GemStone/S 64 Bit™
Release Notes

Version 3.7.2

December 2024

INTELLECTUAL PROPERTY OWNERSHIP

This documentation is furnished for informational use only and is subject to change without notice. GemTalk Systems LLC assumes no responsibility or liability for any errors or inaccuracies that may appear in this documentation.

Warning: This computer program and its documentation are protected by copyright law and international treaties. Any unauthorized copying or distribution of this program, its documentation, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted under the maximum extent possible under the law.

The software installed in accordance with this documentation is copyrighted and licensed by GemTalk Systems under separate license agreement. This software may only be used pursuant to the terms and conditions of such license agreement. Any other use may be a violation of law.

Use, duplication, or disclosure by the Government is subject to restrictions set forth in the Commercial Software - Restricted Rights clause at 52.227-19 of the Federal Acquisitions Regulations (48 CFR 52.227-19) except that the government agency shall not have the right to disclose this software to support service contractors or their subcontractors without the prior written consent of GemTalk Systems.

This software is provided by GemTalk Systems LLC and contributors "as is" and any expressed or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall GemTalk Systems LLC or any contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

COPYRIGHTS

PATENTS

GemStone software has been covered by U.S. Patent Number 6,256,637 "Transactional virtual machine architecture", Patent Number 6,360,219 "Object queues with concurrent updating", Patent Number 6,567,905 "Generational garbage collector with persistent object cache", and Patent Number 6,681,226 "Selective pessimistic locking for a concurrently updateable database".

TRADEMARKS

GemTalk, **GemStone**, **GemBuilder**, **GemConnect**, and the GemStone and GemTalk logos are trademarks or registered trademarks of GemTalk Systems LLC, or of VMware, Inc., previously of GemStone Systems, Inc.

UNIX is a registered trademark of The Open Group.

Intel is a registered trademarks of Intel Corporation.

Microsoft, **Windows**, **Windows Server**, and **Azure** are registered trademarks of Microsoft Corporation.

Linux is a registered trademark of Linus Torvalds and others.
Red Hat, **Red Hat Enterprise Linux**, **RHEL**, and **CentOS** are trademarks or registered trademarks of Red Hat, Inc.
Rocky Linux is a trademark or registered trademark of Rocky Enterprise Software Foundation.
Ubuntu is a registered trademark of Canonical Ltd., Inc.

AIX, **Power**, **POWER**, **Power8**, **Power9**, and **VisualAge** are trademarks or registered trademarks of International Business Machines Corporation.

Apple, **Mac**, m**acOS**, and **Macintosh** are trademarks of Apple Inc.

Instantiations is a registered trademarks of Instantiations, Inc.

CINCOM, **Cincom Smalltalk**, and **VisualWorks** are trademarks or registered trademarks of Cincom Systems, Inc.

Raspberry Pi is a trademark of the Raspberry Pi Foundation.
RabbitMQ is a trademark of VMware, Inc.
Prometheus is a registered trademark of The Linux Foundation.
Grafana is a registered trademark of Raintank, Inc. dba Grafana Labs.

Other company or product names mentioned herein may be trademarks or registered trademarks of their respective owners. Trademark specifications are subject to change without notice. GemTalk Systems cannot attest to the accuracy of all trademark information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

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 3.7.2 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 3.7.2.

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 1. Release Notes for 3.7.2

Chapter 2. Traits

Chapter 3. GCI and FFI Changes

Chapter 4. Bug Fixes

Chapter

1 Release Notes for 3.7.2

Overview

GemStone/S 64 Bit™ 3.7.2 is a new version of the GemStone/S 64 Bit object server. Version 3.7.2 includes feature enhancements, and bug fixes, including significant fixes affecting Gem out of memory issues.

These Release Notes include changes between the previous version of GemStone/S 64 Bit, v3.7.1, and v3.7.2. v3.7.1.1-v3.7.1.4 were limited distribution releases; all changes in these versions are included in these Release Notes. If you are upgrading from a version prior to 3.7.1, review the release notes for each intermediate release to see the full set of changes.

For details about installing GemStone/S 64 Bit 3.7.2 or upgrading from earlier versions of GemStone/S 64 Bit, see the *GemStone/S 64 Bit Installation Guide* for v3.7 for your platform.

Supported Platforms

Platforms for Version 3.7.2

GemStone/S 64 Bit version 3.7.2 is supported on the following platforms:

- Red Hat-compatible Linux 7.9, 8.9, and 9.4, and Ubuntu 20.04, 22.04, and 24.04 on x86_64. GemStone is tested on a mixture of Red Hat, CentOS, and Rocky; these are all considered fully certified platforms. Any reference to Red Hat applies to any Red Hat-compatible distribution.
- Ubuntu 20.04 on ARM. Linux on ARM is for development only, not for production.
- macOS 14.5 (Sonoma) and 11.7.10 (Big Sur), on x86 and Apple silicon (ARM). macOS distributions are for development only, not for production.

Note that GemStone/S 64 Bit v3.7 and later on Linux are built using machine instructions that are not available on older x86_64 CPUs (more than about 10 years old); v3.7.2 will not run on these CPUs, regardless of the Linux OS version.

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.

GemBuilder for Smalltalk (GBS) Versions

The following versions of GBS are supported with GemStone/S 64 Bit version 3.7.2:

GBS/VW version 8.8.1

VisualWorks 9.4 64-bit	VisualWorks 9.3.1 64-bit	VisualWorks 9.1.1 64-bit	VisualWorks 8.3.2 64-bit	VisualWorks 8.3.2 32-bit
▸ Windows 10, Windows 11 ▸ RedHat ES 7.9, 8.9, and 9.4; Ubuntu 20.04, 22.04, and 24.04	▸ Windows 10, Windows 11 ▸ RedHat ES 7.9, 8.9, and 9.4; Ubuntu 20.04, 22.04, and 24.04	▸ Windows 10, Windows 11 ▸ RedHat ES 7.9, 8.9, and 9.4; Ubuntu 20.04, 22.04, and 24.04	▸ Windows 10, Windows 11 ▸ RedHat ES 7.9, 8.9, and 9.4; Ubuntu 20.04, 22.04, and 24.04	▸ Windows 11

GBS/VW version 8.7.1

VisualWorks 9.1.1 64-bit	VisualWorks 9.1.1 32-bit
▸ Windows 10, Windows 11 ▸ RedHat ES 7.9, 8.9, and 9.4; Ubuntu 20.04, 22.04	▸ Windows 10

GBS/VA version 5.4.7

VAST Platform 11.0.1	VAST Platform 10.0.2	VA Smalltalk 8.6.3
▸ Windows Server 2016, Windows 10, and Windows 11	▸ Windows Server 2016, Windows 10, and Windows 11	▸ Windows Server 2016, Windows 10, and Windows 11

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 v3.7.2 distribution includes VSD version 5.6.1. This is the same version of VSD that was included with the previous release, v3.7.1.

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

Rowan

Rowan 2 and Rowan 3 are provided with v3.7.2:

- Rowan 2 is version 2.8, on masterV2.8
- Rowan 3 is version 3.2, on masterV3.2

GemBuilder for Java

You must use the concurrently released GemBuilder for Java (GBJ) v3.2 with GemStone/S 64 Bit v3.7.2; GBJ versions 3.1.3 and earlier are not supported. GBJ v3.2 is provided for Linux only, and does not support clients on other platforms; please contact GemTalk Technical Support if you require other platforms.

There are a number of infrastructure changes in GBJ v3.2; see the *Release Notes for GemBuilder for Java v3.2* for more information.

GBJ installation now requires that the Stone's TimeZone match the OS TimeZone.

The v3.7.2 distribution includes the shared libraries that support GBJ 3.2 on Linux:

```
libgbjlnk320-3.7.2-64.so
libgbjts320-3.7.2-64.so
```

GemConnect

Both GemConnect v2.4 and v2.5 are supported with GemStone/S 64 Bit 3.7.2, on Linux only.

On some recent platforms, there are OS library changes needed:

- In the Ubuntu 24.04 release, the library **libaio.so.1** was replaced with a 64-bit version, **libaio.so.1t64.0.2**. Users on Ubuntu 24 and later should setup a symbolic link to accommodate the name change.
- Installations of RedHat 8.x and 9.x no longer automatically include the library **libnsl.so.1**. Users on RedHat 8 and later should manually install this library.

The GemStone/S 64 Bit v3.7.2 distribution includes the shared library that supports GemConnect v2.4. To use GemConnect v2.5, you may use **make** to link with the 3.7.2 release, or contact GemTalk Technical Support.

Installation and Upgrade

New script to support OS configuration

the **configuregs** script has been added to making basic configuration of simple systems easier; for example, setup of the `/opt/gemstone/locks` and `logs` directories. This avoids the need to use **installgs**, or to manually configure these files.

This script writes answers to an answers file, which can be edited and reused.

New script to setup a simple repository using data-conf

A new script, **createNewGemStoneRepository** has been added, to make setting up a simple Data-conf configured repository simple. A data-conf repository is created with extents and other files in a repository other than $GEMSTONE/data, and uses the **-E** argument and a copy of the `gemstone_data.conf` configuration file.

Installation Guide

Chapter 1 of the Installation Guide for v3.7.2 has been extensively updated.

- OS configuration information has been moved to the end of the chapter, since this is primarily optimizations that do not need to be considering when performing the installation.
- The material regarding creating a new repository has been moved to a separate chapter, Chapter 2.
- Use of the new **configuregs** script has been added, in addition to the **installgs** description, which has been simplified.
- Use of the new **createNewGemStoneRepository** script has been added. The classic way of setting up a basic new GemStone repository in $GEMSTONE/data is also now documented.

Upgrade

GemStone 3.7.2 documentation now includes support for upgrade directly from 3.3.x. This was tested and supported for earlier 3.7.x versions, but not documented.

Improvements/fixes to RPM installation process

The RPM installation process has been reviewed and improved.

Previously, RPM installation was documented to use `rpm -i`. RPM installation more correctly should be done using `dnf` (or `yum` on older OS versions). The previous RPM distribution's package name for GemStone, however, was GemStone64.x86_64. Since this was the same for all GemStone versions, installing a new version using `dnf/yum` would override any earlier RPM versions. (#51189)

With v3.7.2, the RPM package is named GemStone64-3.7.2.x86_64, and `dnf/yum` is safe.

On earlier versions of GemStone, you should continue to use `rpm -i`.

RPM installations could have failed in v3.7 and later with spurious dependency errors.

In addition, on minimal systems in which Perl was not installed, RPM installations could have failed due to Perl dependencies. Now, the RPM installation using `dnf/yum` will install a number of required Perl packages if they are not already present. On earlier GemStone versions, with `rpm -i`, you should include the flag `--nodeps`. (#50965).

On a system in which `/opt/gemstone/locks` and `/opt/gemstone/log` directories were not already defined, the permissions of the newly created directories were not correct. (#50966)

TimeZone installation now supported from the image

The GemStone distribution extent has the TimeZone of America/Los_Angeles, so most new installations will need to update the TimeZone. Previously, this was done by editing and executing the $GEMSTONE/upgrade/installtimezone script.

Now, this can be done from the image, using the new methods. These methods must be executed as SystemUser.

`TimeZone class >> installNamedZone:` *zonenameString*
: Install the named timezone, from the GemStone distribution Olson zoneinfo database in $GEMSTONE/pub/timezone. The new TimeZone is made the default/current. The existing default TimeZone is sent become: with the new TimeZone, so existing references to the old TimeZone automatically now refer to the new TimeZone.

`TimeZone class >> installOsTimeZone`
: Install a TimeZone based on the OS time zone settings for the host on which the Gem executing this command is running. This uses /etc/localtime which is correct for Linux and Mac. The new TimeZone is made the default/current. The existing default TimeZone is sent become: with the new TimeZone, so existing references to the old TimeZone automatically now refer to the new TimeZone.

Change in behavior in 3.7.2 that may affect existing application code

In earlier versions, there was a file descriptor leak in GsHostProcess. This bug has been fixed, which results in an apparent behavior change in `GsSocket >> readWillNotBlock`; see “File descriptor leak in GsHostProcess” on page 60.

The "leaked" file descriptor was the Gem process's writing end of the stdout and stderr pipes, and the read end of the stdin pipe. After the child was forked, the Gem previously failed to close its stdout/sterr write file descriptor, and this socket is inherited by the child. When the child process exited, it closed its own write file descriptor, but the Gem's was still open (although there was no way for data to appear on this socket).

As a result, in previous releases `GsSocket >> read:*` operations from the Gem to the child process, after the child process had exited, hung waiting for data to appear on the still-open socket.

With the fix for #47630, `GsSocket >> read:*` operations from the Gem to the child process after the child process had exited now correctly see the EOF and return 0.

However, `GsSocket >> readWillNotBlock` also now (correctly) returns true in the case when the child has exited. Since the `read:*` operation on this socket will return an EOF without blocking, this is intended behavior, although it was not visible in earlier releases since the socket was not actually closed.

If existing code does not also check for `read:*` returning 0, and only checks the result of `readWillNotBlock`, then it may fail to recognize when the child has exited. Applications that are upgrading should check their application code to ensure the correct loop checks are done when communicating with the child process in GsHostProcess.

In general, `readWillNotBlock` is not a reliable way to detect child status, although it is useful to avoid a `read:*` call waiting indefinitely for data to appear. For instance, there can be a false answer if the child process has not started writing or if it pauses in its writing.

Provided that your code can handle the GsProcess blocking until the child process has sent some data or closed its end of the pipe, there is no need to invoke `readWillNotBlock`.

Changes in this Release

Updated library versions

The version of openssl has been updated to 3.0.15.

The version of libssh has been updated to 0.10.90.

32-bit Linux no longer supported

With v3.7.2, GemStone/S 64 Bit does not support 32-bit Linux. The directories /bin32 and /lib32 are no longer included in the distribution.

Likewise, GBS/VW with 32-bit VisualWorks is no longer supported on Linux.

The Windows client does include 32-bit libraries and GBS is supported with 32-bit client Smalltalk on Windows.

Removing GemStone system locks on reboot

If GemStone server processes are terminated uncleanly, such as a SEGV or an OS shutdown, they may leave their lock files behind in `/opt/gemstone/locks/`. These can normally be cleaned up using **gslist -c**. However, if the OS is rebooted and some process running as a different UNIX user gets the process ID contained in one of the stale .LCK files, then GemStone code cannot safely delete the stale .LCK file.

It is recommended to clear out all .LCK files in /opt/gemstone/locks/ on reboot. An example has been added to demonstrate this:

```
$GEMSTONE/examples/admin/systemd/gemstone-clean-locks.service
```

The example services for the Stone and NetLDI, `gemstone.service` and `netldi.service`, have been updated to use `gemstone-clean-locks.service`.

Gem log header includes additional NRS information

The Gem log header includes a section specific to the Gem's login, in addition to the header information by the NetLDI that spawned it. This now includes an additional line providing the NRS from the Gem's environment, since this may be different from that used by the NetLDI.

Multithreaded scan operations now have a thread limit of 128

Issues were found in which the C code supporting multithreaded scans was not sufficiently protected by mutexes when moving to a new phase of the operation, which could result in unexpected errors or missing results (#50904, #50929). This code has been adjusted to include the appropriate mutexes, and there is now a hard limit of 128 on the number of threads used in a multithreaded scan.

Multithreaded-related query methods no longer require SystemControl

Methods that fetch the current settings for mtMaxThreads, mtThreadsLimit, and mtPercentCpuActiveLimit previously required both SessionAccess and SystemControl privileges; now, only SessionAccess is required. SystemControl is still required to modify these settings.

Improved reclaim behavior with low free space

Previously, the ReclaimGem provided the parameter #reclaimMinFreeSpaceMb, which was designed to suspend reclaim under low free space conditions, to avoid excessive extent file growth before any of the newly reclaimed pages became available. This parameter was not checked at the correct point and was thus ineffective, with the risk of runaway extent growth during reclaim. (#51260, #51258, also on page 57)

However, pausing reclaim has risk of reclaim becoming stuck and the repository remaining in a state of low free space, since reclaim is the way pages are made available again.

To avoid the risk of either stuck reclaim or runaway growth, the Reclaim Gem's parameter #reclaimMinFreeSpaceMb has been removed. Reclaim now is suspended when free space is below the Stone's STN_FREE_SPACE_THRESHOLD. Under low space conditions, reclaim will proceed slowly until pages become available.

The ReclaimGem's SessionStat29, #FreeSpaceLowCount, will report the number of times reclaim was paused due to low free space.

1. Collection and String changes

SequenceableCollection + removed

In 32-Bit GemStone/S before v5.0, `SequenceableCollection>>+` was used to combine collections and strings; this was made obsolete in version 5.0, but has remained in the image to support customers who have progressively upgraded from very old versions.

This method creates issues with Pharo compatibility, and has been removed in this version.

This should not affect customers who have used the deprecation mechanism to remove dependence on deprecated methods. `SequenceableCollection >>+` has been deprecated using GemStone's deprecation mechanism since that mechanism was introduced in v3.0 (2011).

fillFrom:resizeTo:with: no longer usable

The primitive invoked by `Array >> fillFrom:resizeTo:with:` and `KeyValueDictionary >> fillFrom:resizeTo:with:` was subject to out of memory errors (see "Memory issues from fillFrom:resizeTo:with:" on page 55).

The use of this method in KeyValueDictionary table resize was unnecessary; resetting the collection is just as efficient. (#50990). This method has been removed from the image.

`Array >> fillFrom:resizeTo:with:` remains in the image to allow detection using the deprecation mechanism, but is obsolete and signals an error.

The method `Array >> fillFrom:to:with:` has been added, as a replacement for filling an Array. Note that code that previously used `fillFrom:resizeTo:with:` will need modification, since the Array sizing must be done separately.

`Array >> fillFrom:` *startIdx* `to:` *endIdx* `with:` *anObject*
Store anObject into instVars *startIdx..endIdx* of the receiver. The receiver will be grown if necessary. Attempts to grow the receiver beyond 2034 total instance variables (named and unnamed) signals an error.

Double and QuadByteStrings disallow 16rD800..16rDFFF

Characters with codepoints in the range 16rD800..16rDFFF are not valid for Unicode. Characters with these codepoints are disallowed in UnicodeStrings.

Now, they are also disallowed from DoubleByteString and QuadByteString instances, to avoid the problem conversion of DoubleByteString and QuadByteString instances to Unicode16, Unicode32, or a UTF class. Strings containing codePoints in this range are inherently unusable for any string-related purpose.

Improvements to reduced-conflict collections

A number of bugs have been fixed and feature requests implemented, related to RC collections. In addition to the changes listed here, see specific bugs under "Conflict handling in Reduced-conflict collections" on page 56.

Improved handling for heavy reduced-conflict processing

When reduced conflict (RC) collections encounter what would normally be conflicts with other sessions, the conflict is resolved using RC replay. Previously, the Gems sent a set of replay objects to the Stone, which kept a queue of reduced-conflict transactions to resolve (the RcTransQueue). On a busy system, this queue backed up and the application would see commits involving RC collections take longer.

Now, reduced-conflict commit resolution is done by the Gem itself, while it holds the commit token. This ensures that each Gem will succeed in resolving the RC conflict, rather than resolving it only to fail again. The Gems no longer send an RcReadSet to the Stone, and the Stone no longer keeps a queue of reduced-conflict transactions to resolve.

RcKeyValueNoRebuildDictionary added class

This is a variant of RcKeyValueDictionary that does not automatically execute `rebuildTable:`. This allows more concurrency in the RC replay of additions. You must ensure that the table size is adequate for the number of key/value pairs.

Results of RcIdentityBag >> maxSessionId changed

The method `RcIdentityBag >> maxSessionId` previously returned the size of the internal components structure. Since there are two entries per sessionId, this was incorrect. `maxSessionId` now returns the highest sessionId represented in the internal structure.

RcIdentityBag added methods

The following methods have been added to RcIdentityBag:

`RcIdentityBag >> maxSessionId:` *aSessionId*
: Force the receiver to include component bags for Ids up to *aSessionId*. This method is not reduced conflict and should not be used while other sessions are modifying the collection.

`RcIdentityBag >> removeOneObject`
: Remove and return one object from the receiver. An object that was added by the current session is returned, if possible, otherwise an object added by another session, otherwise nil if the receiver is empty.

Non-identity-dictionary >> keys now returns appropriate class of result

Sending #keys to instances of following classes previously returned instances of IdentitySet, although the dictionary is not identity-based. For these classes, sending #keys now returns an instance of Set:

```
KeyValueDictionary
IntegerKeyValueDictionary
KeySoftValueDictionary
StringKeyValueDictionary
RcKeyValueDictionary
```

Optimizations and bug fixes in TreeDictionary/TreeSet

TreeDictionary/TreeSet are kinds of Dictionary/Set optimized to avoid performance variability in adding elements. These classes were added in v3.7.1. There have been a number of optimizations in the supporting code in v3.7.2. In addition, the following bugs have been fixed:

TreeSet audit returned false following an includesIdentical:

If you execute `includesIdentical:` on a TreeSet, and immediately perform an audit, the audit failed. This is a false result, and does not indicate a problem. (#50921)

TreeSet detect: and detect:ifNone: failed

The methods `TreeSet >> detect:` and `TreeSet >> detect:ifNone:` errored. (#51150)

2. SSH and Socket Changes and Bug fixes

Support for OpenSSH keys and conversion between key types

To support OpenSSH keys, and allow conversion between RSA and PEM format keys, the classes GsSshPublicKey and GsSshPrivateKey have been added.

GsSshPrivateKey

GsSshPrivateKey encapsulates an OpenSSH private key.

The following instance creation methods create new instances of the receiver based on an OpenSSH base64 string, or a file containing an OpenSSH base64 string, and optional a passPhrase string or a file containing a passPhrase.

The following OpenSSH key types are supported: ecdsa, ed25519, and rsa (dsa keys are no longer supported by libssh).

```
GsSshPrivateKey class >> newFromOpenSshString: aBase64String
   withPassphraseFile: aPfFile

GsSshPrivateKey class >> newFromOpenSshString: aBase64String
   withPassphrase: aPf

GsSshPrivateKey class >> newFromOpenSshFile: fileNameString
   withPassphraseFile: aPfFile

GsSshPrivateKey class >> newFromOpenSshFile: fileNameString
   withPassphrase: aPf

GsSshPrivateKey class >> newFromOpenSshFile: fileNameString

GsSshPrivateKey class >> newFromOpenSshString: aBase64String
```

GsSshPublicKey

GsSshPublicKey encapsulates an OpenSSH public key.

The following instance creation methods create new instances of the receiver based on an OpenSSH base64 string, or a file containing an OpenSSH base64 string.

The following OpenSSH key types are supported: ecdsa, ed25519, and rsa (dsa keys are no longer supported by libssh).

```
GsSshPublicKey class >> newFromOpenSshFile: fileNameString

GsSshPublicKey class >> newFromOpenSshString: aBase64String
```

Conversion between types

The following methods have been added for use by concrete subclasses (GsTlsPrivateKey, GsTlsPublicKey, GsSshPrivateKey, GsSshPublicKey, and Gs5509Certificate). Note that libssh does not support importing public keys from OpenSSL PEM format, so conversions are not supported for GsTlsPublicKey.

```
GsTlsCredential >> asOpenSshKey
```
Return an GsSsh*Key, either public or private to match the class of the receiver. Disallowed for GsTlsPublicKey.

`GsTlsCredential >> asOpenSshString`
Returns a String representing the receiver in OpenSSH base 64 format, either public or private depending on the class of the receiver. For private keys, base64 text lines are limited to 70 characters. Disallowed for GsTlsPublicKey.

`GsTlsCredential >> asOpenSshStringOneLine`
Returns a String representing the receiver in OpenSSH base 64 format, either public or private depending on the class of the receiver. Base64 text is placed on a single line. Disallowed for GsTlsPublicKey.

`GsTlsCredential >> asOpenSslKey`
Return an GsTls*Key, either public or private to match the class of the receiver.

Added Testing methods

The following added methods are supported by all kinds of GsTlsCredential

Encryption type:

```
GsTlsCredential >> isDsa
GsTlsCredential >> isEllipticCurve
GsTlsCredential >> isRsa
```

Key type:

```
GsTlsCredential >> isOpenSshKey
GsTlsCredential >> isOpenSslKey
GsTlsCredential >> class isOpenSshClass
GsTlsCredential >> class isOpenSslClass
```

Examples

The following examples and support methods have been added:

`GsSshSocket class >> exampleOpenSshPrivateKey`
Return an instance of GsSshPrivateKey that may be used to access the ssh test server at ssh-test.gemtalksystems.com.

`GsSshSocket class >> examplePrivateKeyAsOpenSshString`
Private key for the ssh test server at ssh-test.gemtalksystems.com in OpenSSH format.

`GsSshSocket class >> nbSshClientExample3`
Example of remote ssh command using non-blocking protocol with PKI authentication.

`GsSshSocket class >> sshClientExample3`
Example of an remote ssh command using blocking protocol with PKI authentication.

Validating an SNI name against the target host name

GsSecureSocket now supports validating that a client connection's SNI (Server Name Indication) name matches a host name in the peer certificate.

By default, this additional validation is **not** done; the same validation is done as in previous releases.

To enable validation, add the expected host name to the GsSecureSocket before making the secure connection, using `GsSecureSocket >> addExpectedHost:` or `GsSecureSocket >> setExpectedHost:`. Multiple hosts can be added to a Socket's expected hosts.

During the connection TLS handshake, if the certificate's names don't match the expected host name, the connection is not completed. This validation is done automatically during the connection, if the expected hosts is set.

To disable validation, set the expected hosts to nil using `GsSecureSocket >> setExpectedHosts: nil.`

The matching can be made more strict by setting additional flags using `GsSecureSocket >> setExpectedHostFlags:.`

GsSecureSocket added methods

`GsSecureSocket >> addExpectedHost:` *aString*
Adds *aString* to the list of expected host names the receiver will connect to. This method must be executed before the #secureConnect method and only with client sockets. Raises an exception if the receiver is not a client socket, #secureConnect has already been executed, or if *aString* is not a non-empty instance of String.

`GsSecureSocket >> setExpectedHost:` *aString*
Sets the expected host name that the receiver will connect to. If *aString* is nil, all previously set host names will be cleared and no host name matching will be performed. This method must be executed before the #secureConnect method and only with client sockets. Raises an exception if the receiver is not a client socket, #secureConnect has already been executed, or if aString is not a non-empty instance of String or nil.

`GsSecureSocket >> setExpectedHostFlags:` *anArray*
Sets the flags which control host name matching during TLS session negotiation. By default no flags are set. *anArray* must contain zero or more symbols of the following symbols:

```
#X509_CHECK_FLAG_ALWAYS_CHECK_SUBJECT
#X509_CHECK_FLAG_NO_WILDCARDS
#X509_CHECK_FLAG_NO_PARTIAL_WILDCARDS
#X509_CHECK_FLAG_MULTI_LABEL_WILDCARDS
#X509_CHECK_FLAG_SINGLE_LABEL_SUBDOMAINS
#X509_CHECK_FLAG_NEVER_CHECK_SUBJECT
```

An empty array causes all previously set flags to be cleared.

A detailed description of these flags may be found in the OpenSSL documentation at: https://www.openssl.org/docs/man3.0/man3/X509_check_host.html

Other added methods for managing certificates

The following methods on GsSecureSocket and `GsX509Certificate` allow you to query for more details about the peer certificate.

`GsSecureSocket >> matchedPeerName`
Returns a String representing the DNS hostname or subject CommonName from the peer certificate that matched one of the hosts set by the #setExpectedHost: or #addExpectedHost: methods. Returns nil if no host matching was performed. Raises an exception if the receiver is not a client socket or if the secure connection has not been established.

`GsSecureSocket >> peerCertificate`
Answer an instance of GsX509Certificate representing the peer's certificate. Raises an exception if the receiver has not completed the TLS handshake with its peer. Returns nil if no certificate was sent by the peer. This will always happen if anonymous TLS is used and can happen when certificate verification is disabled.

`GsSecureSocket >> peerCertificateChain`
Answer an instance of GsX509CertificateChain containing the peer's certificate chain. Raises an exception if the receiver has not completed the TLS handshake with its peer. Returns nil if no certificate was sent by the peer. This always happens if anonymous TLS is used and may happen when certificate verification is disabled.

`GsX509Certificate >> isSelfSigned`
Answer a Boolean indicating if the receiver is a self-signed certificate.

`GsX509Certificate >> issuerName`
Returns a string representing the issuer common name of the receiver.

`GsX509Certificate >> isValidNow`
Answer a Boolean indicating if the receiver is valid at this point in time, that is the current time falls within the window between the receiver's 'not before' and 'not after' times.

`GsX509Certificate >> notAfterTime`
Returns a SmallDateAndTime representing the 'not after' time of the receiver in GMT.

`GsX509Certificate >> notAfterTimeGmtSeconds`
Returns a SmallInteger representing 'not after' time of the receiver expressed as the number of seconds since 00:00:00UTC January 1, 1970.

`GsX509Certificate >> notBeforeTime`
Returns a SmallDateAndTime representing the 'not before' time of the receiver in GMT. GsX509Certificate >> notBeforeTimeGmtSeconds. Returns a SmallInteger representing 'not before' time of the receiver expressed as the number of seconds since 00:00:00UTC January 1, 1970.

`GsX509Certificate >> notBeforeTimeGmtSeconds`
Returns a SmallInteger representing 'not before' time of the receiver expressed as the number of seconds since 00:00:00UTC January 1, 1970.

`GsX509Certificate >> subjectAlternateNames`
Returns an Array of Strings representing the contents of the subject alternate extension contained in the receiver, or an empty array if the receiver does not contain the extension.

`GsX509Certificate >> subjectName`
Returns a string representing the subject common name of the receiver.

GsSftpSockets may not be nonblocking

There is a known bug in libssh, #58: Opening sftp fails in non-blocking mode, that prevents non-blocking GsSftpSockets from being used. (#50942)

The following methods are not supported for GsSftpSocket:

```
makeNonBlocking
nbSshConnect
nbSshConnectTimeout:
```

GsSecureSocket examples

Now working on Darwin

`GsSecureSocket >> setCaCertLocation` now includes checking the trusted CA cert locations on the Mac, which allows the GsSecureSocket examples to work on Mac.

Added Non-blocking examples

The following examples have been added:

`GsSecureSocket class >> httpsClientExampleForHost:` *hostName* `withSniName:` *sniName* `blocking:` *bool*
Connect to an https web server on port 443 at the given host and perform a simple GET request. Full verification of the server certificate is performed. Self-signed certificates will fail verification.

`GsSecureSocket class >> httpsClientExampleNB`
Connect to the google https web server on port 443 in nonblocking mode and perform a simple GET request. Full verification of the server certificate is performed. Returns true on success.

`GsSecureSocket class >> httpsSniClientExampleNB`
Example to connect to a webserver that requires SNI in nonblocking mode. Not using SNI (server name == nil) will fail to connect. Returns true on success.

New debugging environment variable

GS_DEBUG_SSL_SHUTDOWN_DIR

Setting this environment to a directory will write a debug log file for all SSL_shutdown() operations.

GsSecureSocket secureAccept may fail with SSL error

During `secureAccept`, GsSecureSocket calls `readWillNotBlockWithin:`, which calls `_peek`, which calls `SSL_peek()`. This call would fail with an error if the socket was not yet connected, depending on the timing. (#50937)

3. Other image changes

GsHostProcess with Arguments

Using GsHostProcess to execute C code with arguments that include whitespace and quotes has Smalltalk-specific requirements for escaping/formatting. Hand applying formatting is error-prone; and including unmatched single quotes was not possible.

Now, arguments can be supplied as an array of strings to an additional argument. The following methods have been added:

`GsHostProcess class >> execute:` *commandLineString* `args:` *anArray*
A variant of the existing `GsHostProcess class >> execute:` method with the added `args:` argument.

`GsHostProcess class >> execute:` *commandLineString* `input:` *stdinString* `args:` *anArray*
A variant of the existing `GsHostProcess class >> execute:input:` method with the added `args:` argument.

`GsHostProcess class >> fork:` *commandLineString* `args:` *anArray*
A variant of the existing `GsHostProcess class >> fork:` method with the added `args:` argument.

In these methods, the *commandLineString* is parsed for space separated arguments. *anArray* may be nil or an Array of instances of String, Unicode7 or Utf8.

Elements of *anArray* are appended to the arguments contained in *commandLineString*, to form the total argv array of the child; any quoting or whitespace within an element of *anArray* is passed directly to the child in the corresponding element of child's argv.

When using GsHostProcess, if any argument has complicated quoting or whitespace, it is recommended that *commandLineString* should contain only the full path to the executable, and arguments should be passed as elements of *anArray*.

Example

The following examples show the existing way to specify a filename containing a space, along with the new method using the `args:` keyword.

```
GsHostProcess execute: '/bin/cat ''a b.txt'''
GsHostProcess execute: '/bin/cat' args: {'a b.txt'}
```

Support for warming AllSymbols

When doing limited cache warming on repository startup, you may now also manually warm AllSymbols, using the new methods:

`System class >> warmAllSymbols:`
`System class >> warmAllSymbols:` *numSessions* `parameters:`
aGemStoneParameters

These functions start an external session to perform the warming. Note that `warmAllSymbols:` takes its login parameters from the current session. This is not sufficient to login if the current user's password is not swordfish or the NetLDI is performing authentication.

In most cases you should use `warmAllSymbols:parameters:`, which allows you to pass in an instance of GemStoneParameters, either based on the current session or using other parameters.

For example:

```
| params |
params := GemStoneParameters newDefault.
params
   password: 'currentGemStoneUserPassword';
   hostPassword: 'myHostPassword';
   yourself.
System warmAllSymbols: 5 parameters: params.
```

Changes in commit conflict reports

Changes in conflict types

Due to changes in commit conflict handling (as described on page 19), the conflict type #Rc-Write-Write has been replaced by #Rc-Retry-Failure.

The commit failure type #RcReadSet has been added, containing the RcReadSet contents to assist in debugging.

Additional information in commit conflict reports

The conflict details returned from `System class >> commitConflicts`, `detailedConflictReportString` and `transactionConflicts` have been updated to include timestamps.

The method `System class >> _gemCommitConflictDetails` (and the primitive it invokes) now returns a SmallInteger rather than a boolean.

AllSymbols audit

The method `CanonSymbolDict >> _audit` has been added, providing a way to audit AllSymbols.

System class

Optimization to System use of HiddenSets

Support for legacy System hiddenSets is done by sending messages to GsBitmap. The hiddenSet ids are different between System and GsBitmap. Previously, this required several lookups to produce the desired GsBitmap.

System now includes a number of class variables, which reference the GsBitmap ids, and methods that previously hardcoded the set by symbolicName now reference the class variable.

This is transparent to the user.

Added methods for configuration information

Methods have been added to provide the configuration values for the Gem and Stone in a string format:

`System class >> gemConfigurationReportString`
Returns a String representation of gemConfigurationReport.

`System class >> stoneConfigurationReportString`
Returns a String representation of stoneConfigurationReport.

Other added methods

Several methods have been added to support split tranlogs; these are listed on page 33.

This convenience method has also been added:

`System class >> pagesNeedReclaimCount`
Return number of pages not yet reclaimed by the reclaimGem.

Other Changes

`System >> currentSessionsReport`, and other session report methods invoking `_sessionsReport:`, now include an identification of the session that is executing the query.

External Session changes

Now supporting automatic conversion of Unicode7 return values

External session executions that result in kinds of CharacterCollection return the contents to the calling session in ByteArrays encoded as UTF-16. This is converted to the appropriate class, if possible, by the method `resolveResult`.

Previously, results that were instances of Unicode7 were returned as ByteArrays. Now, these are resolved, and an instance of Unicode7 is returned. This applies to both GsExternalSession and GsTsExternalSession

Recommendation to encode returned strings

Instances of Unicode16 or Unicode32 are still returned as ByteArrays, as well as DoubleByteStrings and QuadByteStrings. If your application uses strings containing characters out of the ASCII range, it is recommended to use `encodeAsUTF8` in the code executing on the external session, and returning the encoded string.

GsExternalSession automatic conversion of Utf8 to Strings

Previously, GsExternalSession returned ByteArrays for return types of Utf8. Now, these are automatically decoded to kinds of String. This applies to all byte sizes of strings. This is the same behavior as GsTsExternalSession has had in previous releases.

Appropriate class for decoded Utf8 results

When an instance of Utf8 is returned, it was previously always returning a Unicode7, Unicode16 or Unicode32. Now, it will return the appropriate class, either a legacy String or Unicode string, depending on the StringConfiguration of the calling session's repository.

forkAndDetachBlock:/String: no longer terminates with apparant error

A non-blocking external session is created using `GsTsExternalSession>> forkAndDetachBlock:` or `forkAndDetachBlock:`. When this session logs out, it signals error 4137, which is a normal end of session for a non-blocking session. This was presented in the gem's log file such that it looked like a gem error. The message has been updated to include "The session is terminating normally". (#51264)

Added methods

The following methods have been added:

`GsTsExternalSession >> parameters:` *aGemStoneParameters*
Set the external session's parameters instance variable to *aGemStoneParameters*.

`GemStoneParameters >> stoneName`
Alternate accessor method for the instance variable gemStoneName.

`GemStoneParameters >> stoneName:` *aName*
Alternate settor method for the instance variable gemStoneName.

`GemStoneParameters class >> newDefault`
Create a new instance of GemStoneParameters based on the current session, with the default password 'swordfish' and no default host password.

`GemStoneX509Parameters >> addGemArg:` *aString*
Add a gem argument to extraGemArgs without deleting an existing value in extraGemArgs.

Issues in GsNetworkResourceString defaultGemNRSFromCurrent handling of NetLDI

If GEMSTONE_NRS_ALL includes a #dir or #log directive, this was not included in `GsNetworkResourceString defaultGemNRSFromCurrent`. (#50954)

When using a NetLDI that does not have a name in the services database, and GEMSTONE_NRS_ALL is not defined, `GsNetworkResourceString defaultGemNRSFromCurrent` parsed incorrectly, and included a trailing ! in the NetLDI name (which may be the port number). This causes logins using that NRS (e.g. with GsTsExternalSession) to fail. (#50901)

Other Added methods

UserProfile added method

`UserProfile >> loginHook`
If a method or block has been installed using loginHook:, return that; otherwise, return nil.

SessionTemps added method

`SessionTemps >> increment:` *aSymbol*
Increment the value at the given symbol by 1. If aSymbol not found as a key assume previous value was zero. Returns the SmallInteger value after increment.

SymbolList added method

`SymbolList >> dictionariesAndAssociationsOf:` *aSymbol*
For all of the Associations in the enumeration of the receiver for which key == aSymbol. Returns nil or an Array of the form { { dictionary . association } ... }.

GsTestCase logging changes

GsTestCase logging now reports stacks for assertion failures more similarly to the way errors are reported. Instead of just `FAILURE:`, the leading line now will report:

```
ERROR 2753  a ResumableTestFailure occurred (error 2753)
```

`GsTestCase >> assert:identical:` has improved printing when SmallInteger arguments do not match.

JsonParser parsing of exponential notation

Previously, JsonParser parsed expressions using scientific notation, such as '1E-1', into instance of Fraction. Now, instances of SmallDouble or Float are returned.

Changes in Number>>kind for some classes

A fraction of 0/0 previously reported kind as infinity, now this is a NaN.

FixedPoint previously always returned kind as normal; now it will return NaN, infinity, and zero if appropriate.

Logging to GCI server rather than client

There were remaining maintenance methods that logged output to the client rather than the server. To allow these to execute using the thread-safe GCI, such as via GsTsExternalSession, these have been changed to log output to the server. Logging to client or server is the same in linked sessions, but in a topaz RPC sessions, output now goes to the Gem log. The following methods are affected:

```
GsNMethod class>>convertArrayBuildersIn:list:
GsNMethod class>>convertArrayBuildersInTopazScript:
GsNMethod class>>_convertArrayBuildersIn:list:refDir:
Repository>>_shrinkExtents
System class>>startCheckpointSync
System class>>_commitPrintingDiagnostics
Upgrade3Init class >> _allUsers_addInversePrivilege
```

Newly deprecated methods

Array >> fillFrom:resizeTo:with:

While this method is marked deprecated, it is no longer supported and will raise an error.

System class >> cancelWaitForDebug

This method does not need to be used manually; the relevant topaz commands automatically perform the same function.

Removed methods

Repository >> listInstances:toDirectory:numThreads:objMaxSize:

This method was not functional, and has been removed.

SequenceableCollection >> +

For details, see"SequenceableCollection + removed" on page 18.

Removed private, internal, and refactored methods

```
DecimalFloat >> _asStringE
GciLibrary >> GciHostInstallFaultHandler_:_:
GemStoneParameters >> _getArg:key:
GemStoneParameters >> _secureArg:key:
GemStoneParameters >> _securePasswords
GemStoneParameters >> _zeroArgPrim:
GsTestCase >> repeatIfInterruptedByGC:
GsTestCase >> shouldRepeatIfInterruptedByGC
GsTlsCredential class >> newFromPemString:
HtInternalNode >> makeRoomAtIndex:
KeyValueDictionary >> fillFrom:resizeTo:with:
LogEntry >> argArray:
LogEntry >> receiver:
LogEntry >> selector:
Object >> _primitiveSelectiveAbort
PrivateObject >> _primitiveSelectiveAbort
RcCounter >> _getInvalidSessionIds
RcCounter >> _getSessionElementFor:
RcIdentityBag >> _components:
RcIdentityBag >> _indexForSessionId:
RcIdentityBag >> _privateRemove:fromRemovalBag:logging:updateRc
   ValueCache:
RcIdentityBag >> _privateRemove:logging:updateRcValueCache:
RcIdentityBag >> _processRemovalBagFor:logging:
RcIdentityBag >> _thisSessionAdditionIndex
RcKeyValueDictionary >> _rebuildTable:logging:
RcKeyValueDictionary >> _removeKey:logging:
RcQueueSessionComponent class >> speciesOfElements
RedoLog >> addLogEntry:forConflictObject:
RedoLog >> addLogEntry:forLargeConflictObject:
RedoLog >> clear
System class >> _cancelWaitForDebugExitClient
System class >> _removeFromRcQueue
TimeZone class >> fromSolaris
TimeZone class >> _initialize1
TimeZone class >> _initialize2
```

4. Split Tranlogs Feature

The split tranlogs features allows tranlog records to be copied, as they are written, into smaller chunks whose size is determined by a timeout. The many chunk files that are written during the period that a single ordinary tranlog is written by the Stone, can be accessed or restored without combining, and many GemStone code and utilities treat these as the equivalent of an ordinary tranlog. They can be combined into a matching ordinary tranlog using **copydbf** or the unix **cat** command.

Both logsender and logreceiver support split tranlogs.

The logsender performs the copy-and-split operation. This can be done outside of a hot standby system to produce WORM-level security, reducing the interval in which the tranlog file is writable. The logsender performing the copy-and-split operation cannot also have a connection to the logreceiver to support a hot standby system. It is allowed to connect both a split-tranlogs logsender and a hot-standby logsender to the same stone, so both functions can be supported.

The Stone's behavior when writing tranlogs is unaffected. The Stone will write its ordinary tranlogs as before.

Split-tranlog support for the logreceiver is quite different. The logreceiver receives tranlog records from the logsender and write these to the local disk. With split tranlogs enabled, the logreceiver writes these tranlogs as split tranlogs, rather than full tranlog files.

startlogsender has new arguments to allow specification of a timeout and the target directory for write. The logsender performing tranlog splits does not perform normal hot standby functions; the **-P** argument is used only for **stoplogsender**, not for listening for connections from a logreceiver.

startlogreceiver has a new argument to allow specification of a timeout. When this is used, the tranlogs received from the logsender and written by the logreceiver on the slave system, are written to the **-T** directories as split tranlogs,. These split tranlogs are replayed by continuous restore in the same way as ordinary tranlogs.

startlogsender

startlogsender has the following additional arguments:

```
-W <path>  path to write-once filesystem for writing
      tranlogX.dbf.sN split tranlogs. Requires -F.
-F <writeTimeoutSecs>
      Write tranlogs as split tranlogs to the path specified by
      -W. <writeTimeoutSecs> is the allowed time for holding a
      tranlogX.dbf.sN open before closing that file and
      opening tranlogX.dbf.sM where M equals N + 1. Only used
      for split tranlogs. Requires -D.
```

When **-F** and **-W** are used, **-A** is disallowed, as are the SSL options **-C -J -K -Q -S -V**.

The logsender can be stopped and restarted, and the writing of records to split tranlogs will seamlessly resume where the previous logsender stopped.

startlogreceiver

startlogreceiver has the following additional argument:

```
-F <writeTimeoutSecs>
      Write tranlogs as split tranlogs to the directory
      specified by -T. <writeTimeoutSecs> is the allowed time
      for holding a tranlogX.dbf.sN open before closing that
      file and opening tranlogX.dbf.sM where M equals N + 1.
      Only used for split tranlogs. Requires -D.
```

There are no argument restrictions when using **-F**.

Tranlog chunks

The tranlog chunks are named `tranlog`*TLNumber*`.dbf.s`*chunkID*, where the first chunkID is 0001. The chunk ID is not limited to 4 digits; 0-padding is for convenience for testing using **cat**, etc. With a 30 second write timeout, 4 digits provides enough chunk ids for a single tranlog to be copied as split tranlogs for more than 80 hours before requiring 5 digits. The new chunk file is not actually created until there are transactions ready to be written, so an idle system will not continuously create new chunk files.

Once the timeout has expired and the logsender is ready to start a new chunk file, the previous tranlog is fsynced to disk and made user read-only, with permission 400.

If a write, fsync, or chmod on a chunk file returns an error, the logsender will exit.

Image support

Added System methods to control split tranlogs logsender process:

`System class >> killLogSenderSplitLogs`
: Causes stone to send SIGTERM to the log sender process that was started with the arguments -F -W.

`System class >> logsenderSessionIdSplitLogs`
: Return the session ID of the logsender process started with -W -F, or zero if no logsender is connected to the stone process. Note a logsender can exist without a connection to the stone.

Example

For example, to start writing split tranlogs, with chunks written every 30 seconds to a directory named */gshost/splitTranlogs/*, use an expression such as the following:

`startlogsender -F 30 -W` */gshost/splitTranlogs/* `-P 54321 -s` *myStoneName*

If this runs for 90 seconds in an active system, it would produce the following split tranlog chunk files, in addition to writing tranlog1.dbf in the normal tranlog location:

/gshost/splitTranlogs/`tranlog1.dbf.s0001`
/gshost/splitTranlogs/`tranlog1.dbf.s0002`
/gshost/splitTranlogs/`tranlog1.dbf.s0003`

Copydbf

Chunk files are only opened read-only.

Chunk files can be used directly for **copydbf** or restore, by specifying the first tranlog chunk file (`tranlog1.dbf.s0001`). This will look for all tranlogs chunk files tranlog1.dbf.s0001...tranlog1.dbf.sN until a file N+1 is not found.

For example,

```
copydbf -i tranlog1.dbf.s0001
```

will report the status of all `tranlog1.dbf.s*` files.

You can use **copydbf** to create a merged file, which is the same as the normal tranlog1.dbf written by the Stone. The copydbf argument of the first chunk file (`tranlog1.dbf.s0001`) will automatically pick up the remaining chunk files.

For example:

```
copydbf tranlog1.dbf.s0001 tranlog1.copy.dbf
```

Creates a merged file containing all the `tranlog1.dbf.s*` files (with monotonically increasing numbers). If the stone's ordinary tranlog is named `tranlog1.dbf`, then:

```
diff tranlog1.copy.dbf tranlog1.dbf
```

returns a status of zero.

File-based concatenation

Chunk files can also be concatenated using **cat**, which produces a file that is identical to the ordinary tranlog that was written by the Stone.

For example,

```
cat tranlog1.dbf.s* > tranlog1.mergedChunks.dbf
```

This requires that all chunks be part of the monotonically increasing set; missing files in the sequence are not detected.

Also note that if more than 4 digits have been used when writing files (that is, a file `tranlog1.dbf.s10000` was created), any scripts using **cat** would need to generate the proper sequence of input files, to ensure the files are in the correct order.

Restore from Tranlogs

When tranlogs are opened for restore, it will look for s tranlog named tranlog1.dbf.s0001 as well as the other supported filename patterns (tranlog1.dbf, tranlog1.dbf.lz4, and tranlog1.dbf.gz). If tranlog1.dbf.s0001 is found, it is opened, read-only, as a split tranlog and all the chunk files tranlog1.dbf.s0001... tranlog1.dbf.sN until a file N+1 is not found, are restored.

Once all the chunk files that composed the tranlog are restored, it will continue restoring the next tranlog (tranlog2), which may be in any of the four supported filename patterns.

Other access

Low-level pgsvr tranlog access can read split tranlogs without combining.

However, **printlogs** and **searchlogs** do **not** support access of split tranlogs. To use these utilities, merge the split tranlogs using **copydbf** or file-based concatenation.

5. Changes in Configuration Parameters

Added configuration parameter

The following has been added, to avoid out of memory conditions in which the code_gen region size is the limiting factor.

GEM_TEMPOBJ_CODE_SIZE

Specifies the maximum size of methods area of code_gen in K bytes.

Default units: KB. Suffixes KB, MB, GB may be used to specify units.
Default 0, which translates to 20% of GEM_TEMPOBJ_CACHE_SIZE up to a max of 150MB.
Minimum: 2000KB
Maximum: 600MB. On Linux Arm, the max is 40MB.

GemCommitConflictDetails now accepts 0...3

The gem configuration parameter GemCommitConflictDetails previously accepted true or false as an argument.

Now, it is a numeric value between 0 and 3 (inclusive). 1 and true are equivalent and 0 and false are equivalent.

Values of 2 and 3 provide additional tracing of RC replay.

Cache warming for X509-secured GemStone

The argument string supplied to the X509-Secured GemStone-only parameter NETLDI_WARMER_ARGS now accepts an additional argument to support cache warming:

-w *integer* -- interval in minutes for shrpcmonitor to write a working set file

6. Utility changes

stopstone changes

Previously, if the Stone took a long time to shutdown, **stopstone** could return before the stone was completely shut down and the shared memory segment released, which meant that a subsequent **startstone** failed.

In v3.7.2, the shutdown sequence has been redesigned to avoid this situation. **stopstone** will block until the shared memory segment is released and **startstone** will be able to succeed. The Stone itself will now shut down before the Shared Page Cache Monitor. The stone and SPCMon log file messages reflect this new sequence.

The **stopstone -t** *secs* argument allows you to specify a timeout. By default (if **-t** is not specified), this is -1, and **stopstone** will wait forever for the Stone to stop and the shared memory segment to be available. Previously, **stopstone** errored with an explicit **-t -1** argument.

`System >> shutDown` also blocks until shutdown is complete.

startnetldi further checking for consistent port and numeric name

In recent releases, **startnetldi** has tightened checking to ensure that a setting in $GEMSTONE_NRS_ALL matches the specified netldi server name.

Now, **startnetldi** also checks the **-P** and **-X** arguments, and any mappings in the services database.

- If the services database maps a alphabetic-named netldi to a port, this port number must now match any **-P** argument, **startnetldi** numeric netldi name, or a numeric-named #netldi specification in $GEMSTONE_NRS_ALL or an argument passed with **-X**.
- When the startnetldi **-X** argument is used, a netldi name, or numeric name signifying port use must agree with a numeric netldi name provided as a **startnetldi** argument. Since the **-X** argument overrides a $GEMSTONE_NRS_ALL, this does not need to match.
- A port number supplied to **-P** must match a numeric netldi name provided as a **startnetldi** argument, or a numeric-named #netldi specification in $GEMSTONE_NRS_ALL or an argument passed with **-X**.

Utility failures have additional [ERROR] result

When utilities such as **startstone** and *startnetldi* have argument errors or other startup issues, they report an error message. A number of cases in which the error message was not preceded by [ERROR] or Error: now correctly include this prefix.

stoned command requires -l argument

The **stoned** command is normally invoked by **startstone**, however it is legal to invoke it directly. In 3.7.x, the `-l` argument to **stoned** is required.

In earlier v3.7 versions, the stoned utility SEGVed in absence of **-l** argument; now, it errors and the stone does not start. (#51144)

startstone, startnetldi only print full usage if -h is specified

If an error occurs with a GemStone utility, usually the full usage is printed after the error message. For some utilities this is long, obscuring the actual error. Now, **startstone** and **startnetldi** will print only the usage lines, not the option details, if an error occurs. To see full details, use the **-h** argument.

Topaz Changes

NETLDI: command now usable for standard logins

When using a NetLDI with a name other than gs64ldi, it was previous necessary either to define the GEMSTONE_NRS_ALL, or use a gemnetid login parameter NRS that included the netldi: NRS directive, for example:

```
set gemnetid !@hostx#netldi:12345!gemnetobject
```

Now, the NetLDI name or port can be provided using the **netldi** topaz command. The **netldi** topaz command was previously restricted to use with X509 secured GemStone; now it is usable with standard logins, with different semantics. Note that it is an error to specify a **netldi** command argument that disagrees with a netldi: NRS argument in the environment variable GEMSTONE_NRS_ALL.

This simplifies the required fields for login when not using the default-named netldi. For example, without setting GEMSTONE_NRS_ALL, for a local login, parameters such as the following were required:

```
set stone myStoneName user DataCurator
set gemnetid !#netldi:12345!gemnetobject
```

Which can now be replaced by:

```
topaz> set stone myStoneName user DataCurator netldi 54321
```

For login from a remote node, parameters such as the following:

```
set stone myStoneName user DataCurator
set gemnetid !@hostx#netldi:12345!gemnetobject
```

Can be replaced by:

```
topaz> set stone myStoneName user DataCurator netldi 54321
topaz> set gemnet !@hostx!gemnetobject
```

Trait support functions added

Many functions have been added to support working with traits in topaz; see “Topaz support for traits” on page 43.

Added DISPLAY, OMIT subcommand ORIGIN

DISPLAY ORIGIN

DISPLAY ORIGIN causes the origin of a method that was added from a trait, or loaded from Rowan, to be displayed after the method source when a method is listed. DISPLAY ORIGIN is true by default.

The origin of a method may be an ordinary GemStone class, a trait, or rowan. Where the origin of a method is a class, then no additional information is displayed. If the origin is a trait, the name of the trait is displayed. If the origin is rowan, the name of the Rowan project and Rowan package are displayed.

For example:

```
topaz 1> list method addToLog:
addToLog: aString
resultToReport isNil ifTrue: [resultToReport := String new].
resultToReport add: aString.
---ORIGIN: Trait method #'LoggingTrait'---
```

OMIT ORIGIN

OMIT ORIGIN disables the display of origin for trait and rowan methods when listing methods.

Change in LIST, LIST METHOD:

When displaying a method, if DISPLAY ORIGIN is true (the default), and the origin of the method is a trait or rowan, then the name of the trait, or the Rowan package name of the method, will be listed along with the method source.

Change in TFILE

Now, when reading a class.st file, class creation will be attempted if the class does not exist. If a new version of the class would be created, an Error is signalled.

Chapter

2 Traits

Traits provide a way to share sets of behavior (instance and class methods) between classes, other than using inheritance from a shared superclass. This provides a result similar to multiple inheritance.

Note that traits area new feature in v3.7.2 and should be considered preview. There are known conditions that are not handled optimally, such as creating multiple traits with the same name. Current versions of GemBuilder for Smalltalk (GBS) do not provide support for Traits.

It is not recommended to use traits when doing development in an environment that is not traits-aware, such as GBS 8.8.1 and earlier versions. Methods that originated from trait methods appear to be normal methods in existing tools, and in a non-trait-aware version, it may be difficult to avoid inadvertent and difficult-to-detect changes.

Pharo-style traits in GemStone

GemStone's traits are similar to Pharo traits. GemStone does not support trait composition; only one instance-side trait and one class-side trait can be added to a class. GemStone traits are stateful, and support class and class instance variables access as well as instance variable access.

Terminology

In this discussion, "trait" is used to refer to the logical trait, the set of instance and class methods that are added to a class (although the instance and class sides are added independently). "Trait" is the class that provides the main entry point for a trait, and which implements the instance side trait methods; it provides access to the class side trait methods, but these are associated with a separate class, the ClassTrait.

The term "trait method" refers to the method defined on the trait, which is distinct from the compiled method on the class that originated from the trait method's source string.

Creating and Using traits

A trait is created similar to how a class is created, and methods are added to the trait in a way similar to creating a method on a class. The trait is then added to each of the target classes that will use the trait's behavior.

When a trait method is defined, it is compiled for validation, but the trait methods are stored as source strings. When you add the trait to a target class, these source strings are compiled into the method dictionary of the target class.

When you add, modify, or remove a method on the trait, the methods that originated from trait methods are automatically updated on the classes that are using that trait.

The trait's instance, class, and class instance variables must match or be a subset of the instance, class, and class instance variables of each class to which the trait will be added.

Only variables that are accessed from methods on the trait are required to be defined on the trait.

Added Classes

The following classes have been added to support traits:

AbstractTrait

This is an abstract superclass of Trait and ClassTrait

Trait

Trait represents the instance side methods for a trait. An instance of Trait knows its corresponding class side trait (ClassTrait), which is available by sending #classTrait to the instance.

ClassTrait

ClassTrait represents the class side methods for a trait.

GsTraitImpl

The Trait and ClassTrait instances for a given trait reference a shared instance of GsTraitImpl, which contain the source strings, categories, the classes that are using this trait, and other implementation information. This class is not intended to be used directly.

Precedence

If a selector that is part of a trait is already implemented directly by the target class, the trait's implementation is not used. If a method with the given selector is later compiled directly on the class, it will override an existing trait implementation.

Example

This is an example of a minimal set of trait creation and use, assuming an existing class MyClass:

```
Trait
    name: 'TestTrait'
    instVars: #(  )
    classVars: #(  )
    classInstVars: #(  )
    inDictionary: UserGlobals.

TestTrait compile: 'logFileName
 ^ self class logFileName' category: 'TestTrait-Filename'.

TestTrait classTrait compile: 'logFileName
 ^ ''/tmp/traitLogFile.out''' category: 'TestTrait-Filename'.

MyClass addInstanceAndClassTrait: TestTrait.

exec MyClass logFileName %
exec MyClass new logFileName %
```

Image support for traits

The GemStone image transparently supports traits. Note that tools such as GBS may not distinguish methods that originate from trait methods from ordinary methods or support filein or other trait operations.

Fileout support for traits

Image methods in Behavior and ClassOrganizer transparently handle fileout of methods and traits. The method source for methods that originate from trait methods is not filed out as if they were methods on the class; instead, expressions that can recreate the traits are included.

For example:

```
! ------------------- Traits for MyClass
doit
MyClass addTrait: TestTrait.
MyClass addClassTrait: TestTrait classTrait.
true.
%
```

You must separately ensure that the trait (in the example, a trait named TestTrait) is filed out. This can be done using new methods in Class, Behavior, and ClassOrganizer:

`Behavior >> fileOutTraits`
: Returns a String or MultiByteString with all the receiver's trait specifications (traits and classTraits) in Topaz Filein format. Result should be filed in after filing in methods.

`Behavior >> fileOutTraitsOn:` *stream*
Writes code to create trait registrations onto the given stream in filein format.

`ClassOrganizer >> fileOutTraitsClassesAndMethodsInDictionary:` *aSymbolDictionary* `on:` *aStream*
Files out all source code for classes in aSymbolDictionary in Topaz filein format on aStream.

`ClassOrganizer >> determineTraitFileoutOrder:` *traitdict*
Returns an ordered collection of the values that are traits in *traitdict*, specifying the order of fileout. The argument should be a SymbolDictionary.

`ClassOrganizer >> fileOutTraitDefinitions:` *order* `on:` *stream*
Writes out code in topaz filein format on the given stream, that creates the given traits. *order* arg is an array of traits.

`ClassOrganizer >> fileOutTraits:` *classdict* `order:` *order* `on:` *stream*
File out each class's trait registrations.

GsFileIn support for traits

GsFileIn includes support for the following topaz commands that support traits. For more detail on these topaz functions, see "Added functions for trait support" on page 43.

TRCLASSMETHOD

TRCLASSMETHOD:

TRMETHOD

TRMETHOD:

TRREMOVEALLMETHODS

TRREMOVEALLCLASSMETHODS

ClassOrganizer support for traits

In addition to the fileout support methods above, the following methods have also been added to find out information about traits.

`ClassOrganizer >> traitImplementorsOfReport:` *aSelector*
Returns a String describing the methods that are implementors of the specified selector in traits.

`ClassOrganizer >> traits`
Returns the list of Traits found by the receiver during the class scan.

`ClassOrganizer >> traitStringsReport:` *aString* `ignoreCase:` *icBool* `includeTraitComments:` *commentsBool*
Return a report of the trait methods that match the given string.

Information on Compiled Methods from Traits

The following methods are available to get information about a compiled method, allowing you to distinguish a regular methods from a trait or Rowan method:

`GsNMethod >> isFromTrait`
Return true of the method originated from a trait method definition

`GsNMethod >> origin`
If the receiver is a trait method, answer the Trait instance that created the method. If the method is a loaded method, answer the Rowan loaded method instance. Otherwise answer the class of the method.

Distinguishing Traits

The following methods distinguish a trait from another object:

`Object >> isTrait`

`AbstractTrait >> isTrait`

Trait management in Class

The following methods allow you to add, remove, and get information about traits.

`Class >> addClassTrait:` *aClassSideTrait*

`Class >> addInstanceAndClassTrait:` *anInstanceSideTrait*

`Class >> addTrait:` *anInstanceSideTrait*

`Class >> classTraits`

`Class >> classTraits:` *stringExpressionDefiningClassTrait*

`Class >> removeClassTrait:` *aClassSideTrait*

`Class >> removeTrait:` *anInstanceSideTrait*

`Class >> traits`

`Class >> traits:` *stringExpressionDefiningTrait*

Instance and class traits are managed separately; methods without 'class' require or return an instance side trait.

It is not required that the instance and class traits be related to each other. The exception is `addInstanceAndClassTrait:`, which is a convenience method; this adds both the instance and the class trait to the receiver (the class trait is derived from *aTrait* `classTrait`).

Topaz support for traits

Added functions for trait support

SET TRAIT: *aTrait*

Sets the current trait. After you issue SET TRAIT, you can list the trait's categories and methods with the TRLIST CATEGORIES command. You can select a category to work with through either the SET CATEGORY: or the CATEGORY: command.

The current trait may also be redefined by the TRLIST CATEGORIESIN:, TRMETHOD:, TRCLASSMETHOD:, and TRFILEOUT TRAIT: commands. The current trait is cleared by the ABORT, LOGOUT, LOGIN, and SET SESSION commands.

TRSTRINGSIC *anObjSpec*

Reports Trait methods that contain the argument in their source string, and trait comments that include the argument. The comparison is case-insensitive.

Expects one argument, an objectSpecification, the value of which may be a String or Symbol. The search is in the compilation environment 0 (see ENV).

Equivalent to:

```
ClassOrganizer new traitStringsReport: (aString) ignoreCase:
   true includeClassComments: true
```

TRSTRINGS *anObjSpec*

Reports Trait methods that contain the argument in their source string, and trait comments that include the argument. The comparison is case-sensitive.

Expects one argument, an objectSpecification, the value of which may be a String or Symbol. The search is in the compilation environment 0 (see ENV).

Equivalent to:

```
ClassOrganizer new traitStringsReport: (aString) ignoreCase:
   false includeTraitComments: true
```

TRREMOVEALLMETHODS [*aTrait*]

Removes all methods from the trait whose name you give as a parameter. The specified trait will automatically become the "current" trait.

Example: trremoveallmethods TArray

TRREMOVEALLCLASSMETHODS [*aTrait*]

Removes all class methods in from the trait whose name you give as a parameter. The specified trait will automatically become the "current" trait.

Example: trremoveallclassmethods TArray

TRMETHOD[: *aTrait*]

Compiles an instance method for the trait whose name you give as a parameter. The trait of the new method will automatically become the "current" trait. If you don't supply a trait name, the new method is compiled for the current trait (as set with the SET TRAIT:, TRFILEOUT TRAIT:, TRMETHOD:, or TRLIST CATEGORIESIN: command). Compiles in the compilation environment 0.

Text of the method should follow the TRMETHOD: command on subsequent lines. The method text is terminated by the first line that starts with the '%' character. For example:

After the method has been successfully compiled in the trait, the method will be compiled in the dependent classes registered for the instance side trait.

TRLOOKUP

TRLOOKUP includes subcommands that allow you to browse traits and trait methods.

TRLOOKUP method *selectorSpec*

The arguments after 'TRLOOKUP' are case sensitive. Using the current trait, search for specified instance method. Print the trait in which method found, and list the category and source code. See <selectorSpec> in Object_Specification_Formats.

TRLOOKUP cmeth *selectorSpec*
TRLOOKUP classmethod *selectorSpec*

The arguments after 'TRLOOKUP' are case sensitive. Using the current trait, search for specified class method. Print the trait in which method found, and list the category and source code. See <selectorSpec> in Object_Specification_Formats.

TRLOOKUP *aTrait* >> *selectorSpec* [*envIdInteger*]

Make the specified trait the current trait, and do a lookup for the specified instance method A <traitname> may not be one of 'meth', 'method', 'cmeth', 'classmethod'

TRLOOKUP *aTrait* class >> *selectorSpec* [*envIdInteger*]

Make the specified trait the current trait, and do a lookup for the specified class side method. A <traitname> may not be one of 'meth', 'method', 'cmeth', 'classmethod'

TRLIST

TRLIST includes subcommands that allow you to browse traits and trait methods.

TRLIST TRAITS

Lists all of the dictionaries in your symbol list and all of the traits they contain.

Takes an optional argument aString, if given, limit the result to those traits for which (aTrait name includesString: aString)==true.

TRLIST TRAITSIN: *aSymbolDictionary*

Lists the traits in a specific dictionary. For example: `trlist traitsin: UserGlobals`

TRLIST CATEGORIESIN[: *aTrait*]

Lists all of the instance and class method selectors for the named trait, by category, in the compilation environment 0 (see ENV). If you do not specify a trait name, the categories of the current trait will be listed. If you do specify a trait name, that trait becomes the current trait for subsequent Topaz commands. (The current trait is also set with the SET TRAIT:, FILEOUT TRAIT:, or TRMETHOD: command.)

TRLIST CCATEGORIES[: *aTrait*]

Lists all of the class method selectors for the named trait, by category, in the compilation environment 0 (see ENV). If you do not specify a trait name, the class categories of the current trait will be listed. If you do specify a class name, that class becomes the current class for subsequent Topaz commands.

TRLIST CLASSMETHOD[: *anObjectSpec*]
TRLIST CMETHOD[: *anObjectSpec*]

With an argument that is a <selectorSpec>, lists the category and source of the given classmethod selector for the current trait in the compilation environment 0 (see ENV). (The current trait is set with the SET TRAIT:, TRFILEOUT TRAIT:, TRMETHOD:, or TRLIST CATEGORIESIN: command). See <selectorSpec> in Object_Specification_Formats

TRLIST CSELECTORS [*aString*]

List selectors of all class methods for the current trait in compilation environment 0 (see ENV). Accepts an optional string token, for example: `LIST CSEL policy` lists only selectors which contain 'policy' using case-insensitive match.

TRLIST CPRIMITIVES [*aString*]

List selectors of all class methods for the current trait in compilation environment 0 which are primitives. Accepts an optional string token, for example: `TRLIST CPRIM at` lists only selectors which contain 'at' using case-insensitive match.

TRLIST ICATEGORIES[: *aTrait*]

Lists all of the instance method selectors for the named trait, by category, in the compilation environment 0. If you do not specify a trait name, the categories of the current trait will be listed. If you do specify a trait name, that trait becomes the current trait for subsequent Topaz commands.

TRLIST LINENUMBERS

If a current trait method is set, lists this method with line numbers prefixed to the source lines. The current trait method can be set using TRLOOKUP.

TRLIST METHOD: *aSelector*

Lists the category and source of the given instance method selector in the current trait and compilation environment 0.

TRLIST PRIMITIVES [*aString*]

List selectors of all instance methods that are primitives, in the current trait and compilation environment 0. Accepts an optional string token; this lists only selectors for primitive methods that contain <aString>, using case-insensitive match.

TRLIST SELECTORS [*aString*]

List selectors of all instance methods in the current trait and compilation environment 0. Accepts an optional string token; this lists only selectors that contain <aString>, using case-insensitive match.

TRIMPLEMENTORS *aSelectorSpec*

Expects one argument, an selectorSpec, the value of which is either a String or Symbol.

This command is equivalent to: `(ClassOrganizer newForEnvironment: <current env>) traitImplementorsOfReport: aString`

TRFILEOUT

Writes trait definitions and/or methods to a named file in a format that can be fed back into Topaz with the INPUT command. The current compilation environment(see ENV) is ignored by traits and environment 0 is exclusively used. If you supply a parameter that does not match one of the FILEOUT sub-commands, Topaz will assume it is a method selector for the selected trait and will attempt to file it out. The first line of the output file is a fileformat command reflecting the way output was generated.

Examples:

trfileout trait: TReadStreamLegacy **tofile:** tlegacy.gs **format:** 8bit

set trait TReadStreamLegacy
trfileout category: 'Instance Creation' **tofile:** tlegacy.gs **format:** utf8

Keyword arguments must be provided in this order:

exactly one of TRAIT: CATEGORY: CLASSCATEGORY: METHOD: CLASSMETHOD:
optional TOFILE:

If TOFILE: is given, then it may be followed by an optional FORMAT:

If FORMAT: is omitted, output is is controlled by the last FILEFORMAT.

TRFILEOUT TOFILE: *filename*

The output of the FILEOUT command is sent to the file that you name following the TOFILE: keyword. For example:

```
trfileout trait: TReadStreamLegacy tofile: tlegacy.gs
```

If you specify a host environment name such as $HOME/foo.bar in UNIX as the output file, Topaz will expand that name to the full filename.

If the output file does not contain an explicit path specification, Topaz writes to the named file in the directory in which you started Topaz.

If the filename begins with '&', the '&' is discarded and output is appended to the specified file, otherwise the specified file is overwritten.

TRFILEOUT FORMAT: *format*

Overrides the current FILEFORMAT setting. Argument must be a string, either UTF8 or 8BIT. The FORMAT: keyword must be after a TOFILE: keyword.

TRFILEOUT TRAIT: *aTrait*

Writes out the trait definition and all the method categories and their methods. The trait that you file out will be selected as the current trait for use with other Topaz commands.

TRFILEOUT CATEGORY: *aCategoryName*

Writes out all the methods contained in the named category for the current trait. For example

```
set trait TReadStreamLegacy
trfileout category: 'Accessing' tofile: tlegacy.gs format: utf8
```

TRFILEOUT CLASSCATEGORY: *aCategoryName*

Writes out all the class methods contained in the named category for the current trait. For example:

```
set trait TReadStreamLegacy
trfileout classcategory: 'Instance Creation' tofile: tlegacy.gs
   format: utf8
```

TRFILEOUT CLASSMETHOD: *selectorSpec*

Writes out the specified class method (as defined for the current class). The category of that method is automatically selected as the current category for use with other Topaz commands. Method lookup uses current compilation environment (see ENV). For example:

```
set trait TReadStreamLegacy
trfileout classmethod: new
```

TRFILEOUT METHOD: *selectorSpec*

Writes out the specified method (as defined for the current class). The category of that method is automatically selected as the current category for use with other Topaz commands. The method lookup uses current compilation environment (see ENV). For example:

```
set trait TReadStreamLegacy
trfileout method: next
```

You may omit the "METHOD:" keyword if the method selector does not conflict with any of FILEOUT's subcommands. For example, to file out a method named "category:", you'd need to explicitly include the METHOD: keyword as shown here:

```
trfileout method: category:
```

TRCLASSMETHOD[: *aTrait*]

Compiles a class method for the trait whose name you give as a parameter. The trait of the new method will automatically become the current trait. If you don't supply a trait name, the new method is compiled for the current trait (as set with the SET TRAIT:, TRFILEOUT TRAIT:, TRMETHOD:, or TRLIST CATEGORIESIN: command).

Text of the method should follow the TRCLASSMETHOD: command on subsequent lines. The method text is terminated by the first line that starts with the '%' character.

After the method has been successfully compiled in the trait, the method will be compiled in the dependent classes registered for the class side trait.

Chapter

3 GCI and FFI Changes

Added GCI functions

The following functions have been added:

GciCheckNetldiName

```
(BoolType) GciCheckNetldiName(
    const char* aName
    );
```

Return FALSE if GEMSTONE_NRS_ALL environment variable is defined and contents of GEMSTONE_NRS_ALL disagrees with specified netldi name.

GciExecuteStr__

```
(OopType) GciExecuteStr__(
      const char source[],
      OopType symbolList);
```

This is a variant of GciExecuteStr/GciExecuteStr_, which does not take the enviromentId argument and returns returns OOP_ILLEGAL in case of error.

GciFetchDateTime_

```
(BoolType) GciFetchDateTime_(
      OopType datetimeObj,
      GciDateTimeSType *result);
```

this is a variant of GciFetchDateTime, but returns a BoolType instead of (void).

GciFetchGbjInfo

```
(BoolType) GciFetchGbjInfo(
    OopType theObject,
    int64 startIndex,
    int64 bufSize,
    GciGbjInfoSType *info,
    ByteType *buffer
    );
```

Similar to GciFetchObjInfo, but does not add *theObject* to export set. The value of **info* is a GciGbjInfoSType

GciNewStringFromUtf16

```
(OopType) GciNewStringFromUtf16(
    ushort* words,
    int64 numWords,
    int unicodeKind
    );
```

The value of **words* must be UTF16 encoded data. This method returns OOP_ILLEGAL if an error occurred. *unicodeKind* may be:

- 0 create a String, DoubleByteString or QuadByteString
- 1 create a Unicode7, Unicode16 or Unicode32
- -1 create a string or unicode string per (Globals at:#StringConfiguration)

GCI_OOP_TO_CHAR_

```
(uint) GCI_OOP_TO_CHAR_(
      OopType anOop);
```

This is the same as **GciOopToChar**, but does not check that *anOop*'s class is Character. This is an optimization for when *anOop*'s class in known.

GciOopToFlt_

```
(BoolType) GciOopToFlt_(
      OopType theObject,
      double* result);
```

This is a variant of the existing GciOopToFlt function, which returns a boolean and puts the result into the result argument *double**.

GciPerform__

```
(OopType) GciPerform__(
      OopType receiver,
      const char selector[],
      const OopType args[],
      int numArgs);
```

This is a variant of GciPerform/GciPerform_, which does not require the environmentId argument, and returns OOP_ILLEGAL in case of error.

GciPerformFetchOops

```
(int) GciPerformFetchOops(
    OopType receiver,
    const char *selector,
    const OopType *args,
    int numArgs,
    OopType *result,
    int maxResultSize
    );
```

Do a perform of *selector* on *receiver*, with *args* and *numArgs*. The result of the perform is expected to be an oop format object. Return up to *maxResultSize* of the instVars of the result of the perform. The result of the perform is not added to the export set nor is it returned.

GciSetSessionId_

```
(BoolType) GciSetSessionId_(
   GciSessionIdType sessionId);
```

This is a variant of GciSetSessionId, which returns false if the argument is not valid.

Changed Functions

The following function has changes:

GciHostInstallFaultHandler

GciHostInstallFaultHandler no longer expects arguments.

Added Thread-safe GCI functions

GciTsDirtyExportedObjs

```
(BoolType) GciTsDirtyExportedObjs(
    GciSession sess,
    OopType theOops[],
    int *numOops,
    GciErrSType *err
    );
```

This function returns a list of objects which are in the ExportedDirtyObjs, i.e. objects in the PureExportSet and whose state has been changed by one of the following:

1. Smalltalk execution
2. Calls to GciStorePaths, GciSymDictAtObjPut, GciSymDictAtPut,GciStrKeyValueDictAtObjPut, GciStrKeyValueDictAtPut
3. Any GCI call from within a user action.
4. Committed by another transaction if a commit or abort was executed.
5. Aborted the modified state of a committed object.

GciTsDirtyObjsInit() must be called once after login before GciTsDirtyExportedObjs() can be used. Calls to Ghostwrites functions do NOT put the modified object into the

set of dirty objects. The assumption is that the client does not want the dirty set to include modifications that the client has explicitly made. EXCEPTION: GciStore*, etc, calls from within a user action WILL put the modified object into the set of dirty objects. The *numOops* argument is used as both an input and an output parameter.

On input the value of *numOops* should be set to the maximum number of oops that can be returned in this call, i.e., the size (in oops) of the buffer specified by the first argument. On output the *numOops* argument is set to the number of oops returned in the buffer.

The function result indicates whether the operation of returning the dirty objects is done. If not done, i.e. FALSE, it is expected that the user will make repeated calls to this function until it returns a TRUE to indicate that all of the dirty objects have been returned. If repeated calls are not made, then the unreturned objects will persist in the list until the function is next called.

GciTsFetchGbjInfo

```
(int64) GciTsFetchGbjInfo(
    GciSession sess,
    OopType objId,
    BoolType addToExportSet,
    GciTsGbjInfo *result,
    ByteType *buffer,
    size_t bufSize,
    GciErrSType *err
    );
```

The function result is >= 0 for success, -2 if the object does not exist, and -1 if an error other than read authorization failure was returned in **err*.

addToExportSet has an effect only if the function result is 1.

If *buffer* not NULL, then up to *bufSize* bytes of the body of the object are returned in **buffer*, and function result is the number of instVars returned. If *buffer* == NULL then function result is 0 for success or -1 for error.

If read authorization is denied for *objId*, then result->access == 0, the rest of **result* other than *result->objId* is zero, and function result is zero.

GciTsGbjInfo includes additional information about the class of the object, to allow faster handling of varying kinds of results. See extraBits field of GciTsGbjInfo.

GciTsKeepAliveCount

```
(int64) GciTsKeepAliveCount(
    GciSession sess,
    GciErrSType *err
    );
```

For use in testing the libgcits library. This functions returns number of keep alive bytes received by the session, or -1 if an error was returned in **err*

GciTsKeyfilePermissions

```
(int64) GciTsKeyfilePermissions(
    GciSession sess,
    GciErrSType *err
    );
```

Returns -1 for error, or the uint32 value of keyfilePermissions from the stone process

GciTsNewStringFromUtf16

```
(OopType) GciTsNewStringFromUtf16(
    GciSession sess,
    ushort *words,
    int64 nWords,
    int unicodeKind,
    GciErrSType *err
    );
```

The *words* argument must be UTF16 encoded data. This function returns OOP_ILLEGAL if an error occurred.

unicodeKind may be:

- 0 create a String, DoubleByteString or QuadByteString
- 1 create a Unicode7, Unicode16 or Unicode32
- -1 create a string or unicode string per (Globals at:#StringConfiguration)

GciTsPerformFetchOops

```
(int) GciTsPerformFetchOops(
    GciSession sess,
    OopType receiver,
    const char* selectorStr,
    const OopType *args,
    int numArgs,
    OopType *result,
    int maxResultSize,
    GciErrSType *err
    );
```

Do a perform of *selector* on *receiver*, with *args* and *numArgs*. The result of the perform is expected to be an oop format object. Return up to *maxResultSize* of the instVars of the result of the perform. The result of the perform is not added to the export set nor is it returned.

Other GCI changes

GciRtlLoad, GciRtlLoadA, GciTsLoad ignored path arg

These methods includes a path argument which is intended to allow searching for shared libraries that are not in the standard locations. This argument was not actually used; only the standard locations were being searched. (#50958)

Linked GCI application now also looks for gem.conf

When searching for an executable configuration file, the default name that a linked GCI application is an application name set by **GciInitAppName**. If this has not been set, it will now also look for a file named `gem.conf`, as well as the documented default name `gci.conf`. The search order is *appName*`.conf`, `gci.conf`, `gem.conf`.

Int16Array, Int32Array, Int64Array at:put: did not have correct range checking

The method`at:put:` was previously inherited for the classes Int16Array, Int32Array, Int64Array; the inherited range checking was incorrectly for unsigned rather than the signed range.

FFI Changes

Incorrect error from CByteArray >> encodeUTF8From:into:allowCodePointZero:

`CByteArray>>encodeUTF8From:into:allowCodePointZero:` returned an error indicating an incorrect source class in cases where the destination CByteArray is too small for the converted source string. (#50974)

Chapter

4 Bug Fixes

The following bugs were present in v3.7.1 and are fixed in v3.7.2.

Gem out-of-memory issues

Incorrect handling of code_gen space in Gem memory

The code_gen space, used to keep temporary instances of methods, had deficiencies in handling large amounts of methods, which could result in out of memory errors. Specifically, instances of GsNMethod in the code_gen memory area were not pruned correctly during in-memory garbage collection, and in-memory GC of code_gen is not sufficiently aggressive enough under heavy load conditions (#51019, #51066)

Computed size of code_gen too small

The calculation of the code_gen size based on GEM_TEMPOBJ_CACHE_SIZE in 3.7.x created a smaller size area than in earlier versions.

A configuration parameter has been added to control the code_gen size; see "GEM_TEMPOBJ_CODE_SIZE" on page 35

Abort failed to clear references from modified committed objects to in-memory objects

No-longer valid references from committed objects to in-memory objects were not cleared on abort, which resulted in the subsequent in-memory garbage collection not freeing as much memory as was possible. (#51073)

Memory issues from fillFrom:resizeTo:with:

The primitive invoked by `Array >> fillFrom:resizeTo:with:` and `KeyValueDictionary >> fillFrom:resizeTo:with:` was subject to out of memory errors. The primitive has been removed; for details on how this has been handled, see "fillFrom:resizeTo:with: no longer usable" on page 18. (#50990).

AlmostOutOfMemory handlers not effective if needed while in primitive

Defining exception handlers for AlmostOutOfMemory or AlmostOutOfMemoryError allows you to catch conditions in which your Gem would otherwise run out of temporary object memory and exit, and perform some actions to make more memory available. However, if the operation that required more memory than is currently available occurred when the code is executing in a primitive call, neither AlmostOutOfMemory nor AlmostOutOfMemoryError handlers are able to intercept the exception and take action to avoid an exit with out of memory.

Now, some primitives that are more likely to see this condition now perform checking before attempting to allocate memory, including the primitives invoked by methods such as `String >> copy`, `String >> addAll:`, and `String >> ,`. If the result would consume more than 16K bytes of temp object memory, it checks for available space in temporary object memory. If space is insufficient, a VM memory mark/sweep is executed, and if space is still insufficient, it signals a not-resumable AlmostOutOfMemoryError; regardless of the state of either of `AlmostOutOfMemoryError class >> enabled` or `AlmostOutOfMemoryError class >> threshold`. An AlmostOutOfMemoryError handler will need to determine how to handle the situation; it is recommended to use AlmostOutOfMemoryError rather than AlmostOutOfMemory (which is a kind of Notification). (#51076)

Issues in Native code generation

Results of ifTrue/ifFalse blocks

Issues have been found in native code that caused code errors or SEGV. These issues are exposed by expressions that assign the result of ifTrue/ifFalse expressions that return Array constructors. (#51093, #51094)

Incorrect handling of AlmostOutOfStack after process switch

When a GsProcess switch occurred, the stack limit is set too low for a GsProcess that is executing within the yellowZone handling an AlmostOutOfStack or AlmostOutOfStackError. (#51168)

Conflict handling in Reduced-conflict collections

A number of improvements have been made in RC collections to avoid commit conflicts. In particular, RcIdentityBag has internal changes. See “Improvements to reduced-conflict collections” on page 19 for more information.

Failure to detect write conflict after reduced-conflict replay

When two sessions make changes to the same object in an reduced-conflict collection, the second session to commit sees a write-write conflict during commit, selectively aborts, and replays the change. If there was another non-RC commit meanwhile to this same object, such that it should result in a second write-write conflict, it was not detected by the replay. This caused a later Stone crash with errors such as "page NNN already shadowed". (#50235)

RcIdentityBag conflicts with unexpected larger sessionIds

When a new session, that has a sessionId greater than any sessionId that has previously made changes to an RcIdentityBag, adds or removes an element, an internal structure may need to be enlarged. If another session was also making potentially conflicting changes to the RcIdentityBag, the replay did not handle this correctly and could trigger a concurrency conflict. (#51033)

RcKeyValueDictionary rebuildTable likely to cause commit conflicts

When multiple sessions are modifying an RcKeyValueDictionary and adding keys that were not previously present, the internal structure supporting the dictionary may require rebuilding to a larger size, to accommodate the additional keys. If a commit conflict occurs, the rebuild table is replayed, but it was likely to encounter further commit conflicts. (#51032)

Issues related to reclaim

ReclaimGem parameter #reclaimMinFreeSpaceMb not respected

This GcGem parameter is designed to suspend reclaim activity when the amount of free space in the repository is low, to avoid using up all free space. The check for this condition was being not done at the point where it would have been effective, and did not avoid reclaim bringing down free space. This parameter has been removed; see "Improved reclaim behavior with low free space" on page 17. (#51260, #51258)

After commitRestore, ReclaimGem may not be restarted

Under rare conditions, after a commitRestore, the automatic restart of the ReclaimGem may not be performed. (#51001).

Signal handler chaining may cause SEGV in linked session

If the java library is loaded into a process before the GemStone linked library, then handling of a SEGV from Java may cause a SEGV in the GemStone code. Loading libjvm into a linked or RPC gem it is not an issue, since the libjvm signal handler will run first and the SEGV is not likely to occur; this bug primarily affects GemBuilder for Java. The GemStone VM does not produce SEGVs normally, unless native code is disabled (in which case SEGV is used to implement AlmostOutOfStack). (#51106).

Improved performance of restoreFromTranlogs

During restoreFromTranlogs, the restoreFromTranlogs did not track the volume of changes, and thus did not commit often enough. When the number of modified data pages and/or write set became very large, commit is inefficient. restoreFromTranlogs now tracks the commit size, and commits more frequently. (#50960)

Insufficient information on extent write errors that succeed on retry

When a write to the extent encounters a write error, it retries the write, which may succeed. These errors are noted in the stone log in case they indicate I/O issues. Now, more detailed information is provided about the write error. (#51105)

Risk of stuck spin lock on LostOt

If a LostOT or SIGTERM occurs when a Gem is performing a multithreaded operation, and a slave thread is in the middle of a cache read, this thread may have left a stuck spin lock when it terminated. (#50984)

Issues related to repository scan operations

Possible missing results from findReferencesToInstancesOfClasses, etc.

The C code that supports the multithreaded repository scan methods `findReferencesToInstancesOfClasses:`, `allReferencesToInstancesOfClasses:`, and related methods may theoretically miss some results due to a missing mutex. (#50929)

GsObjectInventory overstated String bytes used, especially for large strings

GsObjectInventory reports the number of bytes used by instances of particular classes in the repository. For Strings, when the repository contains many large Strings, the number of bytes used was considerably overstated; overages by as much as 8x have been observed. (#51053)

Listing instances in memory did not observe limit argument

The methods `Repository >> _listInstancesInMemory:limit:option:`, and methods that invoke them such as `Repository >> listInstances:limit:toDirectory:withMaxThreads:maxCpuUsage:memory:`, did not observe the limit: argument, and returned all instances. (#51080)

Repository >> objectsInMemoryLargerThan: not working

The method `Repository >> objectsInMemoryLargerThan:` incorrectly returned empty results. (#51125)

Class >> instancesInMemory may incorrectly return an empty Array

Due to an C heap memory initialization issue, `Class >> instancesInMemory` could return an empty Array when there are instances in memory to return. This has been observed only on macOS on Apple silicon (ARM) in internal testing. (#50680)

Numerics issues

Values of different classes of numeric values that are close but not equal may compare as equal

When numbers of different classes, such as 0.3s1 (ScaledDecimal) and 0.3 (SmallDouble) are compared, one value must be coerced for the comparison. This did not always preserve the difference when the values are close. Now, precise comparisons are done. Some values that were close in value and previously compared as equal may now not be equal. This affects comparisons done using =, ~=, >, >=, <, and <=. (#50599)

Non-numeric arguments to comparison operators

Using a non-numeric argument to <, <=, >=, > now signals an ArgumentTypeError, rather than a MessageNotUnderstood or other error. (#50599)

Incorrect results from -1 bitShift: with arguments greater than 60

When the argument to `-1 bitShift:` is greater than 60, the result was incorrect, due to a failure to overflow to a large integer. Ths only affected a receiver of -1. (#51097)

Some kinds of Number did not understand #isZero

Kinds of Integer, BinaryFloat and DecimalFloat did not understand #isZero. (#51193)

OffsetError from ScaledDecimal >> kind

The `ScaledDecimal >> kind` method yielded an OffsetError, rather than an expected kind (#normal, #zero, etc.). (#51133)

Issues with GsUuidV4 comparison operators with unexpected argument types

The method `GsUuidV4 >> <` SEGVed when the argument is certain specials (such as SmallDoubles). (#51121)

The methods `GsUuidV4 >> =` and ~= returned ArgumentErrors when the argument is a special (such as SmallIntegers). (#51120)

DateTime newWithDate:time: truncated Time to seconds

Instances of Time support microsecond resolution. When creating a DateTime using `newWithDate:time:` or `newWithDate:time:timeZone:`, the Time argument was truncated to seconds, although DateTime supports down to millisecond resolution. Note that a sub-millisecond resolution of the Time instance is still lost. (##50971)

In solo mode, transactionMode: could cause abort

In solo mode, sending transactionMode: should have no effect, since commits are disallowed. This operation incorrectly aborted the session (performed a soloAbort). (#50986)

Stone continued running if ShrPcMon SEGV or other death

If the shared page cache monitor exits normally, such as by a kill -TERM, the Stone is also shut down. However, with a SEGV or kill -9, the Stone continued running and processing session commands, although logins would fail. (#51104)

NetLDI in root mode reported owner as regular user

When the NetLDI is running in root mode (with the executable owned by root with the s bit set), the owner was reported as the user that started the NetLDI, not as root. Now, gslist -x will report the owner as root. (#50995)

File descriptor leak in GsHostProcess

There was a file descriptor leak in GsHostProcess that was fixed in earlier versions; this fix was not complete, and additional leaks have been found and closed. (#47630)

This results in behavior changes in `read:*` and `readWillNotBlock` messages to the GsHostProcess's child's sockets. For more details, see "Change in behavior in 3.7.2 that may affect existing application code" on page 15.

Cache Statistics Issues

SEGV on programmatic cache statistics access by name for ProcessName

Executing `System stoneCacheStatisticWithName: 'ProcessName'`, or other by-name programmatic cache statistics named 'ProcessName', resulted in a SEGV. (#51122)

Memory corruption from hostEasyStatistics*

The System class methods `hostEasyStatisticsForProcess:` and `hostEasyStatisticsForMyProcess` were added in 3.7.1, for a more performant partial set of statistics. The C code supporting this function had a risk of corrupting static C memory. (#50998)

System cacheStatsForGemWithName: some stats incorrectly 0

When using `System cacheStatsForGemWithName:` to access statistics values, the values for a number of statistics were incorrectly reported as 0, vs. the correct values reported by `System myCacheStatistics`. The specific incorrect stats that were incorrect varied over GS64 versions. (#51130)

Cache stats of -1 may be returned as 4294967295 by stoneCacheStatistics

Some statistics, such as CommitQueueThreshold that are intended to be reported as -1 were reported as 4294967295 by methods such as `System class >> stoneCacheStatistics`.

In hot standby, currentSessionNames may intermittently show an internal session

In a hot standby setup, the Stone uses an internal session to manage the restore into the slave. This session was intermittently visible from `System class >> currentSessionNames`, but was not identified. In particular, after a failover this session may not have been completely logged out. (#51226)

Repository >> addTransactionLog:size: did not correctly handle size argument

When this methods is executed to add a tranlog directory programmatically, updated configuration lines for STN_TRAN_LOG_DIRECTORIES and STN_TRAN_LOG_SIZES are added to the configuration file used by the stone (normally system.conf).

STN_TRAN_LOG_SIZES can be specified in units of KB, MB, or GB. The bug is that the line written to system.conf uses the units in the existing STN_TRAN_LOG_SIZES setting, which may not be in MB. While the internal value in the Stone is correct, on Stone restart, the incorrect unit is read and applied, and may result in unexpected tranlog sizes. (#51005)

LostOT may result in errors described as PageLocate error

During LostOT handling, if a remote session does not respond and is marked as invalid, the specific error reporting depends on the process that first encounters the invalid session. This could be reported as a PageLocate error, which was unnecessarily alarming. This is now reported more clearly as the result of a LostOT (#51084, #50722)

Issues affected X509-Secure GemStone

Working-set cache warming was not supported for X509-Secured caches

When using the X509-Secure GemStone feature, warming caches that are started by the NetLDI are supported via the configuration parameter NETLDI_WARMER_ARGS. This did not support working-set cache warming, in which active pages are tracked and preferentially warmed on restart. Now, you may include **-w** *interval* argument in this configuration parameter.

Note that, in order to warm a mid-level cache, NETLDI_PORT_RANGE and STN_PGSVR_PORT_RANGE must match.

GemStoneX509Parameters >> extraGemArgs: were ignored

When using `GemStoneX509Parameters >> extraGemArgs:` to apply gem options such as a larger temp obj cache, the composition of the Gem command was not correct, and these arguments were not used. (#51221)

Multithreaded scan operations could hang in a remote session

Operations such as listInstances that perform a multithreaded scan, hung in an X509 gem that is running on host remote from the Stone. (#51220)

System currentUserSessionCount included hostagent

`System class >> currentUserSessionCount` incorrectly counted the hostagent as a user session. (#51219)

SEGV handler now includes raw Smalltalk stack details

Previously, if the Gem signalled a SEGV, even if it was executing Smalltalk code, the Smalltalk stack was not printed. Now, this is automatically included in the log. (#51055)

Darwin error "attempt to create a CByteArray or CPointer that would reference VM memory"

In some cases with very large TOC on Darwin, this error was seen. This issue is related to incorrect handling of the result of a dlopen() call. (#51060)

Object >> _primitiveAt:put: could incorrectly grow non-indexable objects

Objects that are instances of classes that are not indexable can still be grown using `Object >> _primitiveAt:put:`. This should not happen in normal use, and this method is private and should not be invoked directly. (#50970)