GemStone/S 64 Bit™
Release Notes

Version 3.6.2

November 2021

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 is or has been covered by U.S. Patent Number 6,256,637 "Transactional virtual machine architecture" (1998-2018), Patent Number 6,360,219 "Object queues with concurrent updating" (1998-2018), Patent Number 6,567,905 "Generational garbage collector with persistent object cache" (2001-2021), and Patent Number 6,681,226 "Selective pessimistic locking for a concurrently updateable database" (2001-2021).

TRADEMARKS

GemTalk, **GemStone**, **GemBuilder**, **GemConnect**, and the GemTalk logo are trademarks of GemTalk Systems LLC, or of VMware, Inc., previously of GemStone Systems, Inc., in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Solaris, **Java**, and **Oracle** are trademarks or registered trademarks of Oracle and/or its affiliates. **SPARC** is a registered trademark of SPARC International, Inc.

Intel and **Pentium** are registered trademarks of Intel Corporation in the United States and other countries.

Microsoft, **Windows**, and **Windows Server** are registered trademarks of Microsoft Corporation in the United States and other countries.

Linux is a registered trademark of Linus Torvalds and others.

Red Hat and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.

Ubuntu is a registered trademark of Canonical Ltd., Inc., in the U.S. and other countries.

SUSE is a registered trademark of Novell, Inc. in the United States and other countries.

AIX, **POWER7**, **POWER8**, **POWER9** and **VisualAge** are trademarks or registered trademarks of International Business Machines Corporation.

Apple, **Mac**, **MacOS**, and **Macintosh** are trademarks of Apple Inc., in the United States and other countries.

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

Raspberry Pi is a trademark of the Raspberry Pi Foundation

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 LLC
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 3.6.2 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 3.6.2.

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 3.6.2

Chapter

1 Release Notes for 3.6.2

Overview

GemStone/S 64 Bit™ 3.6.2 is a new version of the GemStone/S 64 Bit object server. Version 3.6.2 provides a number of new and enhanced features and fixes a number of bugs. We recommend everyone using or planning to use GemStone/S 64 Bit upgrade to this new version.

These Release Notes include changes between the previous version of GemStone/S 64 Bit, v3.6.1, and v3.6.2. If you are upgrading from a version prior to 3.6.1, review the release notes for each intermediate release to see the full set of changes.

For details about installing GemStone/S 64 Bit 3.6.2 or upgrading from earlier versions of GemStone/S 64 Bit, see the *GemStone/S 64 Bit Installation Guide* for v3.6.2 for your platform.

Supported Platforms

Platforms for Version 3.6.2

GemStone/S 64 Bit version 3.6.2 is supported on the following platforms:

- Red Hat Enterprise Linux Server and CentOS Linux 7.9 and 8.4; and
  Ubuntu 18.04 and 20.04
  GemStone performs testing on a mixture of Red Hat and CentOS servers; both are fully certified platforms. Any reference to Red Hat applies to both distributions.
- Solaris 10 on x86
- AIX 7.1 and 7.2
- OSX 11.1 (Big Sur) with Darwin 20.2.0 kernel on x86, and OSX 10.15.6 (Catalina) with Darwin 19.6.0 kernel; and
  OSX 11.6 (Big Sur) with Darwin 20.2.0 kernel on Apple M1
  (Mac is supported for development only)

For more information and detailed requirements for each supported platforms, please refer to the *GemStone/S 64 Bit Installation Guide* for v3.6.2 for that platform.

GemBuilder for Smalltalk (GBS) Versions

GemStone/S 64 Bit version 3.6.2 requires GBS version 8.5 or later for VisualWorks Smalltalk, or version 5.4.6 or later for VA Smalltalk.

The following versions of GBS are supported with GemStone/S 64 Bit version 3.6.2:

GBS/VW version 8.5

VisualWorks 9.0 32-bit and 64-bit	VisualWorks 8.3.2 32-bit and 64-bit	VisualWorks 8.2.1 32-bit and 64-bit
▸ Windows 10 ▸ RedHat ES 7.9 and 8.4; Ubuntu 18.04 and 20.04	▸ Windows 10 ▸ RedHat ES 7.9 and 8.4; Ubuntu 18.04 and 20.04	▸ Windows 10

GBS/VA version 5.4.6

VAST Platform 10.0.2	VAST Platform 9.2.2	VAST Platform 8.6.3
▸ Windows Server 2016 and Windows 10	▸ Windows Server 2016 and Windows 10	▸ Windows Server 2016 and Windows 10

For more details on supported 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 v3.6.2 distribution includes VSD version 5.5.2. This previous version of GemStone/S 64 Bit, v3.6.1, included VSD v5.5.1.

VSD version 5.5.2 includes several minor bug fixes. For details on the changes, see the Release Notes for VSD v5.5.2.

Note that in GemStone/S 64 Bit v3.6 and later, **statmonitor** writes additional information to the statmonitor file. As a result, statmonitor files from v3.6 and later cannot be read by versions of VSD earlier than v5.5. VSD 5.5.2 can read statmonitor files generated in older versions of GemStone/S 64, 32-bit GemStone, and GBS, as well as those generated by GemStone/S 64 Bit v3.6.

VSD 5.5.2 is included with the GemStone distribution, and can also be downloaded as a separate product from https://gemtalksystems.com/vsd/

Changes in this version

Updated library versions

The version of OpenSSL has been updated to 1.1.1l

The version of Kerberos has been updated to 1.19.2

The version of OpenLDAP has been updated to 2.4.59

Number rounding/truncation

When a number is rounded using `roundTo:` or `truncateTo:`, it returns an object of the same class as the receiver, per the ANSI standard. However when rounding, for example, a float to an integer, it may be more useful to return an objects of the class of the argument.

New methods have been added to round/truncate and return an object of the class of the argument. These have been added to AbstractFraction, Float, Integer, and ScaledDecimal.

`roundAndCoerceTo:` *aNumber*
Returns the multiple of *aNumber* that is nearest in value to the receiver.

`truncateAndCoerceTo:` *aNumber*
Returns the multiple of *aNumber* that is closest to the receiver, on the same side of the receiver as zero is located. Returns the receiver if the receiver is a multiple of *aNumber*.

Added ability to customize passivation of dynamic instance variables

The following method has been added:

`Object >> shouldWriteDynamicInstVar:` *instVarName*
Returns whether the dynamic instance variable with the name *instVarName* should be written out. during passivation. The default is to write out all dynamic instance variables.

This method is invoked during passivation. Classes may implement an override to this method to customize the passivated form of an object that has dynamic instance variables.

GemStone-specific SUnit classes now renamed and in image

The classes GSTestCase, GSTestResult, and GSTestSuite, which have been distributed in the `$GEMSTONE/examples/testing` directory, have been renamed to GsTestCase, GsTestResult, and GsTestSuite, and are now in Globals in the image.

These classes provide additional protocol and GemStone-specific behavior to the existing SUnit framework. These classes are also used internally at GemTalk for product testing.

The distribution no longer includes the directory `$GEMSTONE/examples/testing`.

NonPersistentArray

The class NonPersistentArray has also been added to the image; this is used by the GsTest framework, but is available for general use. It is a kind of Array with the class option #instancesNonPersistent.

Performance improvements in WriteStreamPortable

Some work has been done to improve `nextPut*` performance for WriteSteamPortable.

Converting ByteArrays containing UTF-16 into UTF-8

ByteArrays can contain UTF-16 encoded data; however, each word of UTF-16 character data requires two bytes in the ByteArray, and may be arranged as little or big endian.

To convert such ByteArrays to UTF-8 (instances of the class Utf8), the following methods have been added:

`ByteArray >> decodeFromUTF16BeToUTF8`
Decode UTF16-BE contents of the receiver, and return an instance of Utf8.

`ByteArray >> decodeFromUTF16LeToUTF8`
Decode UTF16-LE contents of the receiver, and return an instance of Utf8.

Utf8 >> readStream now permitted

Previously, the method `Utf8 >> readStream` was implemented to disallow this operation. The disallowed implementation in `Utf8` has been removed, so the version inherited from SequencableCollection can be used.

Added ability to check if private key matches an encrypted extent

To verify that a private key matches the key used to encrypt an extent, without needing to decrypt the extent, a new option has been added to copydbf, the -c option. This takes similar arguments to the copydbf command to decrypt an extent, but does not take a destination filename and does not decrypt, and prints a string with the result. The return code is 0 for a match, 2 for a mismatch, and 3 for error.

Usage for this options is:

`copydbf` *srcFile* `-c -D` *privKey* `-K` *dir* `[-K` *dir*`+] [-j` *passphrase* `| -J` *pfFile*`]`

Argument details are in the help text (copydbf -h output).

For example:

```
unix> copydbf extent0.sdbf -c -D server_1_serverkey.pem
-K $GEMSTONE/examples/openssl/private/allcerts
-J $GEMSTONE/examples/openssl/private/server_1_server_passwd.txt
[Info]: Private key server_1_serverkey.pem matches secure extent
extent0.sdbf
```

gslist -m contacting remote NetLDI running with authentication

gslist -m *hostNameOrIp* contacts the NetLDI running on the remote host, to query for information on running processes on *hostNameOrIp*. If the NetLDI on the remote host is running with authentication (**startnetldi -a** argument), this could cause issues with Unix/LDAP account disable. If the NetLDI additionally was running in secure mode (**startnetldi -s** argument), **gslist -m** could not be used.

NetLDI -a attempted authentication with empty password

When a **gslist -m** request was made, it synthesized an empty password for the request to the remote NetLDI. If that remote NetLDI was running with authentication for new

processes only (using **startnetldi -a** but not **-s**), the NetLDI attempted to authenticate using PAM, which would fail. This did not cause the **gslist** query to fail, since authentication for **gslist** is not required in this mode, and results were returned.

However, if this call was made often enough, these failures could result in the underlying account being locked as a result of too many failed authentication requests. (#49591)

The NetLDI no longer authenticates an empty password.

gslist -m not usable with remote NetLDI in secure mode

When **gslist -m** contacts the NetLDI on a remote host, it uses the userID of the user executing the **gslist**, and an empty host password. If the remote NetLDI is running in secure mode (**startnetldi -s** argument), then **gslist -m** would fail. (#49594)

The following option has been added to **gslist**:

```
-a userid Unix userId used for contacting remote netldi
   with -m. If -m also specified, prompts for a Unix password.
   Has no effect unless -m is specified, required if remote
   netldi was started with -s.
   Do not use if remote netldi started with -k (for Kerberos).
```

On a system where the NetLDI is running with authentication (the **startnetldi -a** but not **-s**), using **-a** with **gslist -m** to a remote host is optional. If the NetLDI is running in secure mode (**startnetldi -a** and **-s** options), the **gslist -a** option is required with **-m**. The **gslist -a** argument has no effect running on the local host.

When **gslist** includes the **-m** and **-a** arguments, **gslist** prompts the user, on the command line, for the password for the remote host.

Note that single sign on using Kerberos legitimately uses an empty password. This mode is not affected by changes in this release; the **-a** option should not be used in a Single sign-on system.

Admin GcGem's number of threads is now dynamically configured

The way the number of threads that the AdminGem uses for epoch garbage collection and write set union sweep is now calculated dynamically, and has a minimum of 2 threads and a maximum of 256. The names of the configuration parameters #epochGcMaxThreads and #sweepWsUnionMaxThreads are no longer descriptive; these are not maximum values, and the values printed in the AdminGem log on startup do not reflect the number of threads actually used.

The number of threads that are used for an AdminGem operation is now a percentage of the size of the union set to be swept, limited by the current setting of the Stone configuration STN_NUM_GC_RECLAIM_SESSIONS and the setting for #epochGcMaxThreads/#sweepWsUnionMaxThreads.

The number of threads used by a sweep is now printed in the AdminGem log; for example:

```
--- 08/12/2021 12:25:02.814 PDT Starting SweepWsUnion, WSUsize=
2000 PDsize=1410703 startCR=6877 numThreads=2
```

or for Epoch:

```
--- 08/12/21 11:16:03.273 PDT
[Info]: Starting Epoch GC numThreads=2
```

Object Read Tracking

Object Read Tracking enables GemStone to record the time and userId when specific objects in the repository are read. Records of read operations for all session are written to comma-delimited (CSV) files by the Stone.

Read tracking is enabled for a repository using the new stone configuration parameter STN_OBJECT_READ_LOG_ENABLED (page 18).

Objects for which reads are tracked are associated with specific GsObjectSecurityPolicies; each object in that GsObjectSecurityPolicy will have reads tracked. Objects can also be added to the tracking log dynamically.

By default, reads by all UserProfiles (other than special system users, listed on page 18) are tracked. Individual UserProfiles can be configured to not have reads tracked, so for example batch jobs can run without tracking; and UserProfiles with appropriate privileges can dynamically disable and re-enabled object read tracking.

Configuring objects to be tracked

A new instance variable, `trackReads` has been added to GsObjectSecurityPolicy; by default, this is false. When this is set to true, all objects associated with that security policy have reads tracked.

The following methods have been added:

`GsObjectSecurityPolicy >> trackReads`
Returns a Boolean, true if object read logging is enabled for objects in this security policy.

`GsObjectSecurityPolicy >> trackReads:` *aBoolean*
If *aBoolean* is true enables object read logging for objects in this security policy, false disables it. Signals an Error if any element of AllUsers has a userId incompatible with writing to a .csv file.

Before any read tracking can be recorded, the application configure the objects to be tracked by doing the following:

- Define a security policy with `trackReads` set to true
- Determine which application objects require read tracking
- Associate these objects with the read tracking security policy.

In addition, specific individual objects can be tracked within a specific session by using the following method:

`Object >> trackRead`
If receiver is not a special object, and (System objectReadLogEnabled == true) and ((System myUserProfile _hasPrivilegeName: #DisableObjectReadLogging) == false), add the receiver to the sessions read tracking buffer. Returns true if the object was added to the read tracking buffer, false otherwise.

This addition to the read tracking buffer is independent of whether (self objectSecurityPolicy trackReads == true) and is independent of whether this method has already been sent to the receiver.

Enable and disable Read Tracking

Read tracking must be enabled by setting the configuration parameters STN_OBJECT_READ_LOG_ENABLED (page 18) and STN_OBJECT_READ_LOG_DIRECTORIES (page 18) before the Stone is started up.

If a session logs in as a user with the DisableObjectReadLogging privilege, then read tracking is not enabled for that particular user.

If a session logs in as a user without the DisableObjectReadLogging, objects that have object read logging configured are written to the object read logs.

If the user additionally has the DynamicDisableObjectReadLogging privilege, the user can disable read tracking temporarily within that session by executing

```
System setObjectReadTracking: false
```

When object read tracking is dynamically disabled, a record is written to the object read tracking log.

To determine if the current session has read logging currently enabled, send:

```
System objectReadLogEnabled
```

The following methods have been added:

`System class >> objectReadLogEnabled`
: Answer a Boolean indicating if the Stone was started with the configuration parameter STN_OBJECT_READ_LOG_ENABLED set to true. Note that the actual state of object read logging will depend on the privileges of the current session's userProfile and if object reads tracking was dynamically disabled; see `System class >> setObjectReadTracking:`.

`System class >> objectReadTrackingEnabled`
: Returns true if object read tracking is currently being performed for this session, for any objects with tracking configured via `GsObjectSecurityPolicy >> trackReads` or `Object >> trackRead`. It returns false if no object read logging will occur. Specifically, this method will:

  - return false if Object read tracking is not enabled for the Stone
  - return false if the current session's UserProfile has the DisableObjectReadLogging privilege
  - return false if the session has dynamically disabled Object Read Tracking; that is, `System setObjectReadTracking:`*false* must not have been executed and be still in effect.

`System class >> setObjectReadTracking:` *aBoolean*
: Dynamically disable or re-enable object read tracking for the current session, and return the previous status of object read tracking. Requires the user executing have DynamicDisableObjectReadLogging privilege. If `setObjectReadTracking:` successfully disables or re-enables read tracking, a record is written to the Object read tracking log.

  When *aBoolean* is false, then if object read logging is not enabled for the Stone, or if the current session's UserProfile has the DisableObjectReadLogging privilege, or if `setObjectReadTracking:` *false* was previously invoked to disable read tracking, then this method has no effect, and returns false. A true value for *aBoolean*

only has effect if `setObjectReadTracking:` *false* previously executed successfully.

`setObjectReadTracking:` is expected to be used in constructions such as:

```
|  prev |
[  prev := System setObjectReadTracking: false.
   query Execute ] ensure:
      [ System setObjectReadTracking: prev ]
```

Read Tracking

Tracking occurs when an object is faulted into the temporary object memory of a session. Only one read is recorded per object per session. Once an object has been tracked as read, if the object is read again, no further tracking events are recorded for that object in that session.

The session sends tracking records to the Stone:

- At each transaction boundary (commit or abort)
- When the gem's tracking buffer is full. The buffer is about 100K bytes.
- When the gem explicitly calls `System class >> flushObjectReadBuffer`
- At session logout.

Each read record in the buffer has the timestamp of the first object recorded in the buffer (this avoids excessive calls to fetch exact timestamps from the OS). There is a final record in the buffer, indicating the end of the buffer, which also has an exact timestamp.

Limitation in Read Tracking

If the session terminates with a fatal error or is killed with SIGTERM, the last tracking buffer may be lost. This creates a way for a user to intentionally (as well as inadvertently) read an object without this read action being recorded in the log.

Object Read Log File

The object read log location is specified using a new stone configuration option, STN_OBJECT_READ_LOG_DIRECTORIES (page 18). This specifies a list of one or more directories in which to write object read log files. The Stone will not start up if STN_OBJECT_READ_LOG_ENABLED is true and STN_OBJECT_READ_LOG_DIRECTORIES does not contain at least one valid, writable directory.

The object read log file name will be composed as follows:

ObjectReadLogDir`/`*stoneName*`-ObjectReadLog_`*timestamp*`[.current].csv`

where:

- *ObjectReadLogDir* is the directory for object read logs, one of the directories specified by STN_OBJECT_READ_LOG_DIRECTORIES in the stone configuration file.
- *stoneName* is the short name of the stone (without the NRS).
- *timestamp* is UTC (GMT) time when the file was created, in the format `YYYY-MM-DD-hh:mm:ss.nnn`, where `nnn` is the 3 digit milliseconds of the time.
- `[.current]` is a suffix that appears only on the file that is currently being written by the Stone. Files that are not actively being written to do not have this suffix.

The maximum size of the object read log file in bytes is specified using the new stone configuration parameter STN_OBJECT_READ_LOG_MAX_FILE_SIZE (page 18). When the maximum file size is reached, the current object read log file will be closed and renamed to remove the `.current` suffix, after which a new log file will be opened.

If a buffer of records from a Gem needs to be written, but will not fit into the current object read log file without exceeding the file maximum size, a new log is started. The contents of a Gem's buffer of read records is not split over two object read log files. Each Gem's buffer of records is written contiguously, before another Gem's buffer.

When stone starts, if a single `.current.csv` file is found within the directories specified by STN_OBJECT_READ_LOG_DIRECTORIES, and if this file is smaller than the maximum file size, it will be reopened for writing. Otherwise a new object read log file will be started.

When the local system time crosses midnight UTC, the current object read log is closed and a new one is started.

The following methods have been added:

`System class >> currentObjectReadLogFile`
: Answer a string containing the current object read log file which the stone is writing to, or nil if the object read logging feature is disabled.

`System class >> startNewObjectReadLog`
: Requests Stone close the current object read log file and start a new one. Blocks until the operation has completed. Returns true on success, false if the object read logging feature is disabled, and raises an exception on error. Requires FileControl privilege.

`System class >> flushObjectReadBuffer`
: Forces the gem to immediately send any buffered object read records to the stone.

File Management

Applications must manage storage and archiving of object read log files, to ensure the file system containing these logs does not become full.

It is recommended that the log files be compressed using gzip or lz4 data compression, and that archiving scripts run using CRON so that object read log files are automatically archived.

If a write to the object read log file fails, the following actions will be taken in sequence:

1. Log a message to the stone log regarding the write failure.
2. Retry the write 3 times.
3. If the write still fails, open a new object read log file in the next directory.
4. Repeat item 2 until all object read log directories have been tried.
5. Log a message to the stone log indicating all object read log space is full.
6. Suspend (block) the session and retry opening a new object read log file periodically.
7. Terminate the session if the writes cannot be completed within 5 minutes or the blocked session causes a commit record backlog, whichever occurs first.

Errors opening, renaming and writing the object read log file are recorded in the Stone log.

Added Configuration File Options

STN_OBJECT_READ_LOG_ENABLED

A Boolean (true or false) indicating if object read logging is enabled.

Runtime equivalent: #StnObjectReadLogEnabled (read-only at runtime).
Default: false.

STN_OBJECT_READ_LOG_DIRECTORIES

A list of directories where object read log files can be written. Must contain at least 1 element if STN_OBJECT_READ_LOG_ENABLED is true. The Stone will not start if STN_OBJECT_READ_LOG_ENABLED is true and this configuration parameter is not set to at least one valid directory.

Runtime equivalent: none
Default: none

STN_OBJECT_READ_LOG_MAX_FILE_SIZE

Maximum size in bytes an object read log file may grow to before the current file is closed and a new object read log file is opened.

Default: 1MB, Min: 512KB, Max: 16GB
Runtime equivalent: #StnObjectReadLogMaxFileSize (only modifiable by SystemUser)

UserProfiles exempt from read tracking

UserProfiles may be given a new privilege, DisableObjectReadLogging (page 18).

UserProfiles with this privilege do not have their object reads tracked. To allow an automated batch job to operated without generating excessive read tracking, execute that job as a UserProfile with the DisableObjectReadLogging privilege.

To ensure the UserProfiles can be clearly written to the .csv file, creation of new UserProfiles requires that the userId contain only compatible characters; see "Admin GcGem's number of threads is now dynamically configured" on page 13.

Added Privileges

DisableObjectReadLogging

UserProfiles with this privilege never have their object reads tracked. SystemUser, DataCurator, GcUser, SymbolUser, HostAgentUser and CodeLibrarianUser always have this privilege.

Existing and new users without this privilege will have reads tracked, if object read tracking is enabled for the repository and has not been dynamically disabled.

DynamicDisableObjectReadLogging

UserProfiles with this privilege may dynamically disable their read tracking using `System setObjectReadTracking: false`. Users without this privilege cannot disable read tracking. Has no effect for users with DisableObjectReadLogging, who never have read tracking enabled.

Added Cache Statistics

The following cache statistics related to object read tracking have been added:

NumInReadTrackingQueue (Stn)
Number of sessions waiting for data to be written to the Read Tracking Log.

ObjectsReadTracked (Gem)
Number of object faults for which a record was added to the Read Tracking Log.

ReadTrackingFileSize (Stn)
Size in bytes of the currently open Read Tracking Log.

ReadTrackingServiceCount (Stn)
Number of Read Tracking buffers processed by stone.

Object Read Log File Format

The object read log is in CSV file format, as specified by RFC 4180.

The first line of each file will contain column headers.

There are three record types: O, F, and R.

The first field of a record is a single letter, either O, F, or R, indicating the type.

- An O record includes the details on a specific read record.
- An F record marks the end of a group of O records that were in the same buffer composed by a gem.
- An R record is generated when `System class >> setObjectReadTracking:` is used, indicating dynamic disable or re-enable of object tracking.

The fields for each record are:

```
O, TimeGMT, UserName, UserProfileOop, GemProcessId,
   GemHostName, GemHostIpAddress, GemClientIpAddress,
   ObjectReadOop, ObjectReadClassOop, ObjectReadClassName
R, TimeGMT, UserName, UserProfileOop, GemProcessId,
   GemHostName, GemHostIpAddress, GemClientIpAddress, empty,
   empty, ReadTrackState
F, TimeGMT, UserName, UserProfileOop, GemProcessId
```

The following fields are defined:

`TimeGMT` (Integer)
Timestamp in GMT of the object read operation, updated when a gem composes the first O record of a session, when an R record is written, and when the F record is composed by a gem to flush a buffer.

`UserName` (String)
The GemStone user name of the user performing the read, UTF8 encoded and enclosed in double quotes. Taken from the userId instance variable of the session's UserProfile object.

`UserProfileOop` (Integer)
The OOP of the user profile.

`GemProcessId` (String)
The process ID of the session.

`GemHostName` (String)
The name of the host on which the gem is running, the result of calling `hostname()`. This field is enclosed in double quotes.

`GemHostIpAddress` (String)
The IP Address of the gem's end of the GCI client to gem connection; the result of calling `getsockname()` on the socket for the GCI client to gem connection. Does not do a reverse DNS lookup. For linked sessions, such as topaz -l, this field is empty.

`GemClientIpAddress` (String)
The IP Address of the gem's client's host, the result of calling `getpeername()` on the socket which is the gem's end of the GCI client to gem connection. For linked sessions, such as topaz -l, this field is set to `gcilnk`.

`ObjectReadOop` (Integer)
The OOP of the object read by the user.

`ObjectReadClassOop` (Integer)
The OOP of the class of the object read.

`ObjectReadClassName` (String)
The name of the class, UTF8 encoded.

`ReadTrackState` (String)
Either '`True`' or '`False`'

UserProfile userId permissible Character limitations

The UserIds of UserProfiles are limited to Strings containing Characters with values < 256.

Newly added UserProfiles are also now required to have all Characters within the String be alphanumeric, _ (underscore, code point 95), - (hyphen, code point 45), . (dot, code point 46), or space (code point 32).

Existing UserIds with other characters (this is expected to be unlikely), will continue to behave as before; however, if such UserIds exist in GemStone, sending `GsObjectSecurityPolicy >> trackReads:` will error.

LdapDirectoryServer now supports TLS options

The following instance variables have been added to LdapDirectoryServer, allow you to control the LDAP TLS options from GemStone rather than from LDAP configuration files. This is necessary when the process is in setuid mode.

baseDN
Specifies the default base DN to use when performing ldap operations. The base DN must be specified as a Distinguished Name in LDAP format.

tlsCaCert
Specifies the file that contains certificates for all of the Certificate Authorities the client will recognize.

tlsCaCertDir
Specifies the path of a directory that contains Certificate Authority certificates in separate individual files. tlsCaCert is always used before tlsCaCertDir.

tlsCert
Specifies the file that contains the client certificate.

tlsKey
Specifies the file that contains the private key that matches the certificate stored in the tlsCert instance variable. The private key must not have a passphrase.

tlsReqCert
A symbol which specifies what checks to perform on server certificates in a TLS session, if any. The following symbols are recognized:

#never - The client will not request or check a server certificate.
#allow - The server certificate is requested. if not provided or if the certificate is invalid, it is ignored and the session proceeds normally.
#try - the server certificate is requested. If not provided, the session proceeds normally. If a bad certificate is provided, the session is immediately terminated.
#demand (the default) - the server certificate is requested; if not provided or if the certificate is invalid, the session is immediately terminated.

Added LdapDirectoryServer methods

The following methods have been added:

`LdapDirectoryServer >> validatePassword:` *aPassword* `forUserId:` *aUserId* `withBaseDn:` *aBaseDn* `filterDn:` *aFilterDn*
Use the receiver LdapDirectoryServer to validate the given userId and password. See the method comments for examples on using this method for validation.

`LdapDirectoryServer class >> basicNewWithUri:` *uri* `bindDN:` *aBindDn* `password:` *password* `baseDN:` *baseDn* `tlsCaCert:` *caCert* `tlsCert:` *cert* `tlsKey:` *key* `tlsReqCert:` *aSymbol*
Creates a new instance of the receiver but does not add it to the list of LdapDirectoryServer objects used to authorize logins.

`LdapDirectoryServer class >> newWithUri:` *uri* `bindDN:` *aBindDn* `password:` *password* `baseDN:` *baseDn* `tlsCaCert:` *caCert* `tlsCert:` *cert* `tlsKey:` *key* `tlsReqCert:` *aSymbol*
Creates a new instance of the receiver and adds the resulting object to the list of LdapDirectoryServer objects used to authorize logins.

`LdapDirectoryServer class >> testConnectionToServer:` *uri* `bindDN:` *aBindDn* `password:` *password* `baseDN:` *baseDn* `tlsCaCert:` *caCert* `tlsCert:` *cert* `tlsKey:` *key* `tlsReqCert:` *aSymbol*
Attempts to perform a bind using *aBindDn* and *password* to the LDAP server specified by *uri*. Also sets the TLS credentials, if those arguments are not nil. Returns true if the connection was successful, otherwise returns false.

Other Added Methods

`Behavior >> recompileMethodAt:` *aSelector*

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

`AppendStream >> collection`
Similar to `AppendStream >> contents`, but does not clear the underlying collection.

`AppendStream >> position`
The current size of the underlying collection.

`PositionableStreamLegacy >> collection`
The equivalent to `contents`, for compatibility

Print peer information in log when invalid data on listening socket

When invalid data is received on a listening socket (in the Stone, NetLDI, or Shared Page Cache Monitor), details about the peer sending the invalid data is printed in the log file.

Multi-threaded warnings on cache slot use now include operation name

MT operations that request more cache slots than are available print a warning message to stdout and the Stone log. These messages now include the operations, such as `mfc` or `listinstances`.

Improved printing for page cache faults

Additional printing is done, including more page state for the page causing the fault and for other system pages.

Removed Methods

The following methods have been removed as part of the fixes for passivate, described in on page 27:

```
GsObjectSecurityPolicy >> basicLoadFromNoRead:
Object >> basicLoadFromNoRead:
Repository >> basicLoadFromNoRead:
```

The following private methods have been removed:

```
Object >> _idxForCompareEqualToDoubleByteSymbol:collator:
Object >> _idxForCompareEqualToQuadByteSymbol:collator:
Object >> _idxForCompareEqualToSymbol:collator:
GsProcess class >> anonymousSelectors
```

Enhancements to FFI/CByteArray

New functionality has been added to CByteArray as well other FFI classes.

Extracting a string from a CByteArray

`CByteArray >> stringFrom:` *zeroBasedStart*
return a new String containing the bytes of the receiver from specified start byte to the first byte preceding a zero byte. Interprets the data starting at &body[*zeroBasedStart*] as NUL terminated char* data, and returns an instance of String containing that data.

`CByteArray >> strlenFrom:` *zeroBasedStart*
return result of strlen() starting from the byte of receiver specified by *zeroBasedStart*.

`CByteArray >> decodeUTF8from:` *zeroBasedStart* `to:` *zeroBasedEnd* `unicode:` *unicodeBoolean*
Decode the UTF8 encode bytes of the receiver from *zeroBasedStart* to *zeroBasedEnd*; if *unicodeBoolean*==false returns a String, DoubleByteString or QuadByteString, if *unicodeBoolean*==true returns a Unicode7, Unicode16 or Unicode32. If *zeroBasedEnd* == -1, strlen() is used starting from *zeroBasedStart* to determine the number of bytes to be decoded.

Extracting a string from a CPointer

`CPointer >> decodeFromUTF8ToString`
Decode the NUL terminated UTF8 data starting at self memoryAddress and return a String, DoubleByteString or QuadByteString.

`CPointer >> decodeFromUTF8ToUnicode`
Decode the NUL terminated UTF8 data starting at self memoryAddress and return a Unicode7, Unicode16 or Unicode32.

`CPointer >> stringFromCharStar`
Return a String from the char* data starting at self memoryAddress.

`CPointer >> utf8FromCharStar`
Return a Utf8 from the char* data starting at self memoryAddress.

CByteArray added instance creation methods

The following added methods allow you to create a CByteArray based on an Array of strings or integers.

`CByteArray class >> fromArrayEncodeUtf8:` *arrayOfStrings*
`extraNullPointer:` *addExtraNull*
Takes an array of Strings objects and creates a C array of pointers to UTF8 encoded bytes. Each element of *arrayOfStrings* must be a String or MultiByteString. Space for a NUL character for each element is included in the total. If *addExtraNull* is true, an extra NULL pointer is appended to the list of pointers to terminate the list.

`CByteArray class >> fromArrayOfByteObjects:` *arrayOfByteObjs*
`extraNullPointer:` *addExtraNull*
Takes an array of byte objects and creates a C array of pointers to bytes. Each element of *arrayOfByteObjs* must be a String; note that DoubleByteString or QuadByteString are not allowed. Space for a NUL character for each element is included in the total. If *addExtraNull* is true, an extra NULL pointer is appended to the list of pointers to terminate the list.

`CByteArray class >> fromArrayOfInt32:` *arrayOfInt32s*
Takes an array of 32-bit ints and creates a C array.

`CByteArray class >> fromArrayOfInt64:` *arrayOfInt64s*
Takes an array of 64-bit ints and creates a C array.

Importing an Array into a CByteArray

The following methods import the contents of an Array into a CByteArray

`CByteArray >> addArrayOfByteObjects:` *arrayOfByteObjs* `extraNullPointer:` *addExtraNull*
Write the elements of *arrayOfByteObjs*, which must be a kinds of String (single-byte), to the receiver, starting at offset 1 and overwriting existing contents. Signal an error if any of the elements of *arrayOfByteObjs* contain codePoint zero.

`CByteArray >> addArrayOfInt32:` *arrayOfInt32*
Write the elements of *arrayOfInt32* to the receiver, starting at offset 1 and overwriting existing contents.

`CByteArray >> addArrayOfInt64:` *arrayOfInt64*
Write the elements of *arrayOfInt64* to the receiver, starting at offset 1 and overwriting existing contents.

`CByteArray >> addUtf8Encoded:` *arrayOfStrings* `extraNullPointer:` *addExtraNull*
Write the elements of *arrayOfStrings*, which must be a kinds of String or MultiByteString, to the receiver, starting at offset 1 and overwriting existing contents. Signals an Error if any of the elements of *arrayOfStrings* contain codePoint zero.

Adding a UTF8-encoded string to a CByteArray

`CByteArray >> encodeUTF8From:` *anObject* `into:` *zeroBasedDestOffset* `allowCodePointZero:` *zeroBoolean*
anObject may be a String or MultiByteString. The codepoints in *anObject* are encoded into UTF8 and the resulting UTF8 bytes are copied into the receiver starting at *zeroBasedDestOffset*. If *zeroBoolean*==false, signals an Error if *anObject* contains codePoint zero. Returns a SmallInteger, possibly zero, the number of UTF8 bytes copied into the receiver. There is no terminating NUL byte written to the destination.

Computing required CByteArray space for the contents of an Array

The following added methods compute the space required for a CByteArray that would be created from an Array of strings or integers.

`CByteArray class >> computeSizeForArrayOfByteObjects:` *arrayOfByteObjs* `extraNullPointer:` *addExtraNull*
Compute the number of bytes needed to contain *arrayOfByteObjs* in an instance of the receiver which will represent an array of bytes in C. Each element of *arrayOfByteObjs* must be a kind of String (MultiByteStrings are not allowed). Space for a NULL character for each element is included in the total. If *addExtraNull* is true, space for an additional NULL element in the array of C pointers will be included.

`CByteArray class >> computeSizeForArrayOfInt32:` *arrayOfInt32*
Compute the number of bytes needed to contain *arrayOfInt32* in an instance of the receiver which will represent an array of bytes in C.

`CByteArray class >> computeSizeForArrayOfInt64:` *arrayOfInt64*
Compute the number of bytes needed to contain *arrayOfInt64* in an instance of the receiver which will represent an array of bytes in C.

`CByteArray class >> computeSizeForArrayOfUtf8Encoded:` *arrayOfStrings* `extraNullPointer:` *extraNullBoolean*
Compute the number of bytes needed to contain *arrayOfStrings* in an instance of the receiver which will represent an array of bytes in C. Elements in *arrayOfStrings* must be kinds of String or MultiByteString. Space for a NULL character for each element is included in the total. If *addExtraNull* is true, space for an additional NULL element in the array of C pointers will be included.

`MultiByteString >> sizeForEncodeAsUTF8`

`String >> sizeForEncodeAsUTF8`

`Utf8 >> sizeForEncodeAsUTF8`

Handling code point zero

The C primitive that handles CByteArray copy now can detect if the argument contains NUL characters, which may or may not be allowed depending on the contents of the CByteArray. The following is the new method:

`CByteArray >> copyBytesFrom:` *anObject* `from:` *oneBasedStart* `to:` *oneBasedEnd* `into:` *zeroBasedDestOffset* `allowCodePointZero:` *zeroBoolean*
Copy the specified bytes of *anObject* into the receiver at the given offset. *anObject* may be any byte format object or a CByteArray. Returns number of bytes copied (possibly zero).

Other Added FFI methods

`CPointer >> isNull`
Answer true if the receiver is a NULL pointer, false otherwise.

`CPointer >> notNull`
Answer true if the receiver is not a NULL pointer, false otherwise.

`GsFile >> filePointer`
Returns a CPointer representing the underlying FILE * in C. Valid for open server files only. Returns nil if the file is compressed, on the client, or if an error occurs. Care must be taken to not access the result of this method after the receiver has been closed.

All thread-safe GCI functions now available in GciTsLibrary

GciTsLibrary dynamically creates a list of callouts to the thread-safe GCI functions. Previously, only a limited list of functions was provided; now, all are included.

Added cache statistics

In addition to Object Read Tracking statistics described on page 19, the following statistics have been added:

ObjectsSelectiveAborted (Gem)
Number of invocations of objects rolled back by selective abort since start of session.

PrimSelectiveAbortCount (Gem)
Number of invocations of Object>>_primitiveSelectiveAbort since start of session.

Added Errors

GS_ERR_READ_TRACKING_FULL/4020
When using Object Read Tracking, Error when all STN_OBJECT_READ_LOG_DIRECTORIES are full.

GS_ERR_INVALID_TRANLOG_RECORD / 4021
An invalid tranlog record was received by the Stone.

ERR_SshSocketError/2759
Reserved to support features in future releases.

ERR_AwsError/2760
Reserved to support features in future releases.

ERR_PostgresError/2761
An exception signaled when Postgres operations fail; for use by GemConnect for Postgres.
A new class, PostgresError, has been added to the image to support this error.

Bugs Fixed

The following bugs in v3.6.1 are fixed in v3.6.2.

Invalid record may be written to tranlog

Under some conditions, a Gem may attempt to write an invalid record kind 0; in the observed case, it should have been a BEGIN_DATA. Now, if such a case occurs, the Gem will get a fatal error (see "GS_ERR_INVALID_TRANLOG_RECORD / 4021" on page 26), and the data is not written to the tranlog. (#49715)

Issue with incremental tranlogging

A case was observed in which an entry in an NSC was not recorded in the transaction log when STN_TRAN_INCREMENTAL_LOGGING=true. (#49713)

SPC Monitor crash with SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_SIZE_MB=0

With the settings:

```
SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_SIZE_MB=0;
SHR_PAGE_CACHE_LARGE_MEMORY_PAGE_POLICY=2;
```

On stone startup, the SPC Monitor crashes with a SIGFPE (divide by zero). (#49737)

Native code errors in some cases using Array constructors {}

For some uses of the {} Array constructor syntax using nested blocks, with native code enabled, the Gem could encounter a SEGV or endless loop in methodLookupCache. (#49712)

Incorrect results from Object >> passivate

Passivation of complex objects produced incorrect results in a few cases.(#49566)

If an instance of ScaledDecimal, Fraction, or FixedPoint has an instance variable containing a LargeInteger, and if that instance of ScaledDecimal, Fraction or FixedPoint appears more than once in the object graph, the passivation of the second reference is incorrect, and the reactivated object will contain an incorrect value/class.

When an Object's closure contains ExecBlocks (such as SortedCollections, which contain a sortBlock), the source string for the block is included in the passivated form. If in-memory GC occurs while passivating, the sourceString may be flushed and not included in the passivated form, which creates an error on activation.

File descriptor leak fetching host statistics on Linux

Fetching statistics programatically (such as using `System fetchSystemStats`) leaks three file descriptors per execution. This can cause systems to run out of file descriptors. (#49570)

startstone issues if STN_FREE_SPACE_THRESHOLD space is unavailable

On stone startup, if the amount of free space was less than the setting for STN_FREE_SPACE_THRESHOLD, the startstone failed to complete. (#49417)

copydbf fails on decryption of encrypted extent that was pregrown or restored

When an encrypted extent is pregrown during stone startup, or a backup is restored into an encrypted extent and the extent grows, the unused space is initialized to zeros. If this encrypted extent is then decrypted, these pages of zeros are incorrect identified as garbage; and the copydbf operation that is performing the decryption crashes. (#49683)

Tranlog restore may encounter illegal store error

When restoring a transaction log into a restored database or a hot standby, there is a risk that the replay of a selective abort in the tranlog, under certain conditions may cause an illegal store error, which terminates the restore process. (#49578)

Epoch garbage collection failure may leave Admin GcGem in transaction

While Epoch GC runs, the AdminGem sets itself to be in automatic transaction mode. If Epoch fails to start, for example if there are not enough available process slots in the shared page cache, it did not correctly reset its transaction mode to #manualBegin, and thus remained in transaction. (#49657)

Upgrade from 3.5.6 to 3.6.x broken

Due to changes in LdapDirectoryServer in v3.5.6, upgrade from 3.5.6 to 3.6 or 3.6.1 fails. 3.5.5 or earlier, which do not contain the changes to LdapDirectoryServer, can successfully upgrade to 3.6x. (#49616)

GsFile issues

GsFile position incorrect for r+ file mode with atEnd

If a GsFile was open for read/write, mode r+ (for example, `GsFile class >> openUpdateOnServer:`), then after invoking `atEnd`, the file position is incorrect; `GsFile >> position` will return 0. Subsequent writes will be in the correct location.

However, if `GsFile >> position:` is sent, followed by `atEnd`, a subsequent write may not be in the correct location. (#49568)

Failure in OS call from GsFile write primitives potential to SEGV

Within the GsFile write primitives, if an OS write call returned a negative value due to an unexpected write error, the code did not immediately return. Further execution could cause a SEGV. (#49540)

Shared Page Cache could have run out of slots

With a large database and a default shared page cache size, it was possible for an operation such as MFC to run out of PCEs (page cache entries), causing the Stone to shut down. (#49582)

objectaudit/pageaudit do not correctly handle some kinds of error state

There are possible cases of internal corruption, that objectAudit and pageaudit do not handle correctly. objectAudit may fail to complete (#49619), and pageaudit may SEGV (#49617).

Extreme log growth attempting to printing stack if stack corrupted

Using kill -USR1 to print stacks to the process log file, in some case where the stack is corrupted, may have entered a loop printing `[ incomplete frame, send in progress ]` to the log file, potentially filling up the disk space. (#49655)

Risk of C memory corruption on in-memory GC in libicu operation

There is a risk that if an in-memory GC occurs while performing a libicu String operation, it may corrupt C memory. (#49521)

GsSocket >> readWillNotBlockWithin: does not handle nil returned by readWillNotBlock

It is possible for `GsSocket >> readWillNotBlock` to return a nil, which resulted in an error in `GsSocket >> readWillNotBlockWithin:`. (#49678)

Multi-threaded scan issues

GC state change during multithreaded scan fails to report error

When voting or an atomic promote occurs while a multithreaded scan such as listInstances is in process, the results of the multithreaded scan are no longer reliable. This previously returned an empty collection; now, an error is signaled. The multithreaded scan operation can be executed again to get the results. (#49651)

Multi-threaded operations could use up all available process slots

When a multi-threader operation such as markForCollection was requested with more threads than currently available process slots in the repository, previously the operation executed with as many threads as available process slots. This prevented other sessions from logging in, including stopstone. In addition, if the multi-threaded operation was expected to run with a large number of threads but only a small number of slots was available, it would run with those few threads, unexpectedly taking much longer to complete. (#49611)

Now, if there are not enough process slots available, the requested operation is executed with half as many threads as requested.

If this is still more than is available, an error is reported, allowing you to determine the reason for unavailable slots and correct the problem, or retry with a smaller number of threads.

Issues related to Symbol Garbage Collection

Cumulative algorithm changes in Symbol GC

Symbol Garbage Collection has undergone a number of internal changes in the algorithm used to ensure behavior is correct for large sets of dead symbols in active repositories, with additional validation on large customer repositories. Note that the documentation describing the internal algorithms used is no longer correct with respect to current behavior.

Symbol GC slow with many dead symbols

When Symbol GC found a very large number of possible dead symbols, the write set union sweep became unreasonably slow. This was a result of the possibleDeadSymbol structure containing large collision buckets; the objects referenced from the collision buckets were being read, which is unnecessary, since all contained objects would be symbols. (#49504)

Logsender not connected to stone could have transmitted incomplete tranlog records

In a hotstandby system, when the logsender is not connected to the master Stone, the logsender relies on disk file information to check for when a transaction log record is ready to send to the logreceiver. However, logical tranlog records may cross a file system block boundary, and this not detected by the logsender's disk file checks, so an incomplete transaction log record may have been sent. (#49450)

kernel.yama.ptrace_scope=1 disallowed stacks with NetLDI in authentication mode

When kernel.yama.ptrace_scope=1, **gdb** is disallowed from attaching to a process. This means that GemStone's **pstack** utility, which is a wrapper for **gdb**, cannot be used to get stack traces. For systems in which the NetLDI is running in guest mode, on `kill -USR1` *pid* or on fatal error, the GemStone process can still print stacks to the log file.

However, on systems with the NetLDI is in authentication mode, the real and effective UserIDs of a process are not the same. This prevented the GemStone process itself from getting a stack, so no useful stacks could be printed to the process log. (#49592)

Page read error message unhelpful

The Error message for a failed read (4009) did not provide information about the page or extent, and was not specific about if a pageserver was involved. (#49613)

Cache warming may have excessive impact on page use

Following cache warming, the cache may be substantially full, which may result in some sessions in FramesFromFindFree. Now, the frame ages for frames holding warmed pages, have been adjusted to ensure greater page availability. (#49629)

FFI-related issues

CCallout string arguments allowed MultiByteString, disallowed Utf8

When invoking a CCallout method and specifying arguments of type `char*` or `const char*`, the checking for suitable GemStone classes was not correct. (#49557)

- These methods allow arguments of type String; MultiByteString was previously accepted, but is now disallowed.
- Arguments of type Utf8 are now allowed.

CDeclaration doesNotUnderstand: #'cByteArraySpecies'

The message sent to flexibly handle CByteArray classes was incorrect. (#49599)

CByteArray >> floatAt:put: did not accept SmallFloat arguments

The method `CByteArray>>floatAt:put:` errored with an argument that was a SmallFloat (#49493). While SmallFloat is largely deprecated in favor of SmallDouble, it is still used in the FFI when interfacing to libraries which take 8-bit floating points.

LDAP issues

GemStone links the OpenLDAP libraries to provide GemStone authentication directly via LDAP. A number of fixes and improvements to the GemStone API to LDAP are included in this release.

Note that GemStone UNIX authentication, which goes through PAM, may also use LDAP is PAM is configured to use LDAP, but this configuration is outside of GemStone; the changes described here only affect UserProfiles that are configured with authentication mode of #ldap; or calls to LdapDirectoryServer functions from within GemStone to authenticate passwords.

LDAP did not work for processes in setuid mode

If the process is in setuid mode (that is, the real and effective UNIX userIds of a process are not the same), LDAP does not read environment variables, nor files in the $HOME directory, and thus was not able to set all required information to connect to an LDAP server. (#49498)

With a setuid mode process, such as a Gem with the s bit set, you will need to use the new LdapDirectoryServer **tls*** instance variables within GemStone Smalltalk to set the certificate requirement, and certificates if necessary, that allow connection to the LDAP server. See “LdapDirectoryServer now supports TLS options” on page 20.

GemStone's LDAP library did not correctly set directory for ldap.conf

GemStone's LDAP library, `libldap-`*N*`.`*N*`.`*N*`-64.so` (depending on version), looked for `ldap.conf` in the installation directory, $GEMSTONE, which did not follow LDAP conventions. (#49515)

Now, it will look for `/etc/openldap/ldap.conf`. This setting can be overridden by environment variables, or by files `.ldaprc` or `ldaprc`. Note that the specific LDAP search sequence is distribution dependent; this is the RedHat/CentOS convention. See the ldap.conf man page for your distribution for details.

LDAP did not support reset of TLS credentials

The binding to TLS credentials in LDAP could not be done more than once; so changing TLS credentials required restarting the process. (#49499)

ClassOrganizer >> referencesToLiteral: could miss literal symbol

If a literal symbol is present in a literal array, the method `ClassOrganizer >> referencesToLiteral:` did not find or return that reference. (#49672)

System _add:to: called missing method

The deprecated System hidden set API method `_add:to:` invoked a deprecated method that had been inadvertently removed, and errored (#49665). While this functionality has been restored, users are encouraged to migrate to GsBitmap.

User Action library load may fail with missing extension

When loading a UserAction library via `System class>>loadUserActionLibrary:`, the specified filename may usually omit the extension; that is, use *libraryName* rather than *libraryName*`.so`. However, the extension was required for library names with GemStone-style formatting (the format, for example, of libgbslnk-3.6.0-64.so).

Loading a user action library with a filename of the pattern *name*-*version*-*bits*`.so` would fail if the `.so` extension was omitted. (#49512)

Debugging issues with stack trimming with blocks

Issues have been found related to stack trimming while in a block. Stack trimming is done by the Topaz **stack trim** command, or `GsProcess>>_trimStackToLevel:` (used by debuggers). Specific issues are stack trim may fail within nested blocks (#49695), and Stack trimming within a loop may reset the state of temporaries to the value previous to the correct state. (#49694)