For threads within the Gem, the **GemThr** type has been added, to distinguish from the **Gem** type. Host system stats are not collected for **GemThr**.

A new method has been added to list statistics that apply to Gem threads:

`System class >> cacheStatisticsDescriptionForGemThreads`
Returns an Array of Strings describing cache statistics applicable to threads started by gems for certain repository-wide operations such as markForCollection, full backup, etc.

### Added API for PageManager statistics

`System class >> cacheStatisticsDescriptionForPageManager`
Returns an Array of Strings describing cache statistics applicable to the stone's page manager thread.

`System class >> pageManagerCacheStatistics`
Return the cache statistics for the page manager thread if this session is located on the same host as the stone process. Only cache statistics applicable to the page manager thread are returned.

`System class >> pageManagerCacheStatisticWithName: aString`
Return the value of the cache stat with the given name, which must match an element of the Array returned by the #cacheStatisticsDescription method applicable to the page manager thread. The UserTime and SysTime statistics cannot be accessed using this method.

`System class >> pageManagerProcessSlot`
Answer the cache slot for the page manager thread or -1 if the slot was not found. A page manager exists only on the shared page cache used by the stone process.

## Changes in memory statistics on Linux

The existing statistics to capture memory use on Linux did not reliably capture important information, and in many cases were misleading.

Now, on Linux, **statmonitor** collects memory use related statistics from `/proc/<pid>/smaps` instead of `/proc/<pid>/maps`.

There is some performance impact, since it is more expensive to read `smaps` than `maps`, particularly on large systems with many processes and short sampling intervals. A new thread in statmonitor is dedicated to reading `smaps`; this thread is designed to consume a upper maximum of 25% of 1 core, although it will normally take less; much less on smaller systems.

### Per-process statistics

The following per-process statistics have been removed:

**ImageKBytes**
**MaxImageSize**
**MaxRSS**
**RSSData**
**RSSKBytes**
**RSSLib**
**RSSStack**
**RSSText**

The following are the new per-process statistics. Note that these are only collected for the main thread within the Gem.

**PrivateClean** (All)
The amount of memory in kilobytes mapped to the process which has not been modified since it was mapped.

**PrivateDirty** (All)
The amount of memory in kilobytes mapped to the process which has been modified since it was mapped.

**PSSSize** (All)
The proportional set size of the process in kilobytes. PSS includes memory counted in PrivateClean and PrivateDirty as well as a portion of the memory shared by this process and with one or more other processes.

**Swap** (All)
The amount of memory in kilobytes used by this process which has been swapped out to disk.

**SwapPss** (All)
The amount of memory in kilobytes swapped out to disk. SwapPss includes memory counted in Swap as well as a portion of the memory swapped out which is shared by this process and with one or more other processes.

### Add statistic for number of memory mapped regions in use

**MemMapRegions** (CachePgsvr Gem GemPgsvrMain Linux_Process Netldi Shrpc Statmon Stn)
The number of memory mapped regions the process is using.

## Added statistics for smaps thread

The following statistics allow monitoring performance of the new statmonitor smaps thread itself.

**SmapsCollectCount** (Linux System)
Number of times the statmonitor smaps thread has collected smaps statistics for all processes it is monitoring.

**SmapsLastActualSleepTime** (Linux System)
The number of real milliseconds the statmonitor smaps thread actually slept in its most recent sleep.

**SmapsLastReqSleepTime** (Linux System)
The number of real milliseconds the statmonitor smaps thread requested to sleep in its most recent sleep.

**SmapsLastTimeCollectingStats** (Linux System)
The number of real microseconds the statmonitor smaps thread spent collecting smaps statistics for all monitored processes for its most recent sample.

**SmapsPidsMonitoredCount** (Linux System)
The number of processes the statmonitor smaps thread is currently monitoring.

**SmapsProcFileFailedReads** (Linux System)
Number of times the statmonitor smaps thread received an error when attempting to read the smaps file for a process. Usually indicates the process has exited.

**SmapsSharedTableBuckets** (Linux System)
Number of collision buckets present in the smaps shared process id table.

**SmapsSharedTableSize** (Linux System)
Number of keys present in the smaps shared process id table.

## New restrictions for per-process Statistics for configurations in root mode with s bit

Access to `/proc/`*pid*`/smaps` is restricted. Statmonitor and gem primitives which collect per-process statistics cannot collect smaps statistics for processes owned by other users. Attempting to collect the smaps statistics does not raise an error, but the values for those specific statistics will always be zero (all other statistics will show up normally for other users).

When running with the NetLDI in guest mode with captive account, each gem process is owned by this account, and thus each Gem has programmatic access to the statistics for other users. As long as statmonitor is executed as the captive account user, and this account is the same account as started the Stone, statmonitor can collect these memory statistics for all processes.

When running the Stone and NetLDI owned by root with the s bit set, Gem processes are owned by the account logging in. In this case, statmonitor requires additional configuration to be able to read these memory statistics.

- You may give statmonitor the cap_sys_ptrace capability:

```
os$ setcap cap_sys_ptrace=pe $GEMSTONE/bin/statmonitor
```

- Alternatively, statmonitor can be run as root with s bit set:

```
os$ cd $GEMSTONE/sys
os$ chown root $GEMSTONE/bin/statmonitor
os$chmod u+s $GEMSTONE/bin/statmonitor
```

### Access from other Gems with NetLDI in authenticated mode

Cache statistics are most commonly recorded by statmonitor. However, you may collect cache statistics programmatically; including host statistics, using methods such as `System hostStatisticsForProcess:`.

With the NetLDI owned by root with the s bit set (authenticated mode), Gems processes run as the unix userid that is logging in; and Gems will not be able to read the smaps memory statistics for Gems that are logged as a different unix user.

For security reasons, it is not recommended to give all Gem processes the cap_sys_ptrace capability; and setting this capability prevents gdb attaching, so pstack and kill -USR cannot write C stack traces. If a Gem must programmatically monitor the memory statistics for other Gem processes, you should configure the monitor Gem to run with a different gem executable than standard Gems in your environment, and give this gem executable the required capability.

1. Make a copy of `$GEMSTONE/sys/gem` for an RPC Gem, or `$GEMSTONE/bin/topaz` if this process will run as linked topaz. For example:

   ```
   os$ cp $GEMSTONE/sys/gem $GEMSTONE/sys/gemTrace
   os$ cp $GEMSTONE/bin/topaz $GEMSTONE/bin/topazTrace
   ```

2. Give this copy the cap_sys_ptrace capability.

   ```
   os$ setcap cap_sys_ptrace=pe $GEMSTONE/bin/topazTrace
   os$ setcap cap_sys_ptrace=pe $GEMSTONE/sys/gemTrace
   ```

3. For an RPC gem, create a custom gemnetobject, and use this for login.

   ```
   os$ cp $GEMSTONE/sys/gemnetobject
      $GEMSTONE/sys/gemnetobject_trace
   ```

   Edit your custom gemnetobject, for example `gemnetobject_trace`, to set the gemname that invokes this executable, by modifying this line:

   ```
   gemname="gemTrace"
   ```

   Login using your custom gemnetobject in the login parameters:

   ```
   topaz> set gemnetid gemnetobject_trace
   ```

4. If you are running NetLDI with the **startnetldi -n** option, you must also add an entry to the `services.dat` file. See the*System Administration Guide*, Appendix A, for more details.

## Other changes in existing statistics

### Details for GarbageCollectionState and GcHighWaterPage

These statistics report numeric values that represent internal state. These have been verified and the statistics definition updated for 3.7.

**GarbageCollectionState (Gem)**

Indicates the phase of a garbage collection task or other multi-threaded operation. Values are defined as follows:

0 - Inactive
1 - Warm Dependency Map
2 - Scan Object Table
3 - Scan Data Pages
4 - Scan Shadowed (scavengable) Data Pages
5 - Scan Remaining Data Pages
6 - Backup Scavengable Pages
7 - Backup Remaining Pages
8 - Restore Data Pages
9 - Rescan Shadowed Pages (for listRefToInstOfClasses)
10 - Rescan Remaining Pages
11 - Audit Scavengable Pages
12 - Audit Remaining Pages
13 - Audit Rescan Object Table (rescan to find references to non existent objects)
14 - Audit Rescan Data
15 - All Symbols Sweep
16 - Mark Sweep
17 - Write Set Union Sweep
18 - Find Connected Objects Sweep
19 - Migrate Merge Deltas

**GcLockKind (Gem)**

Indicates the state of the garbage collection lock and why it is being held:

0 - Free
1 - MarkForCollection
2 - FindDisconnectedObjects
3 - Epoch GC
4 - Write-set union sweep
5 - Reclaim All
6 - Backup
7 - Repository Scan (listInstances, listReferences, etc)
8 - Conversion

### ReposAtMaxSize changed

Previously, **ReposAtMaxSize** was a boolean. Now, it is an integer indicating the current state of the extents:

0 - extents are not at maximum configured size and not in a disk full state.
1 - all extents have file system full state.

2 - all extents are currently at their configured size limit.

### Changes in Session*WithGcLock and GcLockKind

As described on page 38, the GcLock can be held by up to five sessions, which makes existing protocol misleading.

**SessionWithGcLock** has been removed, replaced by the following statistic:

**SessionsWithGcLock**
Number of sessions holding a GcLock or repository Scan lock

The Stone's statistics no longer include **GcLockKind**. **GcLockKind** is now a per-session stat, describing the kind of lock that this session is holding.

## Removed Statistics

The following statistics have been removed (in addition to Linux per-process stats listed above, and **SessionWithGcLock**):

**AbortInProgress**
**ClientAbortInProgress**
**CodeGenFullCount**
**ObjectMemoryGrowCount**
**PreemptedBitmapPages**
**PreemptedCommitRecordPages**
**PreemptedDataPages**
**PreemptedObjectTablePages**
**PreemptedOtherPages**
**SpinLockNewSymSleepCount**
**TimeSleepingMs**
**TrackedSetSize**
**TteCrPageFreeCount**

## Added Statistics

### "Time Since" statistics

The following statistics provide the time since a particular event occurred; this avoids the need to manually determine the delta between a given timestamp and the event timestamp. 0 means the event just occurred, -1 means the event has not occurred in the lifetime of the process.

**SecondsSinceAbort** (Gem Stn)
Stone: number of seconds since any session has aborted. Gem: number of seconds since this session has aborted.

**SecondsSinceCheckpoint** (Stn)
Number of seconds since a checkpoint was started.

**SecondsSinceCommit** (Gem Stn)
Stone: number of seconds since any session has committed. Gem: number of seconds since this session has committed.

**SecondsSinceCrDisposal** (Stn)
Number of seconds since a commit record was disposed.

**SecondsSinceExtentGrow** (Stn)
Number of seconds since any extent was grown.

**SecondsSinceFailedCommit** (Gem Stn)
Stone: number of seconds since any session attempted a commit which ultimately failed.
Gem: number of seconds since this session attempted a commit which ultimately failed.

**SecondsSinceLogin** (Gem Stn)
Stone: number of seconds since any session logged in.
Gem: number of seconds since this session logged in.

**SecondsSinceLogout** (Stn)
Number of seconds since any session logged out.

**SecondsSinceLowFreespace** (Stn)
Number of seconds since repository low free space was detected.

**SecondsSinceNewTranlog** (Stn)
Number of seconds since a new tranlog was started.

**SecondsSinceReclaim** (Stn)
Number of seconds since a reclaim gem committed.

**SecondsSinceReclaimDead** (Stn)
Number of seconds since a reclaim gem committed and reclaimed object identifiers.

**SecondsSinceSigAbort** (Gem)
Number of seconds since the session detected a SigAbort.

**SecondsSinceSigLostOt** (Gem)
Number of seconds since the session detected a SigLostOtRoot.

**SecondsSinceStartLongPrim** (Gem)
Number of seconds since the session started a long primitive operation.

**SecondsSinceStartStone** (Stn)
Number of seconds since the stone has been available for logins.

**SecondsSinceTranlogFull** (Stn)
Number of seconds since the a tranlog full condition was detected.

### Other Added Cache Statistics

**BitmapPagesPreempted** (CachePgsvr FreeFrameThr PageMgrThr)
Number of bitmap pages removed from the cache by the process.

**BitmapPageWrites** (StnAioThr)
Number of bitmap pages written to disk by the process.

**CacheIdle** (Shrpc)
A boolean indicating if the shared page cache is idle.

**CachePgsvrRemoveFrameId** (CachePgsvr)
The frameId that the cache page server is attempting to recycle.

**CachePgsvrRemovePageId** (CachePgsvr)
The pageId that the cache page server is attempting to remove from the remote shared page cache.

**CacheRegionNumFrames** (FreeFrameThr StnAioThr)
Number of frames in the cache region for the process.

**CommitRecordPagesPreempted** (CachePgsvr FreeFrameThr PageMgrThr)
Number of commit record pages removed from the cache by the process.

**CommitRecordPageWrites** (StnAioThr)
Number of commit record pages written to disk by the process.

**DataPagesPreempted** (CachePgsvr FreeFrameThr PageMgrThr)
Number of data pages removed from the cache by the process.

**DataPageWrites** (StnAioThr)
Number of data pages written to disk by the process.

**FfiGcHeapBytes** (Gem)
Bytes allocated by `CByteArray class >> gcMalloc:` and other C data allocations for instances of GsSocket, etc, and not yet freed by in-memory GC.

**FfiHeapBytes** (Gem)
Cumulative bytes allocated by `CByteArray class >> malloc:`, that the VM GC will not free, not decremented when application FFI code calls free.

**FreeFrameListOkLimit** (FreeFrameThr StnAioThr)
The ok free frame limit for the process.

**FreeFrameLowerLimit** (FreeFrameThr StnAioThr)
The lower free frame limit for the process. When the number of free frames is below this value, the process will aggressively add frames to the free list (free frame threads) or write dirty pages to disk (Aio threads).

**FreeFrameUpperLimit** (FreeFrameThr StnAioThr)
The upper free frame limit for the process.

**GemThreadsInCacheCount** (Shrpc)
The number secondary gem threads currently attached to the cache.

**LowFreespaceCount** (Stn)
Number of times stone detected a low free space condition.

**NumInSecurityDataQueue** (Stn)
Number of sessions waiting for SymbolGem to update security data of their UserProfile.

**ObjectTablePagesPreempted** (CachePgsvr FreeFrameThr PageMgrThr)
Number of object table pages removed from the cache by the process.

**ObjectTablePageWrites** (StnAioThr)
Number of object table pages written to disk by the process.

**OnetimePasswordsActive** (Stn)
Number of onetime passwords in the stone's hash map.

**OnetimePasswordsCreated** (Gem)
Number of onetime passwords this session has created.

**OnetimePasswordsExpired** (Stn)
Number of expired onetime passwords automatically removed by stone.

**OnetimePasswordsNotValidated** (Stn)
Number of failed onetime password validations.

**OnetimePasswordsTotal** (Stn)
Number of onetime passwords ever created by any session since the stone was started.

**OnetimePasswordsValidated** (Stn)
Number of successful onetime password validations.

**OnetimePasswordUsed** (Gem)
A boolean with value 1 if the session used a onetime password to login, otherwise 0.

**OtherPagesPreempted** (CachePgsvr FreeFrameThr PageMgrThr)
Number of pages removed from the cache by the process that were not object table, data, commit record, or bitmap pages.

**OtherPageWrites** (StnAioThr)
The number of pages written by the process that were not object table, data, commit record or bitmap pages since the process was started.

**RecovNumPendingCheckpoints** (Stn)
Number of checkpoints pending replay from the tranlog during recovery.

**RecovSkippedCheckpointCount** (Stn)
Number of checkpoints skipped during recovery.

**RecovTimeWaitingForCheckpoints** (Stn)
Total amount of real milliseconds the stone recovery thread spent waiting for checkpoints.

**ScanLimitNumFrames** (FreeFrameThr StnAioThr)
Maximum number of cache frames scanned by the process before sleeping.

**TimeInCheckpoint** (Stn)
Cumulative number of real seconds the stone has spent writing checkpoints.

**TimeInLowFreespace** (Stn)
Cumulative number of real seconds the stone has spent in low free space mode.

**TranlogFullCount** (Stn)
Number of times stone detected a tranlog full condition.

**TteCrPageInUse** (Stn)
Number of commit record page entries in use in stone's internal commit record cache.

Chapter

# 3 Changes in GemStone Smalltalk

This chapter describes changes and new features important for programmers using GemStone Smalltalk, including:

## 3.1 Changes in Classes, Compilation, and Code Management

### ClassOrganizer >> newExcludingGlobals

This method allows you to perform ClassOrganizer queries that do not return results exclusive to Globals. This can be used, for example, to find application methods that reference GsHostProcess or JsonParser classes (which may need to be recompiled; see the *Installation Guide* upgrade instructions), without including kernel methods that do not need recompiling.

`ClassOrganizer >> newExcludingGlobals` replaces the private method `_newExcludingGlobals`, which was available in some recent releases. Note that `_newExcludingGlobals` did not reliably exclude methods on classes in Globals.

## GsNMethod >> recompileFromSource

This method has been added for convenient recompilation of a method from source code, such as the recompile needed for GsHostProcess and JsonParser. Existing recompile methods were designed for recompiling after bytecode changes; since they relied on the existing literal references, they did not update references to the obsolete class.

Note that if the symbolList of the user that is executing the recompile resolves literal names to different objects than the original compile, this may result in unexpected changes in behavior. Depending on the complexity of your application's use of users' symbolLists, you may wish to examine the source code before recompiling.

## Added method Behavior >> compileMethod:category:environmentId:

The following method has been added, allowing you to compile a method using the current symbol list:

`Behavior >> compileMethod:` *sourceString* `category:` *aCategoryString*
`environmentId:` *environmentId*

## #logCreation and logging class creation

The option #logCreation can be included in the options: of a class creation message, which results in the class creation being logged (using `GsFile gciLogServer:`). The primary purpose was for tracking class creation during build/upgrade filein. This option is not persistent and not printed (reported by Class >> definition), and thus was not retained if a class was filed out and in.

Now, you may set the key #GsClass_logCreation in SessionTemps, to force logging of all class creations for the lifetime of the session, using an expression such as:

```
SessionTemps current at: #GsClass_logCreation put: true
```

In the absence of this setting, the behavior is unchanged.

## Improvements to filein using GsFileIn

GemStone supports filein of topaz-format source code using the class GsFileIn. This provides a functionality similar to topaz `input`, but permits only a subset of the full set of topaz commands. GsFileIn allows you to filein GemStone code programmatically, entirely from within GemStone code.

GsFileIn has been extensively revised. With v3.7, GsFileIn now supports source files that include extended characters encoded as Utf8; and performance has been greatly improved.

While instances of GsFileIn are not normally persisted, on upgrade, any existing instances become instances of ObsoleteGsFileIn.

The `GsFileIn Class >> from*:` API has been streamlined. While `fromServerPath:` and `fromClientPath:` may be used, the preferred methods (methods whose names are more accurate) are `fromGemHostPath:` and `fromGciHostPath:`, respectively. The method `GsFileIn >> fromStream:` has been added, to allow filein from streams.

For more details on GsFileIn, see the class comment, and methods in the image.

### SystemUser SymbolList Changes

In v3.6.x, SystemUser's SymbolList (visible when logged in as SystemUser), included a number of additional SymbolDictionaries, including GsCompilerClasses, ObsoleteClasses, RowanKernel, and in some versions GemStone_Portable_Streams and GemStone_Legacy_Streams. This facilitated work, or were artifacts of, the transition to Rowan-based code management.

For 3.7, all these dictionaries have been removed; the classes are located in subdictionaries within Globals, as needed.

## 3.2 Customer-defined Special Classes

GemStone now provides 16 predefined, customizable Special classes. Each can encode 56 bits of data.

These classes are named Special56bit*N*, and include Special56bit0 through Special56bit15.

To create an application Special class, you will modify one of the Special56bit*N* classes. You may change the superclass, rename the class, add methods, and if desired, change its security policy to allow other users to modify the class.

It is legal to create an association to this class from the same or another SymbolDictionary, but you should leave the association with the original name in the Globals dictionary, for upgrade.

The Special classes include a number of methods that invoke primitives. These and other support methods are in a category 'Base Methods' and should not be modified. You may add all necessary methods to other categories within the class to support the specific functions you need.

For example, to setup a special class in a different SymbolDictionary, with the name *classNameString*, and allow users other than SystemUser to edit methods on the class:

1. Create an association in the target SymbolDictionary to the target name *classNameString*.
2. As SystemUser, implement `Special56bit`*N*` class >> name` to return *classNameString*, and execute *specialClass* >> `changeNameTo:` *classNameString*.
3. To allow developers other than SystemUser to edit this class, change its securityPolicy using:

   *specialClass* `objectSecurityPolicy:` *anObjectSecurityPolicy*
4. It is allowed to change the superclass of a Special class, since the required primitive methods are defined in each class, rather than inherited. The new superclass (and each superclass up through Object) must be created with the #selfCanBeSpecial class creation option.

5. Bit-encode the specific data for your special into the 56 available bits, invoking the Class method `value:` to create an instance, and the instance method `value` to read and decode into your specific data type/s.

Note that care must be taken for fileout and filein, since the base class methods are a mix of GemStone methods and your own application methods. There are a number of ways to manage this, depending on your source management tools. The Money example shows a workaround using a customization of base image fileout.

### Money example

An example of using this feature is the example at:

```
$GEMSTONE/examples/smalltalk/Money.gs
```

This example demonstrates using Special56bit0 to implement a Special Money class encoding an amount and an integer currency ID, to support US and Canadian Dollars, Euro and Yen. Note that this is a trivial example class, and does not provide the necessary currency support for a real-world application.

This example must be filed in as SystemUser. It creates the Money class in the Published dictionary. Methods on the resulting Money class can be edited by a user with access to the DataCuratorObjectSecurityPolicy.

## 3.3 Multithreaded Instance Migration

Faster instance migration is now possible using the new migration API. This allows you to specify up to 2000 classes; for each of these classes, every instance in the repository is migrated in a single operation within C code. Only simple migration operations on non-collection classes is supported. The primitive that performs this operation commits if successful, so this should be done when no other users would be committing, to avoid concurrency conflicts.

Multithreaded migration requires the MigrateObjects privilege; see “MigrateObjects privilege” on page 29.

This initial version of multi-threaded migration supports a limited set of migration pathways.

Supported:

- mapping instance variable values to differently named instance variables
- set values to nil
- preserve or remove dynamic instance variables

Not supported:

- migrating indexable or NSC objects (that is, collections; including those with instance variables)
- modifications to specific dynamic instance variables
- mapping between dynamic and named instance variables

For migrations that are not supported by the multi-threaded migration function, you can continue to use the object based message-send migration options.

Multi-threaded migration is configured using the class InstVarMappingArray, which was previously a support class for instance migration that was not directly used. An instance of this class is the input to the multi-threaded migration operations.

To ensure that your migration produces the results you expect and there are no unforeseen issues, it is recommended that prior to the actual migration, you test the migration with a more limited number of test objects. `Repository >> testMigrateMt:with:` allows you to perform the same migration as `Repository >> migrateMt:`, on a limited set of objects, and does not commit; after running this method commits are disallowed, so you must abort after running this test.

The following methods have been added:

`Repository >> migrateMt:` *arrayOfMappings*

Performs a multi threaded scan of the repository migrating instances of the classes found in the *arrayOfMappings* to their corresponding new classes. Up to 2000 classes can be migrated in a single operation. The classes must be committed. An error is generated if there are any modified persistent objects in the temporary object memory at the time it is executed.

The method will commit all of the migrated objects before it returns. It is possible that the commit may fail due to concurrency conflicts, so to avoid that the operation should be performed while there is no other activity on the system.

The *arrayOfMappings* is Array of instances of InstVarMappingArray which defines the classes that are to be migrated. For example:

```
SystemRepository migrateMt:
   {(InstVarMappingArray mappingFrom: oldClass to: newClass)}
```

Before executing this method you may wish to execute `testMigrageMt:with:` to validate that the migration does what is intended.

`Repository >> fastMigrateMt:` *arrayOfMappings*

Like `Repository >> migrateMt:`, but performs the migration aggressively, using more system resources, in order to complete more quickly.

`Repository >> testMigrateMt:` *arrayOfMappings* `with:` *testObjs*

This method is similar to `migrateMt:` and can be used to test the `migrateMt:` operation on a limited number of objects before performing a full repository scan and migrating all objects.

It performs a multi threaded scan of only the objects listed in the test array, migrating the objects in the same way that `migrateMt:` would, but it does NOT commit them. The session is left in a state with commits disallowed. Once the objects have been inspected and have been verified that the migration performed as expected the user should abort the current transaction.

## InstVarMappingArray

The existing InstVarMappingArray class has been modified and updated, and an instance can be created and customized directly for use by multi-threaded migration. An instance of InstVarMapping is defined for a single specific class, and specifies migratation of instance variables defined on that class and inherited instance variables in the same way.

Note that ordinary object based migration continues to use instances of InstVarMappingArray without specific customization.

The following methods have been added:

`InstVarMappingArray >> mapInstVarNamed:` *oldName* `to:` *newName*
The value at the instance variable named *oldName* in the old class should be migrated to the instance variable named *newName* in the new class.

`InstVarMappingArray >> mapInstVarToNil:` *newName*
The migration should not retain the value of any instance variable *newName* present in the instance of the old class. After migration; the object will have a nil value at this instance variable.

The origin and target classes are now in instance variables, and methods have been added to access these:

`InstVarMappingArray >> oldClass`

`InstVarMappingArray >> newClass`

### Example

For example, to create a mapping from `OrigClass` to `NewClass` that moves the value of the instance variable `oldName` to the location of the instance variable `newName` in the NewClass:

```
| im |
im := InstVarMappingArray mappingFrom: OrigClass to: NewClass.
im mapInstVarNamed: #oldName to: #newName.
im mapInstVarToNil: #oldName.
im preserveDynamic: false.
SystemRepository migrateMt: { im }
```

Note that if you do not specify to set `#oldName` to nil, the value at `oldName` will be retained in the migrated instance (now a NewClass) in `#oldName`, as well as being set in the instance variable `#newName`.

## 3.4 FileSystem

FileSystem is a set of classes, ported and adapted from Pharo, that support operations on files and directories. FileSystem provides a much more flexible and feature-rich environment than GsFile. FileSystem performance is comparable to GsFile.

In these notes, the term 'FileSystem' can be used to refer to the entire set of classes that support file and directory operations, or to the specific class named FileSystem. The specific class FileSystem represents the underlying file environment. Most operations within the general File System environment use the class FileReference, which represents a file or directory.

FileSystem is still under refinement, and may be missing features or contain unexpected behavior. The low level support classes in particular may be refactored and/or have protocol modifications.

### Differences from GsFile

FileSystem supports a superset of the functions provided by GsFile for server files, and the API is significantly different.

Some differences:

- FileSystem is implemented using multiple classes to provide a flexible and portable API, while GsFile is implemented as a single class with fixed behaviors.
- FileSystem is implemented based on an FFI interface to the OS file system functions, while GsFile is implemented with user actions (for historic reasons) and C primitives.
- On error, FileSystem signals an appropriate error; GsFile functions return nil and require you to query for the error details.
- FileSystem does not support operations on the GCI client (for example, reading files on a Windows client), while GsFile does.

FileSystem is supported with Linux on x86 and ARM, and macOs on x86 and Apple Silicon. FileSystem is not supported on AIX; on AIX, you must continue to use GsFile.

## Primary API classes

There are a few main entry points for FileSystem.

### FileReference

FileReference is the primary entry point for file and directory operations. You may create files using `FileReference >> /`, `FileReference >> @`, *aString* `asFileReference`, or by using FileSystem class protocol. FileReferences can also be created from another instance of FileReference. The following are all equivalent:

```
'/gshost/test/foo.txt' asFileReference
FileReference / '/gshost/test/foo.txt'
FileReference / 'gshost' / 'test' / 'foo.txt'
FileReference disk @ '/gshost/test/foo.txt'
FileSystem disk / '/gshost/test/foo.txt'
'/gshost/test/' asFileReference / 'foo.txt'
```

The FileReference that is created prints as:

```
FileReference disk @ '/gshost/test/foo.txt'
```

There are a great many supported operations, including both information about the file or directory, and operations on that file or directory. Note that much of the API is on the superclass, AbstractFileReference. The methods include:

- report if the receiver is a file or directory, and for directories, report details on the contained children (files or directories).
- return details of the path, filename, and extension
- return information about a file, such as size, permissions, modificationTime, etc.
- open a read or write stream on the contents, including encoded and binary streams.

Operations on the contents of a FileReference will normally use streams. Methods such as `readStream` and `writeStream` return instances of Zinc-based streams, in this case ZnCharacterReadStream or ZnCharacterWriteStream. For example:

```
rdStream := 'gshost/test/foo.txt' asFileReference readStream.
[rdStream atEnd] whileFalse:
     [report add: rdStream nextLine; lf.].
rdStream close.
```

Variants such as `readStreamDo:` and `writeStreamDo:` automatically close the file after performing the block operation.

By default, these streams provide automatic UTF-8 encoding, although method variants allow you to specify an encoder to support legacy 8-bit encoding.

Binary read and write streams created using `binaryReadStream` and `binaryWriteStream` create instance of FsBinaryFileStream, allowing byte-based operations with no encoding.

### FileLocator

FileLocator provides similar file operations to FileReference, but the file or directory does not need to have an explicit full path. The location for a FileLocator instance can be relative to your environment; for example, the working directory, the directory containing the extent files, or the directory containing gem logs. This allows you to move code between environments without requiring explicit management of the paths. For example the following code does not need to be changed if executed in different environment:

```
outFile := FileLocator home / 'output.txt'.
outFile
   writeStreamDo: [:stream | stream nextPutAll: 'test results']
   ifPresent: [FileAlreadyExistsException signalWith: outFile]
```

You can see the available environment-dependent origins using `FileLocator class >> supportedOrigins`; class methods are available for all, including `home`, `cache`, `temp`, `userData`, `tranlog`, `preferences`, `extent1Directory`, `extent1`, `documents`, `desktop`, and `workingDirectory`.

To find the resolved value of the FileLocator, you can send the message absolutePath; this returns an instance of a kind of Path.

```
FileLocator extent1 absolutePath printString
%
Path / 'gshost' / 'GemStone3.7' / 'data' / 'extent0.dbf'
```

FileLocator understands most of the methods available for FileReference.

### FileSystem

While the primary entry point is FileReference, the class FileSystem can provide a way to define a FileReference. FileSystem supports ordinary disk based file systems and in-memory file systems.

FileSystem has a number of useful methods for defining FileReferences. For example:

`FileSystem disk`

returns a disk-based FileSystem, which can be used with @ or / to create a FileReference, e.g. `FileSystem disk / '/gshost/test/foo.txt'`.

`FileSystem memory`
returns an in memory-based FileSystem, which can be used with @ or / to create a FileReference, e.g. `FileSystem memory / 'foo' / 'bar'`.

`FileSystem workingDirectory`
creates a FileReference to the current working directory (disk-based).

FileSystem includes instance methods that allow you to perform a number of file and directory operations, but these methods are subject to change or removal in later releases. Use instances of FileReference to perform file and directory operations other than creating references and setting the working directory.

- **FsFileDescriptor**

FsFileDescriptor represents the file itself. FileReference `open:` methods return an instance of this class, which can be used to perform basic file operations.

FsFileDescriptors read and write ByteArrays rather than strings.

## Supporting Classes

### Zinc Stream Classes

Subclasses of ZnStream, including ZnBuffed*Stream, ZnEncoded*Stream, and Zn*Encoder, provide read and write streams for FileSystem. Zn*Encoder streams handle the encoding/decoding from UTF-8 and 8-bit (GemStone legacy) encoded files, and the creation of Legacy or Unicode String instances.

### Opening options

FsFileOpeningOptions provide the operating-system specific options for opening files.

### Error Classes

FsError and its subclasses represent FileSystem errors; FileException and its subclases represent FileReference/FileLocator errors. Other errors such as FsUnixError and its subclasses represent low level UNIX file errors, which are generally handled by public API.

### Path classes

Path, RelativePath, and Absolute Path encapsulate a path. A Path can be obtained from a FileReference and vice versa. While Path is an abstract class, you can use it to create instances, e.g. `Path from: '/gshost/test/'` will return an instance of AbsolutePath.

### Store classes

FileSystemStore and subclasses represent the OS file system or memory file system, with differences for specific operating systems.

### Resolver classes

FileSystemResolver and subclasses support resolving various FileLocator origin options, which vary between operating systems.

### FFI support classes

FsLibcInterface and FsCStruct and their subclasses provide the FFI interface to the operating system file commands.

## 3.5 Support for ssh and sftp using OpenSSL

The libssh libraries include support for ssh and sftp. These commonly used functions are now available programmatically using GemStone classes.

The following classes have been added:

GsSshSocket
GsSftpSocket
GsSftpRemoteFile
SshSocketError (error 2759)

Examples of using these classes are provided in class side methods.

### SSH with GsSshSocket

#### Creating the SSH connection

To create a SSH connection, create a client socket using methods inherited from GsSocket, and connect using `sshConnect`.

For example, the following code connects to the host *hostnameOrIP*:

```
sshSock := GsSshSocket newClient.
sshSock connectToHost: hostnameOrIP timeoutMs: 2000.
sshSock userId: userName; password: userPassword.
sshSock disableHostAuthentication.
sshSock sshConnect.
```

As with all sockets, you should close the socket when you no longer need the connection.

```
sshSock close
```

#### SSH operations

Commands are executed using `executeRemoteCommand:`. For example,

```
result := sshSock executeRemoteCommand: 'uptime'.
```

The commands can be executed non-blocking using the following methods:

```
nbExecuteRemoteCommand:
hasCommandInProgress
nbRemoteCommandResult
nbRemoteCommandResultReady
```

### SFTP with GsSftpSocket

#### Creating Sftp connection

Creating an SFPT connection is just like creating an SSH connection. To create a SFTP connection, create a client GsSftpSocket using methods inherited from GsSocket, and connect using the method `sshConnect`.

For example, the following code connects to the host *hostnameOrIP*:

```
sshSftpSock := GsSftpSocket newClient.
sshSftpSock connectToHost: hostnameOrIP timeoutMs: 2000.
sshSftpSock userId: userName; password: userPassword.
sshSftpSock disableHostAuthentication.
sshSftpSock sshConnect.
```

As with all sockets, you should close the socket when you no longer need the connection.

```
sshFtpSock close
```

### Sftp operations

Once the GsSftpSocket instance is created and connected, there are a number of methods supported, including:

`GsSftpSocket >> remoteFileExists:` *aFileOrDirectory*

`GsSftpSocket >> currentRemoteDirectory`

`GsSftpSocket >> createRemoteDirectory:` *dirname*

`GsSftpSocket >> createRemoteDirectory:` *dirname* `mode:` *modeInt*

`GsSftpSocket >> contentsOfRemoteDirectory:` *dirname*

`GsSftpSocket >> contentsOfRemoteDirectory:` *dirname* `matchPattern:` *aString*

`GsSftpSocket >> renameRemoteFile:` *oldName* `to:` *newName*

`GsSftpSocket >> removeRemoteFile:` *filename*

`GsSftpSocket >> removeRemoteDirectory:` *dirname*

### GsSftpRemoteFile for operations on remote files

The class GsSftpRemoteFile represents an instance of a remote file that can be used for reading and writing. You must have an established GsSftpSocket connection, which is an argument to the instance creation methods.

`GsSftpRemoteFile class >> openRemoteFile:` *fileName* `mode:` *openMode* `permissions:` *permInt* `errorIfExists:` *aBoolean* `withSftpSocket:` *aGsSftpSocket*

`GsSftpRemoteFile class >> openRemoteFileAppend:` *fileName* `withSftpSocket:` *aGsSftpSocket*

`GsSftpRemoteFile class >> openRemoteFileReadOnly:` *fileName* `withSftpSocket:` *aGsSftpSocket*

`GsSftpRemoteFile class >> createNewRemoteFile:` *fileName* `withSftpSocket:` *aGsSftpSocket*

`GsSftpRemoteFile class >> createOrOverwriteRemoteFile:` *fileName* `withSftpSocket:` *aGsSftpSocket*

Once an instance of GsSftpRemoteFile has been created, use the instance protocol to read, write, and fetch information about the remote file. See the image for specific methods.

For example, to download a remote file using SFTP:

```
sshSftpSock := GsSftpSocket newClient.
sshSftpSock connectToHost: hostnameOrIP timeoutMs: 2000.
sshSftpSock userId: userName; password: userPassword.
sshSftpSock disableHostAuthentication.
sshSftpSock sshConnect.
remoteSftpFile := GsSftpRemoteFile
   openRemoteFileReadOnly: remoteFileName
   withSftpSocket: sshSftpSock.
localFile := GsFile openWriteOnServer: localFileName.
bytes := remoteSftpFile readAllInto: localFile.
localFile close.
sftpFile close.
```

## 3.6 GsSecureSocket Additional Features

Note that while the term SSL (Secure Socket Layer) is commonly used within GemStone for clarity, specifically the protocol is TLS (Transport Layer Security), using OpenSSL.

### Support for pre-shared keys

Pre-shared keys are symmetric keys shared in advance among the communicating parties. Support has been added to GsSecureSocket for pre-shared keys. The callback protocols are dependent on the TLS version; methods have been added to define and manage the version, see “TLS Version handling” on page 69.

`GsSecureSocket >> setPreSharedKey:` *aByteArray*

Sets the Pre-Shared Key (psk) for the connection to be the bytes contained in *aByteArray*. Raises an exception if the receiver is already connected or listening for a connection. *aByteArray* must have a size of at least 8 bytes, not more than 64 bytes and be a multiple of 8 (16, 24, 32 etc). A key size of at least 16 bytes is recommended.

Invoking this method an additional time overwrites any previously stored PSK. To clear a the previously set PSK, invoke this method with an argument of nil. Returns true on success.

#### Example in image

`preSharedKeyTlsClientExample:` *logToGciClient* `usingPort:` *portNum* `on:` *host*

Pre-shared key transport layer security (PSK-TLS) requires the client and server to both know a pre-shared secret key before the connection is initiated. The key must be at least 8 bytes in size (16 hex digits) and be stored in a ByteArray.

`GsSecureSocket class >> preSharedKeyTlsServerExample:` *logToGciClient* `usingPort:` *portNum* `on:` *host*

Pre-shared key transport layer security (PSK-TLS) requires the client and server to both know a pre-shared secret key before the connection is initiated. The key must be at least 8 bytes in size (16 hex digits) and be stored in a ByteArray.

## Anonymous TLS

Anonymous TLS connections are exposed to man-in-the-middle attacks and are therefore NOT recommended for most applications.

### Example in image

`GsSecureSocket class >> anonymousTlsClientExample:` *logToGciClient* `usingPort:` *portNum* `on:` *host*
Client side of Anonymous TLS example. Anonymous TLS means encrypting data over the socket connection without verifying the client or the server's identity. Because identities are not verified, certificates are not required to establish a connection.

`GsSecureSocket class >> anonymousTlsServerExample:` *logToGciClient* `usingPort:` *portNum* `on:` *host*
Server side of Anonymous TLS example. Anonymous TLS means encrypting data over the socket connection without verifying the client or the server's identity. Because identities are not verified, certificates are not required to establish a connection.

## TLS Version handling

The added methods to set and get actual, maximum and minimum TLS version accept the following argument symbols for *tlsVersionStringOrSymbol*:

- #TLS_VERSION_DEFAULT = use any supported TLS protocol. This is the default behavior.
- #TLS1_VERSION = TLS version 1.0
- #TLS1_1_VERSION = TLS version 1.1
- #TLS1_2_VERSION = TLS version 1.2
- #TLS1_3_VERSION = TLS version 1.3

### Instance methods for the TLS version

`GsSecureSocket >> tlsActualVersion`
Answers a Symbol indicating the actual TLS protocol version used by the receiver. The TLS protocol version for the connection is negotiated by the receiver and its peer during the TLS handshake. This method returns the actual version symbol, not #TLS_VERSION_DEFAULT. Raises an exception if the receiver has not completed the TLS handshake with its peer. For server sockets, the TLS handshake is complete after the #secureAccept succeeds. For client sockets, the TLS handshake is complete after the #secureConnect method succeeds.

`GsSecureSocket >> tlsMaxVersion`
Gets the maximum TLS protocol version for the receiver.

`GsSecureSocket >> tlsMaxVersion:` *tlsVersionStringOrSymbol*
Sets the maximum TLS protocol version for the receiver. Returns true on success or raises an exception if the argument is invalid. Raises an exception if the argument would set the maximum TLS protocol version to be less than a previously set minimum TLS protocol version.

`GsSecureSocket >> tlsMinVersion`
Gets the minimum TLS protocol version for the receiver.

`GsSecureSocket >> tlsMinVersion:` *tlsVersionStringOrSymbol*
Sets the minimum TLS protocol version for the receiver. Returns true on success or raises an exception if the argument is invalid. Raises an exception if the argument would set the minimum TLS protocol version to be greater than a previously set maximum TLS protocol version.

### Class methods for maximum and minimum TLS version

`GsSecureSocket class >> tlsClientMaxVersion`
Gets the maximum TLS protocol version for client sockets created by the receiver.

`GsSecureSocket class >> tlsClientMaxVersion:` *tlsVersionStringOrSymbol*
Sets the maximum TLS protocol version for client sockets created by the receiver. Returns true on success or raises an exception if the argument is invalid. Raises an exception if the argument would set the maximum TLS protocol version to be less than a previously set minimum TLS protocol version.

`GsSecureSocket class >> tlsClientMinVersion`
Gets the minimum TLS protocol version for client sockets created by the receiver.

`GsSecureSocket class >> tlsClientMinVersion:` *tlsVersionStringOrSymbol*
Sets the minimum TLS protocol version for client sockets created by the receiver. Returns true on success or raises an exception if the argument is invalid. Raises an exception if the argument would set the minimum TLS protocol version to be greater than a previously set maximum TLS protocol version.

`GsSecureSocket class >> tlsServerMaxVersion`
Gets the maximum TLS protocol version for server sockets created by the receiver.

`GsSecureSocket class >> tlsServerMaxVersion:` *tlsVersionStringOrSymbol*
Sets the maximum TLS protocol version for server sockets created by the receiver. Returns true on success or raises an exception if the argument is invalid. Raises an exception if the argument would set the maximum TLS protocol version to be less than a previously set minimum TLS protocol version.

`GsSecureSocket class >> tlsServerMinVersion`
Gets the minimum TLS protocol version for server sockets created by the receiver.

`GsSecureSocket class >> tlsServerMinVersion:` *tlsVersionStringOrSymbol*
Sets the minimum TLS protocol version for server sockets created by the receiver. Returns true on success or raises an exception if the argument is invalid. Raises an exception if the argument would set the minimum TLS protocol version to be greater than a previously set maximum TLS protocol version.

## Modified SNI example code in image

To support running the SNI example on RedHat compatible Linux as well as Ubuntu, the method:

```
GsSecureSocket class >> httpsClientExampleForHost:certificate-
Directory:withSniName:
```

has been removed; it is replaced by the new method:

```
GsSecureSocket class >> httpsClientExampleForHost:withSniName:
```

which invokes the added method:

`GsSecureSocket class >> setCaCertLocation`

Certificate verification for the example is designed to Ubuntu and RedHat-compatible Linux. Look for the trusted CA cert file on the current host. If we find it, use it. If we do not, disable cert verification so the examples work correctly.

Ubuntu: /etc/ssl/certs/ca-certificates.crt

Red Hat-compatible: /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem

Returns the CA cert file used or nil if no file was found and certificate verification has been disabled.

## 3.7 Data Encryption using Azure Key Vault

GemStone now supports key management using Microsoft Azure Key Vault, on Linux on Intel. This is similar to the support for AWS (Amazon Web Services) key management added in previous releases.

Azure Key Vault is a cloud service for securely storing and accessing secrets, such as encryption keys. GemStone classes allow you to use Azure Key Vault to hold keys for data encryption.

To use this feature, you must have an account with Azure, have created an Azure Key Vault, and obtained the client secret string. You must also have created one or more keys in that Key Vault. Consult the Azure documentation for more information.

The Azure SDK for C++ (see https://github.com/Azure/azure-sdk-for-cpp) is an open-source project that provides a C++ interface to Azure services. The GemStone distribution includes shared libraries based on the Azure toolkit. These libraries are included in the Linux/x86 distribution.

Version 3.7 of GemStone includes Azure SDK for C++ v1.9.0. In addition, the distribution includes a new external curl library, used to enable linking into the Azure shared library.

The following classes support this feature:

AzureCredentials - encapsulates the key information needed for authentication to the Azure Key Vault.

AzureDataKey - holds a local encryption key that can be used to encrypt and decrypt Strings and ByteArrays. This requires the Azure Key Vault URL and keyname, as well as the AzureCredentials needed to authenticate.

AzureError - signalled when errors occur in Azure API classes.

GemStone’s API allows you to authenticate a key name in a Key Vault, and use the resulting key instance to encrypt a String or ByteArray, which can be decrypted using that same key.

## AzureDataKeys

A key is created using code such as:

```
cred := AzureCredentials
   newWithClientId: '1234567890'
   clientSecret: 'abcdefghij'
   tenantId: '1a2b3c4d5e6f7g8h'.
key := AzureDataKey
   createKeyUsingAzureCredentials: cred
   keyVaultUrl: 'https://keyvaultname.vault.azure.net/'
   keyName: 'keyname'.
```

This instance of **AzureDataKey** should be persisted in the database for ongoing use for encryption and decryption. If the key instance is lost, any data that was encrypted with that key cannot be decrypted, and is permanently inaccessible, even if the credentials are available.

**AzureCredentials** can be recreated as needed. Since a persisted instance of AzureDataKey is locked in newly logged in sessions, the same or a recreated instance of AzureCredentials will be needed to unlock a persisted instance of AzureDataKey.

### Locking

An AzureDataKey may be in a locked or unlocked state. A newly created key is unlocked. The key can be explicitly locked, and is automatically locked when the creating session logs out. When it is saved to disk, it is saved in the locked state.

If a key is locked, it cannot be used to encrypt or decrypt data. Before it can be used, it must be unlocked using an instance of AzureCredentials that is valid to authenticate the AzureDataKey.

The following methods are available:

`AzureDataKey >> lock`
`AzureDataKey >> unlockWithAzureCredentials:` *anAzureCredentials*
`AzureDataKey >> isLocked`
`AzureDataKey >> isUnlocked`

## Encryption/decryption

Encryption and decryption are done by sending a request to the (unlocked) AzureDataKey.

Encryption is performed using AES-OCB authenticated encryption, which ensures data that has been successfully decrypted has not been modified in any way.

`AzureDataKey >> encrypt:` *srcByteObj* `into:` *destByteObj*
Uses the receiver to encrypt *srcByteObj* and stores the resulting encrypted bytes into *destByteObj* as a Base64 string. *srcByteObj* must be a non-empty byte object. *destByteObj* must be a mutable byte object; any contents will be overwritten.

`AzureDataKey >> encryptAndErase:` *srcByteObj* `into:` *destByteObj*
Encrypt *srcByteObj*, erasing all data in *srcByteObj*, and store the resulting encrypted bytes into *destByteObj* as a Base64 string. *srcByteObj* must be a non-empty byte object. *destByteObj* must be a mutable byte object; any contents will be overwritten.

`AzureDataKey >> decrypt:` *srcByteObj* `into:` *destByteObj*
Uses the receiver to decrypt *srcByteObj* and stores the resulting decrypted bytes

into *destByteObj*. *srcByteObj* must be a non-empty byte object containing Base64 encrypted text. *destByteObj* must be an mutable byte object.; any contents will be overwritten.

For example:

```
| credentials key data encryptedData decryptedData |
credentials := AzureCredentials
   newWithClientId: '1234567890'
   clientSecret: 'abcdefghij'
   tenantId: '1a2b3c4d5e6f7g8h'.
key := AzureDataKey
   createKeyUsingAzureCredentials: credentials
   keyVaultUrl: 'https://keyvaultname.vault.azure.net/'
   keyName: 'keyname'.
data := 'Please excuse my dear aunt Sally'.
encryptedData := String new.
decryptedData := String new.
key encrypt: data into: encryptedData.
key decrypt: encryptedData into: decryptedData.
```

## Key rotation

Azure Key Vault provides the ability to do key rotation for improved security. You may update an instance of AzureDataKey to be associated with a new keyName, vault, and credentials, using the following method. To ensure that the change will be committed if successful, this method write locks the receiver and commits the session, so this requires that there are no uncommitted changes.

`AzureDataKey >> changeKeyNameTo:` *newName* `inKeyVault:` *theVault* `usingCredentials:` *theCreds*

Atomically update the receiver to use the key name *newName*, in the vault *theVault*, which is accessed with *theCreds* in a single operation, and commit. The receiver must be unlocked, the session must not have uncommitted changes, and the session must be able to write lock the receiver.

The key *newName* must already exist in the given vault. You may use the existing vault and credentials to rotate the key within the same vault, or use different parameters.

For example:

```
| key credentials data encryptedData decryptedData |
credentials := AzureCredentials
   newWithClientId: '1234567890'
   clientSecret: 'abcdefghij'
   tenantId: '1a2b3c4d5e6f7g8h'.
key := AzureDataKey
   createKeyUsingAzureCredentials: credentials
   keyVaultUrl: 'https://keyvaultname.vault.azure.net/'
   keyName: 'keyname'.
data := 'Please excuse my dear aunt Sally'.
encryptedData := String new.
decryptedData := String new.
key encrypt: data into: encryptedData.
key
   changeKeyNameTo: 'key2'
   inKeyVault: 'https://keyvaultname.vault.azure.net/'
   usingCredentials: credentials.
key decrypt: encryptedData into: decryptedData.
data = decryptedData
```

## Making a copy of a key

An AzureDataKey can be copied. The copy of an AzureDataKey has the same locked state as the original, and can be unlocked using the same credentials.

`AzureDataKey >> copy`
Answer a new object which is a deep copy of the receiver. The lock state of the receiver is preserved when copied, i.e.: if the receiver is unlocked the resulting copy will also be unlocked.

## Key equality

AzureDataKey objects are considered equal (=) if they have the same vault, key name, and the same encrypted key string. Lock state does not affect equality.

## Tracing Azure calls

The following environment variables can be defined in the client environment to print trace information on Azure calls. Setting these GemStone environment variables enables Azure environments variables to perform the debugging; note that the numeric values are defined by Azure, rather than conforming with GemStone's other debugging environment variables.

GS_AZURE_LOG_DIR
A directory in which to create the azure log file

GS_AZURE_LOG_LEVEL

The level of logging. Valid values are 1, 2, 3, 4. The meanings of these values are:

1 – Verbose: Logging level for detailed troubleshooting scenarios

2 –Informational: Logging level when a function operates normally.

3 – Warning: Logging level when a function fails to perform its intended task.

4 – Error: Logging level for failures that the application is unlikely to recover from.

If both these environment variables are set to valid values, Azure creates a log file named

`azure_sdk_`*YY-MM-DD-HH-mm-ss*`.log`

where *YY-MM-DD-HH-mm-ss* are timestamp values, in the directory specified by $GS_AZURE_LOG_DIR.

## 3.8 Changes in Magnitude Classes

### Float printing improvements

GemStone now includes the library double-conversion, which allows more control over floating point display. Float and SmallDoubles such as 9.45, that previously displayed as the technically correct, but not optimal, '9.449999999999999', now display as '9.45'. (#47003)

In addition, Doubles may have up to 17 significant decimal digits, but the earlier code limited this to 16 decimal places. For example, a float created from '7.6000000000000001' was previously printed as 7.600000000000001, and is now 7.6000000000000005. (#46855)

### Converting Non-specials in special range

A special object is one in which the OOP encodes the value, and thus requires no repository space; this is a canonical form of that value. A number of magnitude classes have subclasses with the prefix 'Small', which provide a special/canonical form of a subset of that class's potential values. While new values are created in canonical form, it is possible to have non-canonical instances that have an exactly equivalent canonical value. For example, an upgraded application may have an instance of Fraction with the value 1/3, rather than a SmallFraction with the value 1/3.

To convert a non-canonical to the canonical equivalent, the method `asCanonicalForm` has been added. This either returns self, or the canonical object exactly equivalent to the receiver. Specific implementations apply to Float, Date, DateAndTime, Time, Fraction, LargeInteger, and ScaledDecimal.

Note that this cannot perform a conversion in place; you must modify the object referring to the non-canonical value to refer to the canonical value.

### highBit now ANSI-compliant

The method `Integer`, `LargeInteger`, and `SmallInteger>>highBit` previously returned nil for a zero receiver. Now, this method returns 0 for a zero receiver, as specified by the ANSI spec.

### Other added method

The following method has been added:

`SmallInteger >> minimum32bitInteger`
Return the minimum value representable by a 32bit signed integer

## 3.9 String and Stream-related Changes

### New ReadByteStream classes optimized for Strings

The class ReadByteStream, and the implementation classes ReadByteStreamPortable and ReadByteStringLegacy classes, have been added. This provides a ReadStream that is optimized for performance, and limited to holding kinds of String, MultiByteString, or ByteArray.

The API of ReadByteStream is largely the same as the existing ReadStream classes.

#### readStream reimplemented to return ReadByteStream

The method readStream, which was previously inherited from SequencableCollection to return an instance of ReadStream, has been reimplemented in MultiByteString, String, and ByteArray (and inherited by respective subclasses), to return an instance of ReadByteStream.

### In Unicode mode, String auto-conversion to Unicode16

When the repository is in Unicode Comparison Mode, when an existing instance of String is modified such that it now includes a Character with any codePoint greater than 255, it is auto-converted to an instance of Unicode16. This includes when sending `add:`, `addAll:`, `at:put:`, and similar protocol.

Auto-conversion avoids subsequent performance impacts of comparisons using the ICU library, since strings must be converted for comparison.

Methods that operate on Bytes rather than Characters, such as `String >> addAllBytes:` and `String class >> withBytes:`, are not subject to this auto-conversion. Instance of DoubleByteString are not affected.

### codePointAt:put: now performs auto-conversion if needed

Previously, sending `codePointAt:put:` to a String or DoubleByteString, with a codePoint that is out of range for that class, resulted in an OutOfRange error.

Now, the receiver will be transparently converted to the class that can contain the given codePoint; that is, converted from a String or DoubleByteString to a DoubleByteString or QuadByteString).

### Empty Unicode7 string literals now canonicalized

In traditional String comparison mode, the string literal '' creates an instance of String, which is canonicalized (to the kernel OOP 233473). In Unicode comparison mode, in which '' creates an instance of Unicode7, these were previously not canonicalized. Now, these are canonicalized (to kernel OOP 165377).

## Support for Lz4 encode/decode added to String and ByteArray classes

The following methods have been added, which allow Lz4 compression into a ByteArray, and decompression from Lz4-compressed data in a ByteArray into a byte-format object; normally but not necessarily a kind of String, MultiByteString, or ByteArray.

Compressing a multi-byte character object converts it to a little endian format before compressing, and decompression assumes the data is in little endian format.

You cannot compress, nor decompress into, a Symbol, DoubleByteSymbol or QuadByteSymbol.

Note that the compressed form does not distinguish the original class; for example, String or DoubleByteString; you must specify the correct class in the argument to the `decompress` method.

`ByteArray, String, MultiByteString >> compressWithLz4:` *opCode*
Compress the receiver with lz4 compression, and return an instance of ByteArray containing the compressed bytes. Compression modes include 0/frame mode and 1/blockmode; frame mode should normally be used.

`ByteArray >> decompressWithLz4IntoNewInstanceOf:` *aClass*
`decompressedSize:` *aSize*
Decompress the receiver using lz4 and store the resulting bytes into a new instance of *aClass*. *aClass* must be a byte format Class. The lz4 compression mode (block or frame) used to compress the data is automatically detected by this method. *aSize* is a SmallInteger which indicates the size (in characters, not bytes) of the data when decompressed. If the contents were compressed using lz4 frame mode, *aSize* may be set to 0 indicating the decompressed size is not known. On success, returns a new instance of *aClass* containing the decompressed data.

## ByteArray read/write of binary numbers in native format

The following methods have been added to allow you to read and write numeric values as raw bytes into a ByteArray in the current native byte order. Values are written in native format, and read as if in native format: no swizzling is done.

Note that there are existing methods in ByteArray to write and read numeric values; these methods read and write in big endian format, regardless of platforms.

`ByteArray >> unsigned16AsNativeAt:` *startIndex*

`ByteArray >> unsigned32AsNativeAt:` *startIndex*

`ByteArray >> unsigned64AsNativeAt:` *startIndex*

`ByteArray >> signed16AsNativeAt:` *startIndex*

`ByteArray >> signed32AsNativeAt:` *startIndex*

`ByteArray >> signed64AsNativeAt:` *startIndex*

`ByteArray >> doubleAsNativeAt:` *startIndex*

`ByteArray >> floatAsNativeAt:` *startIndex*

`ByteArray >> signed16At:` *startIndex* `putAsNative:` *anInteger*

`ByteArray >> signed32At:` *startIndex* `putAsNative:` *anInteger*

`ByteArray >> signed64At:` *startIndex* `putAsNative:` *anInteger*

`ByteArray >> unsigned16At:` *startIndex* `putAsNative:` *anInteger*

`ByteArray >> unsigned32At:` *startIndex* `putAsNative:` *anInteger*

`ByteArray >> unsigned64At:` *startIndex* `putAsNative:` *anInteger*

`ByteArray >> doubleAt:` *startIndex* `putAsNative:` *aDouble*

`ByteArray >> floatAt:` *startIndex* `putAsNative:` *aFloat*

The existing ByteArray methods `at:put:signed:width:` and `at:signed:width:` now accept additional `width:` arguments of 512, 1024, or 2048, which specify native format with 2, 4, or 8 bytes, respectively.

## Added methods to return hashes as ByteArrays

The following methods have been added to CharacterCollection and ByteArray; these are similar to existing methods with the added "Bytes", and return the results in a ByteArray.

`md5sumBytes`
Return the 128 bit MD5 checksum of the receiver as a ByteArray. Computation is per RFC 1321 , http://www.ietf.org/rfc/rfc1321.txt, using L. Peter Deutsch's C implementation from: http://sourceforge.net/projects/libmd5-rfc/
For DoubleByteString and QuadByteString, the computation is based on viewing the string as a ByteArray holding big-endian characters.

`sha1SumBytes`
Return the 160 bit SHA1 checksum of the receiver as a ByteArray. Computation is per FIPS PUB 180-3: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.
For DoubleByteString and QuadByteString, the computation is based on viewing the string as a ByteArray holding big-endian characters.

`sha256SumBytes`
Return the 256 bit SHA256 checksum of the receiver as a ByteArray. Computation is per FIPS PUB 180-3: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.
For DoubleByteString and QuadByteString, the computation is based on viewing the string as a ByteArray holding big-endian characters.

`sha3_224SumBytes`
Return the SHA3 224 bit checksum of the receiver as a ByteArray. Computation is per FIPS PUB 202: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf.
For DoubleByteString and QuadByteString, the computation is based on viewing the string as a ByteArray holding big-endian characters.

`sha3_256SumBytes`
Return the SHA3 256 bit checksum of the receiver as a ByteArray. Computation is per FIPS PUB 202: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf.
For DoubleByteString and QuadByteString, the computation is based on viewing the string as a ByteArray holding big-endian characters.

`sha3_384SumBytes`
Return the SHA3 384 bit checksum of the receiver as a ByteArray. Computation is per FIPS PUB 202: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf.
For DoubleByteString and QuadByteString, the computation is based on viewing the string as a ByteArray holding big-endian characters.

`sha3_512SumBytes`
: Return the SHA3 512 bit checksum of the receiver as a ByteArray. Computation is per FIPS PUB 202: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf.
For DoubleByteString and QuadByteString, the computation is based on viewing the string as a ByteArray holding big-endian characters.

`sha512SumBytes`
: Return the 512 bit SHA512 checksum of the receiver as a ByteArray. Computation is per FIPS PUB 180-3: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.
For DoubleByteString and QuadByteString, the computation is based on viewing the string as a ByteArray holding big-endian characters.

## Utf8 now disallows #readStream

The #readStream method, previously inherited from SequenceableCollection, has been disallowed in Utf8. Utf8 is intended to be only used in the encoded form; you should decode this prior to creating a readStream on it.

## Other Changes

### atOrNil: added

This method has been added to ByteArray, String and String's subclasses DoubleByteString and QuadByteString, as part of the optimizations for ReadByteStream.

`atOrNil:` *anIndex*
: Returns the Character at *anIndex*, or nil if *anIndex* is beyond end of the receiver

This is disallowed for Utf8 and Uft16.

### Stream classes consistently return objects of their collection class

Previously, some methods could return an empty instance of String, rather than an instance of the collection class.

## 3.10 JsonParser and JsonPetitParser

Previous releases includes a PetitParser-based JSON Parser, which could read a JSON string and create the equivalent Smalltalk object structure. Using this class required some experience with PetitParser.

In this version, a new JSON parser has been added. This is a faster and simpler recursive descent parser, not related to PetitParser.

The class name 'JsonParser' is now applied to the new, simple JSON Parser. The old JSON parser, formerly 'JsonParser', is now 'JsonPetitParser'. JsonPetitParser is deprecated.

The API for the new JsonParser is much smaller; like JsonPetitParser, the new JsonParser is also accessed using the class method `parse:`. The new JsonParser signals errors on `parse:` of invalid JSON, unlike JsonPetitParser.

### Upgrade impacts

Compiled methods that contain references to the JsonParser class in an earlier version, will refer to JsonPetitParser after upgrade and continue to work as before. Provided you are only using the JsonParser to parse JSON, it is optional to get the improved performance by recompiling methods that reference the JsonParser class by name.

If you are using JsonParser as part of a PetitParser customization, references to the class by name must be updated to refer to 'JsonPetitParser' instead of 'JsonParser'. It is recommended to edit the source of methods that reference JsonParser class to reference JsonPetitParser, and edit the superclass of subclasses of JsonParser.

If you have subclasses of JsonParser that must be filed in, these will be compiled and refer to the new JsonParser class. If this subclass directly references inherited instance variables, compilation of those methods will fail; and depending on the specific modifications may or may not be functional. In most cases before filein, you should modify the code on disk to refer to JsonPetitParser.

See the *Installation Guide for v3.7*, Upgrade chapter, for recompilation details

## 3.11 External Session Changes

### GsTsExternalSession no longer requires cpp/gcc/g++

Previously, GsTsExternalSessions computed the FFI calls for the GciTsLibrary on demand, using `cpp` to parse `gcits.hf`. This required cpp, and the gcc/g++ compilation environment; including for production use of GsTsExternalSessions.

Now, the FFI calls are precomputed for the current version, and provided in a new file that is loaded during upgrade, `$GEMSTONE/upgrade/GciTsLibrary.dat`.

Note that this applies to GsTsExternalSessions that login with the same GemStone version as the current Gem. For logins to Stones running a different version of GemStone/S 64 Bit, `gcits.hf` still must still be parsed, and cpp/gcc/g++ continues to be required.

### Added methods for GsTsExternalSession

The following methods have been added:

`GsTsExternalSession >> debugGem:` *processId* `token:` *anInteger*
Use GciTsDebugConnectToGem instead of GciTsLogin to establish a connection. Arguments should be copied from the DEBUGGEM printout in a topaz .out file or log file. All normal login arguments are ignored.

`GsTsExternalSession >> newUtf8String:` *aUtf8* `toUnicode:` *aBoolean*
Create a new instance of Utf8 (if aBoolean is false) or a Unicode7, Unicode16 or Unicode32 (if aBoolean is true), in the external session, containing the contents of *aUtf8*. Returns the oop of the object that was created in the external session.

`GsTsExternalSession >> releaseOop:` *anOop*
Queue *anOop* (an oop in the external session) to be released from the ExportSet of the remote session; once 100 oops have been released, the oops are releaseed, allowing the objects to be garbage collected.

`GsTsExternalSession >> releaseOops:` *anArray*
Release the oops in *anArray* (oops in the external session) from the ExportSet of the remote session, allowing the objects to be garbage collected.

`GsTsExternalSession >> send:` *aSelector* `to:` *rcvrOop* `withOops:` *anArray*
The same as `send:to:withArguments:`, but the arguments are already in the form of oops.

### Return value from GsTsExternalSession >> waitForReadReadyTimeOut:

Previous, the method `GsTsExternalSession >> waitForReadReadyTimeOut:` *msToWait* returned self for success, and signaled an error if the timeout expired or an error occurred.

Now, this method returns true on success, false if there was a timeout, and signals an error if there was an error.

### Easier to use GEM_HALT_ON_ERROR in external sessions

The default for haltOnErrNum has been changed to -1, to allow the Gem's configuration file setting for GEM_HALT_ON_ERROR to be in effect for external sessions.

## 3.12 Other Changes and Enhancements

### ProfMonitor improved formatting for object creation tree report

The tree report produced when doing object creation tracing in ProfMonitor or ProfMonitorTree has improved formatting for deeper trees.

### GsSecureSocket example updates

The GsSecureSocket class method

`httpsClientExampleForHost:certificateDirectory:withSniName:`

has been renamed to

`httpsClientExampleForHost:withSniName:.`

The following methods has been added:

`GsSecureSocket class >> setCaCertLocation`
On linux: try to find the trusted CA cert file on this host. If we find it, use it. If we do not, disable cert verification so the examples work correctly.

### isValidIdentifier, validateIsIdentifier optimized

The following methods are now implemented in primitives:

`CharacterCollection >> isValidIdentifier`

`Object >> validateIsIdentifier`

### Other Added methods

`GsFile >> beforeEnd`
Answer true if the receiver is not positioned at the end, false otherwise

`GsFile >> print:` *anObject*
Writes the print representation of the argument to the GsFile receiver instance.

`GsHostProcess >> readOutErr`
Assuming both out and err are sockets, attempt to read from both and return combined result.

`GsHostProcess >> redirectStderrToStdout`
This should be used rather than stderrPath: if you want both stdout and stderr redirected to the same file.

`GsNetworkResourceString >> gemConfig:` *aString*
Include a -C configuration definition for the receiver. *aString* should be a valid **-C** argument; for example, 'GEM_TEMPOBJ_CACHE_SIZE=100MB;'.

`Array >> copyNotNilFrom:` *startIndex* `to:` *stopIndex*
Return an Array containing the non-nil elements that are within the specified offsets of the receiver.

`Array >> indexOfNotNil:` *startOffset* `to:` *endOffset*
If *startOffset* <= *endOffset*, returns the first offset within in the specified range of a non-nil element of the receiver. If *startOffset* > *endOffset*, returns the last offset within the specified range of a non-nil element of the receiver. Returns zero if all are nil. Intended for use only on Arrays of size <= 2000. Performance on larger arrays will be slow.

`TestAsserter >> assert:` *anObject* `identical:` *otherObj*
Assert that two objects are identical.

`TestAsserter >> fail`
Cause the test to report failure unconditionally

`TestAsserter >> fail:` *descriptionString*
Cause the test to report failure unconditionally, with the given description.

## 3.13 Deprecated and Removed Classes and Methods

See “GciDirtyTrackedObjs functionality removed” on page 85 for additional removed methods related to the remove Tracked Dirty Objects set.

### AbstractCharacter removed

The abstract class AbstractCharacter has been removed; Character superclass is now Magnitude. The AbstractCharacter definition is now in ObsoleteClasses.

## Newly deprecated Methods

The following methods are newly deprecated. See the deprecation message for the replacement.

```
GsProcess >> stepIntoFromLevel:
GsProcess >> stepOverFromLevel:
GsProcess >> stepThroughFromLevel:
GsProcess >> terminateTimeoutMs:
GsProcess >> threadRubyEnvId
System >> sessionIdHoldingGcLock
```

## Obsolete Classes not in new images

In a new image, Globals will not include the long-obsolete placeholder classes:

```
GsfClassDefinitionInfo
GsfModificationLog
```

In a new image, (Globals at: #ObsoleteClasses) will not include the following classes:

```
ObsoleteConstrainedPathEvaluator
ObsoleteConstrainedPathTerm
ObsoleteEqualityIndexQueryEvaluator
ObsoleteIdentityIndexQueryEvaluator
ObsoleteIndexedQueryEvaluator
ObsoletePathEvaluator
ObsoletePathSorter
ObsoleteSetValuedPathEvaluator
ObsoleteSetValuedPathTerm
```

These will continue to exist in upgraded images.

## Removed Public Methods

The following methods have been removed, associated with changes mentioned elsewhere in these release notes:

```
CPreprocessor >> defaultSparcSolarisDefinitions
CPreprocessor >> defaultX86SolarisDefinitions
Breakpoint class >> trappable:
GsSecureSocket class >> httpsClientExampleForHost:
   certificateDirectory:withSniName:
GsTestCase >> assert:isEquivalentTo:
Object >> removeObjectFromBtrees
```

## Removed Previously-deprecated Methods

The following methods were previously deprecated, and have been removed:

```
ExecBlock >> valueNowOrOnUnwindDo:
Repository >> shrinkExtents
System class >> cacheStatisticsDescription
```

## Removed Private Methods

```
AbstractException >> _signalAsync
AbstractException >> _signalFromPrimitive:
```

```
AbstractException >> _signalGcFinalize
AbstractException >> _signalTerminateProcess
AbstractException >> _signalTimeout
AbstractException >> _signalWith:
Behavior >> _transientSessionMethodBehaviorsCacheName
CCallout class >> _errno:
CharacterCollection >> _validIdentifier:
ClassOrganizer class >> _newExcludingGlobals
Delay >> _activate
ExecBlock >> _valueNowOrOnUnwindDo:
ExecBlock >> _valueNowOrOnUnwindDoPrim:
GsFileIn >> compileMethodIn:
GsFileIn >> fileStream:
GsFileIn class >> fromNestedClientPath:to:
GsFileIn class >> fromNestedPath:on:to:
GsFileIn class >> fromNestedServerPath:to:
GsFileIn class >> _fromStream:
GsFileIn class >> _fromStream:to:
GsNMethod >> _breakOperation:forStepPoint:
GsNMethod >> _setBreakAtIp:operation:frame:process:
GsNMethod >> __setBreakAtIp:operation:frame:process:
GsNMethod class >> _setBreakAtIp:operation:frame:process:
GsProcess >> _activate
GsProcess >> _continue
GsProcess >> _finishTermination
GsProcess >> _isRubyThread
GsProcess >> _serviceTerminationInterrupt
GsProcess >> _stepOverFromLevel:
GsProcess >> _stepOverInFrame:mode:replace:tos:
GsSocket >> _waitingProcessesInto:
GsSocket >> _waitingProcessesInto:inGroup:
GsTsExternalSession >> _getObjInfo:buffer:
IdentityBag >> _basicAdd:
IdentityBag >> _rcIncludes:
IdentityBag >> _rcIncludesValue:
IndexingErrorPreventingCommit >> _signalWith:
Integer >> _floatParts
Object >> addObjectToBtreesWithValues:
Object >> _respondsTo:private:flags:
Object >> _errorsDuringPreventCommit:
RcIdentityBag >> _thisSessionRemovalIndex
RcKeyValueDictionary >> _keyCollisionOk
SmallInteger >> _floatParts
System class >> _deprecatedCacheStatisticsDescription
System class >> _disallowSubsequentCommits:
System class >> _primitiveAbort
System class >> _sessionsReportExcluding:
System class >> _hostWaitForDebuggerIfSlow
```

Chapter
4
# Changes in the GCI Interfaces

This chapter describes changes and updated information related to the GemBuilder for C and FFI interfaces, including:

## 4.1 GCI changes

### GciDirtyTrackedObjs functionality removed

The Dirty Tracked Objects set is no longer needed, and has been removed.

#### Removed GCI calls

```
GciDirtyTrackedObjs
GciReleaseAllTrackedOops
GciReleaseTrackedOops
GciSaveAndTrackObjs
GciTrackedObjsFetchAllDirty
GciTrackedObjsInit
GCI_JIS_CHAR_TO_OOP
GCI_OOP_TO_JIS_CHAR
```

#### Removed Image Methods related to TrackedObjects

The GciLibrary functions associated with the removed GCI calls are no longer included.

In addition, this method has been removed:

```
System class >> gciTrackedObjsInit
```

### GsBitmap hiddenSetSpecifiers

#TrackedDirtyObjs is now #TrackedDirtyObjs_NotImplemented

#GciTrackedObjs is now #GciTrackedObjs_NotImplemented

## Added Lz4 compression functions

### GciCompressUsingLz4

```
(int) GciCompressUsingLz4(
    const char* src,
    char* dst,
    int srcSize,
    int dstCapacity );
```

Wrapper for the lz4 function `LZ4_compress_default()`.

Compress *srcSize* bytes from buffer *src* into already allocated *dst* buffer of size *dstCapacity*. Compression is faster, and guaranteed to succeed, if *dstCapacity* >= `LZ4_compressBound(`*srcSize*`)`.

If the function cannot compress *src* into a more limited *dst* buffer size, compression stops, function result is zero, and *dst* content is undefined (invalid).

### GciUncompressUsingLz4

```
(int) GciUncompressUsingLz4(
    const char* src,
    char* dst,
    int compressedSize,
    int dstCapacity );
```

Wrapper for `LZ4_decompress_safe()`.

Decompress the compressed string at **src* and place the result in **dst*. **dst* must be already allocated. The exact size of the compressed string at **src*, and the maximum size for the destination must be provided; if *dstCapacity* is insufficient, decompression stops and the functions returns a negative result.

Returns the number of bytes decompressed into **dst*.

## Added variants of existing functions

### GciDbgEstablishToFile_

```
(BoolType) GciDbgEstablishToFile_(
   const char *fileName,
   GciErrSType *err);
```

A variant of `GciDbgEstablishToFile()`, with the additional argument **err*. `GciDbgEstablishToFile_()` returns error info in **err* if the function result is FALSE.

## One-time login support

The new feature to allow a one-time login password to be created (described starting on page 25) can be used in GemBuilder for C, by specifying the new flag GCI_ONETIME_PASSWORD (0x400) with the loginFlags in the argument for GciLoginEx() or GciLoginEx_().

## Configuration file handling support

v3.7 includes additional options and finer control over where and how GemStone executable and system configuration files are handled; this is described starting on page 36.

Support for this in GBC is provided by GciInit_, which is a variation of GciInit that includes additional arguments, and the new class GciCfgFileArgs.

### GciInit_

```
(int) GciInit_(
   BoolType quietMode,
   GciProcessType gciProcessType,
   GciCfgFileArgs *gciCfgFileArgs,
   char *configFileWarn,
   size_t warnSize);
```

A variant of GciInit, with additional arguments to support new configuration file semantics. Only one of GciInit/GciInit_ should be used.

**gciCfgFileArgs* is the results of parsing command line arguments for configuration files; see the definition for GciCfgFileArgs in gci.ht. *gciProcessType* should be GCI_PROCESS_GEM in linked customer applications; GciProcessType is defined in gci.ht.

This function returns 0 for success, -1 for errors in the configuration file, and -2 for memory allocation errors in libgcilnk.

## Removed Mnemonics

The definitions for the obsolete classes OOP_CLASS_EUC_STRING, OOP_CLASS_INVARIANT_EUC_STRING, OOP_CLASS_EUC_SYMBOL, OOP_CLASS_JIS_CHARACTER, and OOP_CLASS_JIS_STRING have been renamed to include an additional _.

## 4.2 GCI thread-safe changes

The thread-safe GCI is provided in gcits.hf.

### Added Thread-safe functions

#### GciTsDebugConnectToGem

```
(GciSession) GciTsDebugConnectToGem(
       int gemPid,
       GciErrSType *anEerr);
```

Connect to a gem's listening debug port on localhost; only supported for gems running on the same node. See the GEM_LISTEN_FOR_DEBUG configuration parameter and DEBUGGEM topaz command. The *gemPid* argument to GcitsDebugConnectToGem is the first arg to topaz DEBUGGEM. A call to GciTsDebugConnectToGem must be followed by a call to GciTsDebugStartDebugService.

#### GciTsDebugStartDebugService

```
(BoolType) GciTsDebugStartDebugService(
       GciSession aSession,
       uint64 debugToken,
       GciErrSType *anErr);
```

This function must be preceded by GciTsDebugConnectToGem which creates a GciSession. Using this session, GciTsDebugStartDebugService will validate the *debugToken* argument and interrupt any Smalltalk execution in progress in the target gem by the equivalent of `GciTsBreak(`*aSession*`, FALSE, &`*anErr*`)`. *debugToken* is the second arg to topaz DEBUGGEM. To terminate the debug session, and assumming the session being debugged was waiting in `System class` >> waitForDebug, the client should execute `System cancelWaitForDebug` or `System _cancelWaitForDebugExitClient`, and then call GciTsLogout.

#### GciTsNbPoll

```
(int) GciTsNbPoll(
       GciSession session,
       int timeoutMs,
       GciErrSType *gciErr);
```

Function results:

1 - result or error is ready, call GciTsNbResult to get the result or error.
0 - result is not ready
-1 - error, (invalid session, no NB call in progress, peer disconnected), details will be in *gciErr*.

Argument *timeoutMs* may be:

0 - do not block, return immediately
-1 - block forever until the Nb call finishes or an error occurs.
> 0 block for up to *timeoutMs* ms before returning the result

## 4.3 Foreign Function Interface (FFI) related changes

### errno: unsafe; use alternate methods to access to CCallout errno

Since CCallout class methods `errno` and `errno:` access a value stored in the GsProcess, there is a risk of that two FFI calls (within a single GsProcess) may access this, and set/retrieve incorrect values. (#49432)

The methods `errno` and `errno:` now signal an error and should not be used. You may still used `callWith:` if you do not need access to the errno information.

The method `CCallout >> callWith:errno:`, which was added in v3.6.4, can be used when the errno information is needed. See the method image comments for details.

Generated library code such as GciLibrary methods now invoke `callWith:errno:` rather than `callWith:`.

### In-memory GC now responsive to memory use by FFI

In-memory garbage collection scans are now also responsive to memory pressure resulting from heavy use of `gcMalloc:`. This avoids excessive CHeap growth with intensive use of FFI with limited pressure on memory from Smalltalk objects.

### CPreprocessor failed to cleanup temporary files

CPreprocessor could leave behind temporary files (#50182)

## 4.4 Updated Compile and Link Information

### Linux Compile and Link Information

#### Complier version

Red Hat Linux ES 9.2: gcc/g++ 11.3.1

Red Hat Linux ES 8.7: gcc/g++ 8.5.0

Red Hat Linux ES 7.9: gcc/g++ 4.8.5

Ubuntu 20.04 LTS: gcc/g++ 9.4.0

Ubuntu 22.04 LTS: gcc/g++ 11.3.0

#### Debugger version

Red Hat Linux ES 9.2: gdb 10.2-10.el9

Red Hat Linux ES 8.7: gdb 9.1

Red Hat Linux ES 7.9: gdb 9.1

Ubuntu 20.04 LTS: gdb 9.1

Ubuntu 22.04 LTS: gdb 112.1

#### Compiling a user action or GCI application

```
g++ -fmessage-length=0 -fcheck-new -O3 -ggdb -m64 -pipe
   -D_REENTRANT -D_GNU_SOURCE -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 -Wno-aggregate-return -Wswitch
   -Wshadow -Wunused-value -Wunused-variable -Wunused-label
   -Wno-unused-function -Wchar-subscripts -Wmissing-braces
   -Wmissing-declarations -Wmultichar -Wparentheses -Wsign-compare
   -Wsign-promo -Wwrite-strings -Wreturn-type
   -Wno-format-truncation -Wuninitialized
```

#### Linking a user action library

```
g++ -shared -Wl,-hlibuserAct.so userCode.o $GEMSTONE/lib/gciualib.o
   -o libuserAct.so -m64 -Wl,-Bdynamic,--no-as-needed -lpthread
   -Wl,--as-needed -lcrypt -ldl -lc -lm -lrt -znoexecstack
```

#### Linking a GCI application

```
g++ userCode.o $GEMSTONE/lib/gcirtlobj.o -m64
   -Wl,-Bdynamic,--no-as-needed -lpthread -Wl,--as-needed -lcrypt
   -ldl -lc -lm -lrt -z noexecstack -Wl,-traditional -Wl,-z,lazy
   -o userAppl
```

## AIX Compile and Link Information

### Complier version

IBM XL C/C++ for AIX, V16.1

### 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
   -DFLG_AIX_VERSION=version -D_REENTRANT -D_THREAD_SAFE
   -qminimaltoc -qlist=offset -qmaxmem=-1 -qsuppress=1500-010:1500
   -029:1540-1103:1540-2907:1540-0804:1540-1281:1540-1090:1540-2988
   -qnoeh -I$GEMSTONE/include -c userCode.c -o userCode.o
```

Depending on your version of AIX, you need to include -DFLG_AIX_VERSION=71 or -DFLG_AIX_VERSION=72.

Also 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 -LcompilerInstallDir/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/compilerInstallDir/lib -lpthreads -lc_r -lC_r -lm -ldl
   -lbsd -lpam -Wl,-brtllib -o userAppl
```

## DARWIN Compile and Link Information

### Complier version

Ventura: Apple clang version 14.0.3 (clang-1403.0.22.14.1)

Monterey: Apple clang version 14.0.0 (clang-1400.0.29.202)

Big Sur: Apple clang version 12.0.0 (clang-1200.0.32.28)

### Debugger version

lldb-1400.0.30.3
Apple Swift version 5.7

### Compiling a user action or GCI application

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

The following warn flags are recommended for compilation:

```
   -Wno-format -Wtrigraphs -Wcomment -Wsystem-headers -Wtrigraphs
   -Wno-aggregate-return -Wswitch -Wshadow -Wunused-value
   -Wunused-variable -Wunused-label -Wno-unused-function
   -Wchar-subscripts -Wno-conversion -Wmissing-braces -Wmultichar
   -Wparentheses -Wsign-compare -Wsign-promo -Wwrite-strings
   -Wreturn-type -Wno-nullability-completeness
   -Wno-expansion-to-defined
```

### 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 Professional 2019 Version 16.11.24
VisualStudio.16.Release/16.11.24+33328.57
Microsoft .NET Framework Version 4.8.09032
Visual C++ 2019   00435-60000-00000-AA889

### 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 /EHsc -volatile:iso
   /DNATIVE /I 'VisualStudioInstallPath\include'
   /I 'VisualStudioInstallPath\Tools\MSVC\14.29.30133\atlmfc\include'
   /I 'VisualStudioInstallPath\Tools\MSVC\14.29.30133\include'
   /I '%GEMSTONE%include' /c 'userCode.c' -FouserCode.obj
   -FduserCode.pdb
```

## Linking a GCI application

```
link /LIBPATH:VisualStudioInstallPath\Tools\MSVC\14.29.30133\lib\x64
   /LIBPATH:VisualStudioInstallPath\Tools\MSVC\14.29.30133\atlmfc\lib\x64
   /LIBPATH:WindowsKitsInstallPath\10\lib\10.0.18362.0\um\x64
   /LIBPATH:WindowsKitsInstallPath\10\lib\10.0.18362.0\ucrt\x64
   /DEBUG /RELEASE /OPT:REF /INCREMENTAL:NO /MAP /nologo
   /MANIFEST /MANIFESTFILE:userAppl.exe.manifest
   /MANIFESTUAC:level='asInvoker' /FORCE:MULTIPLE userCode.obj
   %GEMSTONE%lib\gcirtlobj.obj %GEMSTONE%lib\gcirpc.lib ws2_32.lib
   netapi32.lib advapi32.lib comdlg32.lib user32.lib gdi32.lib
   kernel32.lib winspool.lib Secur32.lib Dbghelp.lib
   /NODEFAULTLIB:libcmt.lib /out:userAppl.exe
```

# Chapter 5 Bug Fixes

The following bugs were present in v3.6.6 and are fixed in this version.

## Checkpoint when tranlogs full can result in Stone shutdown

When the transaction logs are full, the Stone pauses to wait for space to become available. If a checkpoint starts while the Stone is waiting, it may have resulted in the Stone shutting down. (#49704)

## Cache warming could have caused commit record backlog

Cache warming runs in transaction, and when performing a large amount of warming over a slower connection, it could result in a commit record backlog. Now, while still running in transaction, cache warming monitors the commit record backlog and aborts if necessary. (#48419)

## A stuck login with UserSecurityData locked may block further logins

During login, the session briefly locks the UserSecurityData for update. If the login process has problems and does not complete, so the UserSecurityData remains locked, further logins for that user may fail. (#49866)

## When commits suspended, symbol creation causes SymbolGem failure

When commits are suspended (as by `Repository>>suspendCommitsForFailover`), the SymbolGem cannot commit. If a symbol is created (e.g. by executing code), the SymbolGem errored and shutdown, and the Gem hung. (#50185)

## Finding references failed to report objects in large IdentityBags

The operations `allReferences:`, `findAllReferences`, and `findReferences` do not include references for an objects in a large IdentityBag (more than 2K elements). (#40163)

## Backup and Restore issues

### Very slow restore of compressed tranlogs

Restoring compressed tranlogs is unusably slow; the restore is doing excessive seeks due to a prematurely cleared buffer. This affects both gzip and lz4 compressed tranlogs. (#50692)

### Backup to a file could produce unclear error message

When making a programmatic backup and including a directory with the pathname, the error message reported was not helpful. (#50239)

### Commit as restoreFromBackup starts could cause Stone shutdown

When `restoreFromBackup:` is initiated, another session such as the SymbolGem or ReclaimGem could commit. This would cause the Stone to shutdown. The restore code now checks for this condition and errors. (#50184)

## RcIdentityBag >> remove:otherwise: errored

The inherited implementation was not appropriate for RcIdentityBag, so this method errored; an RcIdentityBag-specific method has been added. (#50588)

## Issues with stacks and memory stats when using setcap to enable LargeMemoryPages or OOM killer protection

To configure Linux to allow huge pages, to protect critical processes from the Linux OOM killer, and to lock the shared page cache in memory, the Installation Guide directed using setcap to provide specific capabilities to GemStone executables. Setting these capabilities has the effect of preventing gdb from attaching to generate C stacks, and statmonitor from collecting certain host process statistics. (#50580).

## Bugs in Configuration Parameters

### Improved distribution over extents for DBF_ALLOCATION_MODE

Previously, allocation to extents was by free pages, not by extent size, which meant that in certain configurations the extents would not grow per the weights in DBF_ALLOCATION_MODE. (#49992)

### Computation of SHR_PAGE_CACHE_NUM_PROCS too small

When SHR_PAGE_CACHE_NUM_PROCS is default (-1), it computes a value based on STN_MAX_SESSIONS and other configuration parameters. This computation was incorrect, resulting in a value that was somewhat smaller than necessary, in the case where every slot was required. (#49458)

## Numeric and Time related issues

### Float = and ~= always returned false for SmallScaledDecimal arguments

The primitives supporting the Float methods = and ~= return false, rather than failing the primitive, for SmallScaledDecimal arguments. This bypasses the handling that converts the SmallScaledDecimal to a comparable value. (#50597)

### A Time did not compare equal to equivalent SmallTime

A SmallTime and a Time that represent equivalent points in time did not compare as equal using = (equal sign). (#50118)

### SmallDateAndTime passivate activate failed for years 2035-2072

Activating a passivated SmallDateAndTime in the approximate year range 2035 to 2072 errored; the activation expected a SmallScaledDecimal value in the internal storage, but the stored value was not in the SmallScaledDecimal range. (#50172)

### CharacterCollection >> isDigits incorrect for empty string

For an empty String, this method incorrectly returned true. (#50145)

### Float >> asDecimalFloat results not as precise as possible

For many values, converting a Float to a DecimalFloat using `asDecimalFloat` returned a result that was not the closest possible DecimalFloat to the given Float. (#50212)

### isExceptionalFloat returned true for subnormal numbers

The method `Float>>isExceptional` incorrectly returned true for subnormal Floats. Subnormal values are not exceptional; exceptional numbers are limited to infinity and NaN (not a number) values. (#50378)

## $GEMSTONE paths containing symlinks

The NetLDI and gemnetobject and related scripts did not properly handle the case where the $GEMSTONE path included a symbolic link, and the target of the symbolic link changed while the NetLDI was running. The gemnetobject (and related scripts) used the current value of $GEMSTONE, rather than the argument from the NetLDI, which created an unclear state.

Now, if the resolved $GEMSTONE in the environment in which login is initiated does not match the $GEMSTONE path resolved when the NetLDI was started, login will fail with the message "expanded value of GEMSTONE environment variable changed" (#50173)

As a result, logins will also fail if the `sys` directory within the GemStone distribution is a symbolic link. This is not a supported configuration. The error reported to topaz on login is "socket read EOF"; the actual error message is in the gem log.

## File handle leak in GsTsExternalSession

In-memory GC of a logged-in GsTsExternalSession caused a file handle leak (#50624)

## stoned exited with code 3 on clean shutdown

On a clean stone shutdown using stopstone, stopstone exited with code 0. However, the stoned process itself exited with code 3. Now, stoned will exit with code 0 on clean stopstone. (#50222)

## FFI structures may fault out of memory and lose C data

There are cases in which CByteArrays and CPointers that used by FFI can be committed, faulted out, and lose C data when faulted in. (#49769).

Recent releases include workarounds to avoid the subsequent Gem crash due to bug 49765; the underlying condition is fixed in v3.7.

## Lost InterSessionSignal or other interrupt

Due to non-atomic updates in signal handling, it was possible for an InterSessionSignal to be lost or delayed, or other interrupt to be handled incorrectly; this might, for example, cause failure to respect a socket read or write timeout. (#49877)

## GsHostProcess could not redirect stdout and stderr to same file

To redirect both stdout and stderr to the same disk file, it did not work to individually specify the path for each. A new method has been added to support this use case: `GsHostProcess >> redirectStderrToStdout`. (#50445)

## GsSignalingSocket >> acceptTimeoutMs: may fail when receiver is a GsSecureSocket

Due to an inappropriate use of `_peek:`, `GsSignalingSocket >> acceptTimeoutMs:` failed when the receiver was an instance of GsSecureSocket. (#50504)

## asJson problems with Date, Time, DateTime, and DateAndTime

The results produced by sending `asJson` were empty for instances of Date, Time, and DateAndTime. While DateTime produced a valid string, it was not quoted correctly. (#50391)

## Upgrading did not preserve default legacy indexes

In v3.4, the BtreePlus indexes were introduced, and are used by default in new repositories. Upgraded repositories defaulted to legacy indexes. This behavior on upgrade was lost due to code reorganization in v3.6; upgrading from pre-3.4 to 3.6 or later would result in the default for GsIndexOptions to include btreePlusIndex, rather than legacyIndex. (#50318)

## Class subclassed from nil did not have default doesNotUnderstand handling

If a class was subclassed from nil, and `doesNotUnderstand:` was not implemented for that subclass, sending a message selector to an instance that was not understood resulted in infinite recursion. (#49768)

## Bitmap written by #saveWriteSetUnionToFile not readable

The Admin GcGem option #saveWriteSetUnionToFile, which is false by default, triggers the AdminGem to write a bitmap file containing the write set union. This file was not readable by hidden set nor by GsBitmap protocol. (#49730)

## GsTestCase assert:isEquivalentTo: failed to catch errors

This method has been removed, to avoid a false impression of providing test coverage. This method supported returning true for two numeric arguments that are somewhat close, but not exactly equal. This resulted in failure to detect actual incorrect results. (#50159)

## PositionableStreamLegacy positionW, positionW: did not signal Warning

These methods are intended to signal a Warning to verify code when transitioning an application from 1-based to 0-based legacy streams (that is, not PostionableStreamPortable). Previously, these wrote a stack to the Stone; now, a Warning is signaled.

## PPParser >> child included a send to #assert:

The PetiteParser class PPParser method #child included a send to #assert:, although #assert: is not implemented within that hierarchy. (#50193)

## gemnetobject -T, -N argument was ignored if no separator

When setting the temporary object cache size or native code using gemnetobjects (e.g. set gemnetid 'gemnetobject -T *value*'), the -T -N argument was ignored if there was no space between the -T or -N and the argument. (#50338)