

---

GemStone®

# *GemStone/S 64 Bit Release Notes*

Version 2.2

April 2007

GEMSTONE<sup>™</sup> S 64

---

## INTELLECTUAL PROPERTY OWNERSHIP

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

This documentation, or any part of it, may not be reproduced, displayed, photocopied, transmitted, or otherwise copied in any form or by any means now known or later developed, such as electronic, optical, or mechanical means, without express written authorization from GemStone Systems, Inc.

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 GemStone Systems, Inc. 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 GemStone Systems, Inc.

This software is provided by GemStone Systems, Inc. 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 GemStone Systems, Inc. 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

This software product, its documentation, and its user interface © 1986-2007 GemStone Systems, Inc. All rights reserved by GemStone Systems, Inc.

## PATENTS

GemStone is covered by U.S. Patent Number 6,256,637 "Transactional virtual machine architecture", Patent Number 6,360,219 "Object queues with concurrent updating", and Patent Number 6,567,905 "Generational Garbage Collector". GemStone may also be covered by one or more pending United States patent applications.

## TRADEMARKS

**GemStone**, **GemBuilder**, **GemConnect**, and the GemStone logos are trademarks or registered trademarks 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.

**Sun**, **Sun Microsystems**, **Solaris**, and **SunOS** are trademarks or registered trademarks of Sun Microsystems, Inc. All **SPARC** trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. **SPARCstation** is licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

**HP** and **HP-UX** are registered trademarks of Hewlett Packard Company.

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

**Microsoft**, **MS**, **Windows**, **Windows 2000** and **Windows XP** 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.

**AIX** and **POWER4** are trademarks or registered trademarks of International Business Machines Corporation.

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. All terms mentioned in this documentation that are known to be trademarks or service marks have been appropriately capitalized to the best of our knowledge; however, GemStone 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.

**GemStone Systems, Inc.**  
1260 NW Waterhouse Avenue, Suite 200  
Beaverton, OR 97006

---

## *Preface*

### **About This Documentation**

These release notes describe changes in the GemStone/S 64 Bit version 2.2 release. We recommend that everyone migrating to this version read these release notes before beginning installation, testing or development.

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

These documents are also available on the GemStone customer website, as described below.

### **Terminology Conventions**

This document uses the following terminology:

The term “GemStone” is used to refer both to the product, GemStone/S 64 Bit, or previous GemStone/S server products; and to the company, GemStone Systems, Inc.

### **Technical Support**

GemStone provides several sources for product information and support. The product-specific manuals and online help provide extensive documentation, and should always be your first source of information. GemStone Technical Support engineers will refer you to these documents when applicable.

**GemStone Web Site:** <http://support.gemstone.com>

GemStone’s Technical Support website provides a variety of resources to help you use GemStone products. Use of this site requires an account, but registration is free of charge. To get an account, just complete the Registration Form, found in the same location. You’ll be able to access the site as soon as you submit the web form.

The following types of information are provided at this web site:

**Help Request** allows designated support contacts to submit new requests for technical assistance and to review or update previous requests.

**Documentation** for GemStone/S 64 Bit is provided in PDF format. This is the same documentation that is included with your GemStone/S 64 Bit product.

**Release Notes** and **Install Guides** for your product software are provided in PDF format in the Documentation section.

**Downloads** and **Patches** provide code fixes and enhancements that have been developed after product release. Most code fixes and enhancements listed on the GemStone Web site are available for direct downloading.

**Bugnotes**, in the Learning Center section, identify performance issues or error conditions that you may encounter when using a GemStone product. A bugnote describes the cause of the condition, and, when possible, provides an alternative means of accomplishing the task. In addition, bugnotes identify whether or not a fix is available, either by upgrading to another version of the product, or by applying a patch. Bugnotes are updated regularly.

**TechTips**, also in the Learning Center section, provide information and instructions for topics that usually relate to more effective or efficient use of GemStone products. Some Tips may contain code that can be downloaded for use at your site.

**Community** provides customer forums for discussion of GemStone product issues.

Technical information on the GemStone Web site is reviewed and updated regularly. We recommend that you check this site on a regular basis to obtain the latest technical information for GemStone products. We also welcome suggestions and ideas for improving and expanding our site to better serve you.

You may need to contact Technical Support directly for the following reasons:

- ▶ Your technical question is not answered in the documentation.
- ▶ You receive an error message that directs you to contact GemStone Technical Support.
- ▶ You want to report a bug.
- ▶ You want to submit a feature request.

Questions concerning product availability, pricing, keyfiles, or future features should be directed to your GemStone account manager.

When contacting GemStone Technical Support, please be prepared to provide the following information:

- ▶ Your name, company name, and GemStone/S license number
- ▶ The GemStone product and version you are using
- ▶ The hardware platform and operating system you are using
- ▶ A description of the problem or request
- ▶ Exact error message(s) received, if any

Your GemStone support agreement may identify specific individuals who are responsible for submitting all support requests to GemStone. If so, please submit your information through those individuals. All responses will be sent to authorized contacts only.

For non-emergency requests, the support website is the preferred way to contact Technical Support. Only designated support contacts may submit help requests via the support website. If you are a designated support contact for your company, or the designated contacts have changed, please contact us to update the appropriate user accounts.

**Email: support@gemstone.com**

**Telephone: (800) 243-4772 or (503) 533-3503**

Requests for technical assistance may also be submitted by email or by telephone. We recommend you use telephone contact only for more serious requests that require immediate evaluation, such as a production system that is non-operational. In these cases, please also submit your request via the web or email, including pertinent details such as error messages and relevant log files.

If you are reporting an emergency by telephone, select the option to transfer your call to the technical support administrator, who will take down your customer information and immediately contact an engineer.

Non-emergency requests received by telephone will be placed in the normal support queue for evaluation and response.

## 24x7 Emergency Technical Support

GemStone 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, if they encounter problems that cause their production application to go down, or that have the potential to bring their production application down. For more details, contact your GemStone account manager.

## Training and Consulting

Consulting and training for all GemStone products are available through GemStone's Professional Services organization.

- ▶ Training courses are offered periodically at GemStone's offices in Beaverton, Oregon, or you can arrange for onsite training at your desired location.
- ▶ Customized consulting services can help you make the best use of GemStone products in your business environment.

Contact your GemStone account representative for more details or to obtain consulting services.



## **Chapter 1. GemStone/S 64 Bit 2.2 Release Notes**

Overview . . . . .	1-1
Changes and New Features . . . . .	1-1
Stone optimizations . . . . .	1-1
Segments added . . . . .	1-2
Larger signal buffer for gem-to-gem signals . . . . .	1-3
Linux supports faster performance using optimized POSIX Asynchronous I/O . . . . .	1-3
AIX requires POWER5 Processor . . . . .	1-4
Signal when transaction logs are full . . . . .	1-4
Change in Stop session behavior . . . . .	1-4
New lock - Application Write Lock . . . . .	1-4
New extended character set support . . . . .	1-5
RcQueue improvements . . . . .	1-5
changeClassTo: restrictions removed . . . . .	1-6
Locale Added . . . . .	1-6
Seaside support . . . . .	1-7
Session Specific Methods . . . . .	1-7
Monticello . . . . .	1-8
Compiler change in interpretation of the ‘_’ character . . . . .	1-8
New DbTransient attribute . . . . .	1-8
Reduced conflict retries now retry multiple times . . . . .	1-9
Indexing Changes . . . . .	1-9
Reduced Conflict Equality indexes . . . . .	1-9
Index audit now public . . . . .	1-10
Some milliseconds no longer roll over at 524287999 . . . . .	1-10
Tranlog sizes larger than raw partition sizes corrected at startup . . . . .	1-10
More information provided for error 4059 . . . . .	1-10
Obsolete method deleted . . . . .	1-10

Ability to store DoubleByteStrings in ByteArrays . . . . .	1-10
Sessions can now log themselves out . . . . .	1-11
Change in listInstances transaction behavior. . . . .	1-11
Topaz changes . . . . .	1-11
Hidden set changes . . . . .	1-11
Float operations now in primitives . . . . .	1-11
Number kind indexes have changed . . . . .	1-11
Changes in Errors . . . . .	1-12
Errors on Windows now include details . . . . .	1-13
GCI changes . . . . .	1-13
Configuration Parameter Changes . . . . .	1-14
Changes in Cache Statistics . . . . .	1-15
The following statistic has been renamed: . . . . .	1-15
The following statistics have been added: . . . . .	1-15
The following cache statistics have been removed: . . . . .	1-16
Bugs Fixed . . . . .	1-17
Disposing commit records caused slow stone response . . . . .	1-17
User action calls to assert() could have hung. . . . .	1-17
Duplicate FreePages cache statistics. . . . .	1-18
Out-of-memory error during indexing even with autocommit=TRUE . . . . .	1-18
Statmonitor could get stuck in loop creating files . . . . .	1-18
Methods remained where abort could lose data. . . . .	1-18
Time>>asMillisecondsGmt broken . . . . .	1-18
asFloat incorrect results for some large Integers. . . . .	1-18
Very small Floats converted to Fractions produced incorrect results . . . . .	1-18
copyFrom:to: returned error . . . . .	1-18
String>>replaceFrom:to:with:startingAt: incorrect results when replace was performed using self . . . . .	1-18
Object>>_primitiveSize reported physical rather than logical size for NSCs. . . . .	1-19
Stone log entries did not appear immediately . . . . .	1-19
AIO errors in opening or writing to stone caused stone shutdown . . . . .	1-19
Short stone freeze at checkpoint start . . . . .	1-19
makeusers script had several problems. . . . .	1-19
GsSocket>>connectTo:on: may have failed to connect . . . . .	1-19
Topaz crash on interrupt . . . . .	1-19
Soft break did not interrupt Delays . . . . .	1-19
Object >> pageCreationTime returned nil . . . . .	1-20
Could not shut down Stone when maximum sessions logged in . . . . .	1-20
_shrinkExtents was ineffective. . . . .	1-20
yearStartDst errored if no DST. . . . .	1-20
Method caches not cleared on abort. . . . .	1-20
DoubleByteString >> asNumber returned Integer when should return Float . . . . .	1-20
OldestCrSessionNotInTrans may show session in transaction. . . . .	1-20

## ***Chapter 2. Extended Character Set Support***

Character Set Representation . . . . .	2-2
Categories . . . . .	2-2
Configuring your application to use Extended Character Sets. . . . .	2-4
#CharacterDataTables . . . . .	2-4
Loading Extended Character Sets . . . . .	2-5
Image changes - Character. . . . .	2-6
Files . . . . .	2-7
The Unicode Database . . . . .	2-8
Procedures . . . . .	2-8



# *GemStone/S 64 Bit 2.2 Release Notes*

## Overview

GemStone/S 64 Bit 2.2 is a new version of the GemStone/S 64 Bit object server. This release provides a number of new features, improved performance, and fixes a number of bugs; we recommend everyone using GemStone/S 64 Bit upgrade to this new version. The details of these changes are provided in this document.

This release also provides updated documentation.

These release notes provide changes between the previous version of GemStone/S 64 Bit, version 2.1.4, and version 2.2. If you are upgrading from a version prior to 2.1.4, please also review the release notes for each intermediate release to see the full set of changes.

For details about installing GemStone/S 64 Bit 2.2 or upgrading from earlier versions of GemStone/S 64 Bit or other GemStone server products, see the *GemStone/S 64 Bit Installation Guide* for version 2.2.

## Changes and New Features

### Stone optimizations

This release includes significant improvements that have been made for performance, in particular to improve parallel work on more powerful hardware.

- ▶ the stone is now multi-threaded.
- ▶ the shared page cache monitor is also now multithreaded; slot recovery is done in a separate thread for improved response.
- ▶ Polymorphic method lookup caches have been added; method lookups to super and send that produce do-not-understand have been optimized.
- ▶ With the multithreaded stone, all gems close their inband socket to stone after establishing either SMC communication or Pgsvr-SMC communication to stone.

Note that now, the method `GsSocket>>getStandardId:inband;`, when used with a first argument of 0 to indicate the stone, will return -1 if the session is using SMC to the stone.

## Segments added

Support for Segments has been added. Segment protocol in GemStone/S 64 Bit is similar to that in GemStone/S with the following differences:

- ▶ Segments must be committed before they can be used.
- ▶ Changes to segments, authorizations, and other changes that may affect read or write authorization checks may not take effect until the next login.
- ▶ Committed Segments cannot be deleted.
- ▶ The total number of committed Segments is now limited to 65535.
- ▶ It is allowable to have a nil segment. This is equivalent to world write, but no authorization checks are performed for objects with a nil segment. Nil object segments will provide the fastest performance. Nil can now be used or returned by any segment protocol.

Repositories upgraded from earlier versions of GemStone/S 64 Bit, which did not support segments, will have all objects with a nil segment.

- ▶ New user creation has changes. Most new user creation protocol that does not include a Segment argument, will create a user with a nil default segment. However, the method `addNewUserWithId:password:` will create and commit a new Segment instance for the new user's default segment.
- ▶ login requires that the user logging in have read authorization for both `SystemSegment` and `DataCuratorSegment`.

For more detailed information on Segments and the default segment for a new UserProfiles, see the *Programming Guide for GemStone/S 64 Bit v2.2* chapter on Segments and Security, and the *System Administration Guide for GemStone/S 64 Bit v2.2* chapter on User Administration.

Note that repositories that are converted from GemStone/S 64 Bit versions 2.0.x or 2.1.x will have a Segment with id 20, which must be world write, although there will be no Segments 9...19.

The position in the hierarchy of the Repository class has changed; it is now a subclass of Collection. The previous Repository class has been renamed OldRepository.

GCI instance creation and stores are disallowed to instances of Repository and Segment.

A group that was not in GemStone/S, the DataCuratorGroup, is now provided to simplify granting privileges for Segment creation, and other restricted activities.

The following segment related methods have been added:

```
Repository >> listObjectsInSegments: anArray
```

Returns a list of objects in the receiver that have the segmentIds in *anArray*, which must be an Array of at most 2034 segmentIds. The result of this method is an Array of Arrays, where the contents of each inner array consists of objects whose segmentId is the segmentId of the corresponding element in *anArray*. The result contains both per-

manent and temporary objects. The temporary objects found may vary from run to run.

This method aborts the current transaction and scans the object header of each object in the repository.

Repository >> listObjectsInSegmentToHiddenSet: *aSegmentId*

Scans the entire repository and generates a list of objects that have specified *segmentId*, putting the results into hidden set 1. The previous state of hidden set 1 is cleared. Returns the number of instances found.

This method aborts the current transaction and scans the object header of each object in the repository.

Repository >> listObjectsInSegments: *anArray* toDirectory: *aString*

Scan the repository for objects that have the segmentIds in *anArray*, which must be an Array of at most 2034 segmentIds, and writes the results to binary bitmap files in the directory specified by *aString*. Binary bitmap files have an extension of .bm and may be loaded into hidden sets using class methods in class System. *aString* must specify a path to a writable directory.

Bitmap files are named as follows:

segment*segmentId*-objects.bm

where *segmentId* is an element of *anArray*.

The result is an Array of pairs. For each element of the argument *anArray*, the result array contains *segmentId*, *numberOfInstancesFound*. The *numberOfInstances* is the total number written to the output bitmap file.

This method aborts the current transaction and scans the object header of each object in the repository

## Larger signal buffer for gem-to-gem signals

In previous releases, only one signal at a time could be sent to a session; sending a second signal before the first signal was picked up by the receiver resulted in a runtime error on the sender.

Now, the Stone allows up to 50 pending gem-to-gem signal messages per session before the sending session will receive the buffer is full error. The limit of 50 applies to signals from all sessions, not a particular sender.

## Linux supports faster performance using optimized POSIX Asynchronous I/O

On Linux, when the optimized POSIX AIO library (libposix-aio.so) is available, the Stone process can now take advantage of this to achieve much higher commit rates. For installation instructions, see the *Installation Guide*.

If libposix-aio.so is available, it will be used automatically. If this library is not found, the regular library (librt.so) will be used instead, and messages will be printed to stdout. Performance with the regular library is comparable to previous version of GemStone/S 64 Bit.

If you want to disable use of the optimized library, use the new Linux-only -s startstone parameter.

## AIX requires POWER5 Processor

For best optimization on AIX, GemStone has been recompiled on POWER5 processors.

## Signal when transaction logs are full

When transaction log directories or partitions are full, GemStone cannot process commits, so application activity comes to a halt. To allow immediate notification so the condition can be corrected, a signal is generated on tranlog full conditions, for which a signal handler may be created. This signal is asynchronous, and is similar to the sigAbort mechanism.

In previous releases, this signal was sent to administrative sessions only and did not require specific enabling. Now, any session can be set up to receive the signal, and each session must enable receipt.

The following methods have been added:

```
System Class >> enableSignalTranlogsFull
```

Enables generation of #rtErrTranlogDirFull, 2339, to this session when the Stone detects a tranlog full condition.

```
System Class >> disableSignalTranlogsFull
```

Disables the generation of #rtErrTranlogDirFull, 2339, to this session when the Stone detects a tranlog full condition.

```
System Class >> signalTranlogsFullStatus
```

Returns true to indicate that the session will get a error #rtErrTranlogDirFull, 2339, when the Stone detects a tranlog full condition. Returns false otherwise.

## Change in Stop session behavior

When sessions are stopped using stopUserSessions or stopSession:, the Stone now waits for STN\_GEM\_TIMEOUT seconds before forcibly stopping the session. If STN\_GEM\_TIMEOUT is set to 0, it will wait for 5 minutes.

## New lock - Application Write Lock

A new type of lock has been added, the application write lock. An application write lock differs from a regular GemStone write lock in these ways:

- ▶ When you request an application write lock on an object, the request will not return until the lock is granted, or until the wait times out. This frees you from having to repeatedly request a lock if it is not immediately available. Timeout is controlled by the new configuration parameter STN\_OBJ\_LOCK\_TIMEOUT; see page 1-14 for information on this parameter. This parameter is 0 by default, meaning lock waits do not timeout.
- ▶ When you request an application write lock, you must specify a lock queue. There are two lock queues available in the stone. Once you use a lock queue to lock an object, that queue can only be used to lock that object, until the stone is stopped and restarted. Thus, you must choose no more than two objects that can be application write locked, and you must take your repository down if you want to change that choice.

- ▶ Application write locks can detect whether a request would cause deadlock, and will deny such a request.

The following method has been added:

```
System Class >> waitForApplicationWriteLock: lockObject queue: lockIdx
  autoRelease: aBoolean
```

Wait for an application write lock using the specified object and the application lock queue specified by *lockIdx*, which is either 1 or 2. For a given *lockIdx*, all calls must use a committed *lockObject*, which is registered on first use; it must be identical during the life of a stone process. If *aBoolean* is true, the lock will be cleared on commit or abort, otherwise the lock must be released manually using `removeLock`:

Returns a `SmallInteger`, one of:

1	granted	Lock granted. If <i>aBoolean</i> is true, <i>lockObject</i> was added to the hidden sets <code>CommitReleaseLocksSet</code> and <code>CommitOrAbortReleaseLocksSet</code> of this session.
2074	dirty	Lock granted. <i>lockObject</i> was written by another session since start of this transaction. If <i>aBoolean</i> is true, <i>lockObject</i> was added to the hidden sets <code>CommitReleaseLocksSet</code> and <code>CommitOrAbortReleaseLocksSet</code> of this session.
2418	deadlock	Lock not granted. Returned if this session already has a lock on <i>lockObject</i> , or multiple pending lock requests would cause deadlock.
2419	timeout	Lock not granted. Wait for lock timed out per <code>STN_OBJ_LOCK_TIMEOUT</code> config parameter.

## New extended character set support

Support for case-related operations and collation for all Character sets has been added. For details on this, see Chapter 2, "Extended Character Set Support".

## RcQueue improvements

### RcQueueEntry

A new class has been added, `RcQueueEntry`. This is a subclass of `RcQueueElement` and is used interchangeably. After upgrading, any objects added to an `RcQueue` will be `RcQueueEntry`, rather than `RcQueueElement`. There is no need to perform any modifications to your existing `RcQueues`.

The new `RcQueueEntry` class keeps two timestamps - both seconds (since 2005) and microseconds. This allows much greater precision in sequencing the objects in an `RcQueue`.

### Performance

For performance, conflicts on the `RcQueue` itself are no longer handled by reduced-conflict (RC) retry logic. This means attempts to grow the `RcQueue` while in use are likely to fail. Therefore, an `RcQueue` is no longer lazy initialized - session components are added at `RcQueue` creation time when the `new:` method is used. It is advisable, if you know the maximum number of sessions that will be using the `RcQueue`, to specify the size on creation.

Also, several methods have been reimplemented in C for performance.

### Added methods

`RcQueue >> removeIntoArray: anArray`

Removes the leading element from the receiver and stores the element in `anArray` at index 1. A `SmallInteger` representing the session ID of the session that added the element to the queue is also stored in `anArray` at index 2. If the receiver is empty, returns `nil`.

`RcQueue >> removeIntoArray`

Removes the first element from the receiver and stores the element in a new `Array` at index 1. A `SmallInteger` representing the session ID of the session that added the element to the queue is also stored in the new `Array` at index 2. If the receiver is empty, returns `nil`.

`RcQueue >> removeObject: anObject addedBySessionId: sessionId`

Removes the given object from the receiver assuming it is the oldest object added by the given session Id. Returns the object or `nil` if the object was not found or the receiver is empty.

### changeClassTo: restrictions removed

The method `changeClassTo:` previously only permitted changing the class of an instance to a subclass of its class. These restrictions have been relaxed; the new rules are:

- ▶ The receiver's class must have the same implementation as the new class (bytes, pointers, or non-sequenceable collection).
- ▶ If the new class is a kind of `IdentitySet`, then the current class must also be a kind of `IdentitySet`.
- ▶ Sets and Bags may not be changed from one class to the other, and `IdentitySets` and `IdentityBags` similarly may not be changed from one class to the other.
- ▶ The new class must not be a kernel class for which instance creation is disallowed, nor `GsMethod`, `GsMethodDictionary`, `SymbolDictionary`, or `SymbolList`.

**Please use caution in changing the class of an object; injudicious use of this method can cause problems in your application.**

### Locale Added

A class `Locale` has been added, providing Operating System locale information in GemStone. All locale information is provided; see the image and OS documentation (`man localeconv`) for more information.

GemStone currently only uses the `decimalPoint` setting, to provide localized reading and writing of numbers involving decimal points. Note that Smalltalk syntax requires the use of `'.'` as the decimal point separator, so expressions involving literal floating point numbers will still require use of the `'.'`, regardless of `Locale`.

Methods that behave differently as a result of setting `Locale` include:

```
BinaryFloat subclasses and DecimalFloat Class
  fromString:
  fromStream:
```

```
asString
asStringUsingFormat:
String asNumber
```

To override the OS locale information you may set specific Locale information using:

```
Locale Class >> setCategory: categorySymbol locale: LocaleString
```

Valid categories are:

#LC_CTYPE	locale's character handling
#LC_NUMERIC	locale's decimal point handling
#LC_TIME	locale's date and time handling
#LC_COLLATE	locale's collation setting
#LC_MONETARY	locale's monetary handling
#LC_MESSAGES	locale's messages handling
#LC_ALL	all locale categories

See the OS man page for setlocale for more information.

For example, to set GemStone to use decimal localization appropriate for Germany, use:

```
Locale setCategory: #LC_NUMERIC locale: 'de_DE'.
```

To set up to use UNIX default values:

```
Locale setCategory: #LC_ALL locale: 'C'.
```

## Seaside support

GemStone now supports Seaside. This allows design of web applications using the Seaside web development framework, backed with GemStone's object persistence.

For up to date information, see <http://seaside.gemstone.com/>.

The Seaside framework classes are provided as a goodie, in the \$GEMSTONE/pub/monticello directory.

Several features, including per-session method dictionaries and interface to the Monticello code management tool, were added to support Seaside. These features may also be used independently and are listed separately below.

## Session Specific Methods

GemStone has added support for session methods, allowing specific class and instance methods to be installed in the current session, without affecting other sessions. Each class can have session specific methods, as well as ordinary shared methods. Method lookup starts with the session specific methods of the receiver, then proceeds to shared methods of the receiver, then continues with the session specific methods of the superclass, then shared superclass methods, and so on.

The class `GsSessionMethodDictionary` has been added to support this feature. A session may install a single top level instance of `GsSessionMethodDictionary` on login, or at any time. All added session specific methods are stored in this dictionary, in sublevel `GsSessionMethodDictionaries`, keyed by Class.

Note that in this release, session specific methods do not appear listed with the class selectors and are not visible in the tools; tools support will be added in a future release.

To avoid the risk of unauthorized users modifying kernel class behavior, the methods to add session specific methods – the methods that create, modify, and install GsSessionMethodDictionaries – are protected. To allow users other than SystemUser to use session methods, shared methods must be defined by SystemUser that enter protected mode to create, modify, and install the session method dictionaries. Later, these methods can be executed by ordinary users to perform the creation, modification, and installation of the session method dictionaries. For an example, see the updated *Programming Guide for GemStone/S 64 Bit*.

Session method dictionaries are used by GemStone's implementation of the Monticello code management system. To support Monticello, the concept of Packages has been added. Each package encapsulates a set of class definitions as well as method definitions that may belong to classes that are not in the Package. This feature will be more complete and customer usable in future releases.

Added GemStone classes for initial Package support:

```
GsPackage
GsPackageLibrary
GsPackagePolicy
```

Related methods have been added to existing classes, including:

```
Behavior>>compileMethod:dictionaries:category:intoMethodDict:
    intoCategories:
GsCurrentSession >> installSessionMethodDictionary:
GsCurrentSession >> initialize
```

## Monticello

To allow connection to Monticello, the code repository used for Seaside, Monticello has been ported to GemStone. It is available as a goodie under the \$GEMSTONE/pub/monticello directory.

## Compiler change in interpretation of the ‘\_’ character

The ‘\_’ character, separated by whitespace, is now interpreted by the compiler as an assignment operator. It is no longer permitted to use the underscore character alone as a method name.

This change allows code ported from Squeak, specifically the Seaside framework, to run with fewer changes. Squeak recognizes both ‘\_’ and ‘:=’ as assignment operators. We do not recommend using ‘\_’ as an assignment operator.

## New DbTransient attribute

A class can now have the new attribute DbTransient. This is designed to be similar to Java “transient” variable attribute. The instance variables of instances of a class that is DbTransient are not committed, but remain local to the session. This allows you to reference objects that should not be persistent - such as semaphores - within data structures that are persistent and shared.

When a new DbTransient object (an instance of a class with the DbTransient attribute) is committed, its instance variables are written to the repository as nil. Whenever a DbTransient object is read into a session from the repository, all of its instance variables are nil.

Since DbTransient instances are stored only in memory, they require some special handling. A DbTransient object that has been committed can, like other committed objects, be dropped from the session's private memory if memory becomes low. Like other committed objects, a DbTransient object will be re-read from the repository on demand. Unlike other committed objects, a DbTransient object's instance variables will be nil after a re-read. To prevent losing non-nil instance variable values, you should keep a reference to each DbTransient object from session state. Any object reachable from session state will not be dropped from memory.

Since a DbTransient object will remain in memory while referenced from session state, the reference should be removed when the DbTransient object is no longer needed, to avoid filling up memory and causing an out of memory error.

Note that while DbTransient objects are only committed once (on creation), and so do not normally cause concurrency conflicts, if they are clustered the object will be written (still with all instance variables nil), and could potentially cause a concurrency conflict.

To make a class DbTransient, send:

```
aClass makeInstancesDbTransient
```

*aClass* must be a non-indexable pointer classes. This will cause any instance of *aClass* to be DbTransient. The change takes place immediately. Sending:

```
aClass makeInstancesNotDbTransient
```

will cause instances to be non-DbTransient, that is, allow instance variables to be written to disk. To determine if a class's instances are DbTransient, send:

```
aClass instancesDbTransient
```

## Reduced conflict retries now retry multiple times

Reduced conflict logic provides the capability of retrying operations that failed to commit. Previously, there was one retry before reporting the failure to the session (internal logic provided for three retries, but only one retry was being attempted).

Now, the system will attempt to retry 15 times before reporting failure. A new commit result, #retryLimitExceeded, is returned in this case.

To avoid race conditions, the retries are serialized using a new type of lock, the RcWriteLock. This lock uses an instance of Object, which is in Globals at #GemStoneRCLock. Use of the lock object may be disabled by setting (Globals at: #GemStoneRCLock) to nil; this requires SystemUser privileges.

To support the lock, the method `System Class >> waitForRcWriteLock: rcLockObject` has been added. This is not intended for general use. If any other sessions have locked *rcLockObject*, this method will wait for it to be released before returning; it will time out according to the configuration parameter `STN_OBJ_LOCK_TIMEOUT`.

## Indexing Changes

### Reduced Conflict Equality indexes

A new type of index, a Reduced Conflict Equality Index, has been added. This allows you to avoid some index maintenance related commit conflicts. To create a reduced-conflict equality index, use the following methods:

```
UnorderedCollection >> createRcEqualityIndexOn:  
UnorderedCollection >> createRcEqualityIndexOn:withLastElementClass:
```

To support Rc indexes, the following classes have been added:

```
RcBtreeBasicInteriorNode  
RcBtreeBasicLeafNode  
RcBtreeInteriorNode  
RcBtreeLeafNode  
RcRangeEqualityIndex
```

IdentityIndexes already used reduced conflict support classes, and are unchanged.

### Index audit now public

The `UnorderedCollection >>_auditIndexes` method is now public supported protocol, and has been renamed omitting the leading underscore. The method is now `UnorderedCollection >> auditIndexes`.

### Some milliseconds no longer roll over at 524287999

The methods `Time Class >> millisecondClockValue` and `System Class >> _timeMs` previously rolled back to 0 after 524287999. With the new larger `SmallInteger` range, this is no longer appropriate. The method `System Class >> _timeMsLegacy` has been added with the old behavior.

If you have filed into your application the “goodies” classes `Random` or `FastRandom`, which in earlier releases depended on `_timeMs` returning a value less than 524287999, these will not be automatically upgraded during upgrade/conversion. You must manually file in the updated classes.

### Tranlog sizes larger than raw partition sizes corrected at startup

For all specified tranlogs on raw partitions in `STN_TRAN_LOG_DIRECTORIES`, if the tranlog size that is specified in the corresponding setting in `STN_TRAN_LOG_SIZES` is larger than the physical raw partition size, the size in `STN_TRAN_LOG_SIZES` is reduced as necessary.

### More information provided for error 4059

Error 4059 is returned when a session was forcibly terminated (for example, by `stopSession:`). The message returned now includes a short phrase describing the code path through which the session was terminated.

### Obsolete method deleted

The method `Time Class >> gmtOffsetSeconds`, which had no effect and was marked obsolete, has been deleted.

### Ability to store DoubleByteStrings in ByteArrays

The method `ByteArray Class >> fromString:` will now permit the argument to be an instance of `DoubleByteString`.

## Sessions can now log themselves out

The method `System Class >> logout` has been added, to allow sessions to log themselves out even if they do not have session control privilege.

## Change in listInstances transaction behavior

`listInstances`, `listReferences:`, and similar methods perform an abort prior to executing. In prior releases, if the session was in manual transaction mode and had started a transaction, the abort performed by this method would leave the session outside a transaction. Now, a new transaction is begun under these circumstances.

## Topaz changes

A new command has been added, `STK`. This command is similar to `STACK`, but omits the instance and temporary variables.

`EXPECTVALUE` has additional argument specification allowing OOPs as well as objects to be used.

For more information on these changes, see the updated *GemStone/S 64 Bit Topaz Programming Environment* manual.

## Hidden set changes

The following hidden sets were previously “reserved for GemStone” and are now used:

- 8 - DepMapWriteSet
- 14 - SaveDepMapChangedUnion
- 24 - Enumeration of ReferencedSet

## Float operations now in primitives

The following methods are now implemented as primitives for better performance:

- Float >> ceiling
- Float >> floor
- Float >> rounded
- Float >> roundTo:
- Float >> truncateTo:
- SmallDouble >> ceiling
- SmallDouble >> floor
- SmallDouble >> rounded
- SmallDouble >> roundTo:
- SmallDouble >> truncateTo:

## Number kind indexes have changed

The indexes for `#quietNaN` and `#signalingNaN`, as returned from `_getKind`, have changed; `#quietNaN` is now 5 and `#signalingNaN` is 6.

## Changes in Errors

The following errors have been added in this release

**Table 1 New Errors**

2417	#RT_ERR_NO_MORE_SEGMENTS, #rtErrNoMoreSegments	No more segments can be created. SystemRepository has reached maximum size.
2418	#LOCK_ERR_DEADLOCK, #lockErrDeadlock	RcWrite or Application write lock denied due to possible deadlock. Argument: The object upon which lock was requested.
2419	#LOCK_ERR_TIMEOUT, #lockErrTimeout	RcRetry or Application write lock denied due to timeout. Argument: The object upon which lock was requested.
2420	#LOCK_ERR_INVALID_OBJECT, #lockErrInvalidObject	RcRetry or Application write lock denied; a different object is already registered with the lock queue. Argument: details string
2421	#AUTH_ERR_PROCESS_SWITCH, #authErrProcessSwitch	Processor scheduler cannot switch processes while a primitive is within a bypass-authorization block.
2422	#RT_ERR_OBJ_NOT_IN_EXPORT_SET, #rtErrNotInExportSet	Illegal argument to GciStoreTravDoTravRefs; some objects were not found in the PureExportSet. Argument: details string
4014	#GS_FATAL_ERR_TRANLOG_DIR_FULL	Login denied to other than SystemUser or DataCurator because all tranlog directories or partitions are full. The system is waiting for an operator to make more space available either by cleaning up the existing files or adding a new tranlog directory.
4147	#AUTH_ERR_IN_LOGIN	Fatal read authorization error during login. Argument: detail string

The following errors have had changes:

```
#AUTH_ERR_SEG_READ
#authErrSegRead                2115
```

This error has an additional (third) argument, the details. The arguments are now:

- (1) oop of the object
- (2) its segment
- (3) detail string

```
#AUTH_ERR_SEG_WRITE
#authErrSegWrite                2116
```

The second argument of this error is now the segmentId rather than the oop of the segment.

```
#RT_ERR_OBJ_MUST_BE_COMMITTED,
#rtErrObjMustBeCommitted        2405
```

This error has an additional (second) argument, details. The arguments are now:

- (1) the object
- (2) detail string

```
#AUTH_ERR_SEG_LOGIN_SEG
#authErrSegLoginSeg            4140
```

This error now has an additional (second) argument, the segment id of the bad segment. The arguments are now:

- (1) oop of segment
- (2) segmentId of the segment

## Errors on Windows now include details

When using the shared library files on the Windows platform, the dialog box displayed on an error from GemStone now includes GemStone details, to improve problem diagnosis.

## GCI changes

The following functions have been added to the GCI interface:

```
GciStoreTravDoTravRefs
GciNbStoreTravDoTravRefs
GciClampedTravRefs
GciNbClampedTravRefs
```

These functions are designed for optimized calls made from GBS, to allow future support for single round trip GBS server calls. For details, refer to the updated *GemBuilder for C* manual.

AIX has updated compile flag details; see the updated *GemBuilder for C* manual for details. Both Linux and AIX have updated compiler versions.

## Configuration Parameter Changes

### The following configuration parameters have been removed:

STN\_AIO\_WAIT\_TIME  
STN\_MAX\_SLEEP\_TIME  
STN\_NUM\_LOOPS\_PER\_NET\_POLL  
STN\_OOB\_SOCKET\_POLL\_INTERVAL

### The following configuration parameters have been modified:

STN\_MAX\_AIO\_REQUESTS  
The default has changed from 33 to 128, and the minimum has changed from 33 to 100.

### The following configuration parameters have been added:

#### STN\_COMMITS\_ASYNC

Setting this to TRUE causes stone to acknowledge each commit to the requesting session without waiting for the tranlog writes for that commit to complete.

Default: FALSE

#### WARNING

*When running with this option set to TRUE, if your system crashes during the interval between the return from commit and the actual completion of the commit, work that appeared to have been committed may in fact be lost and not recoverable.*

#### STN\_OBJ\_LOCK\_TIMEOUT

Time in seconds that a session is allowed to wait to obtain one of the special single object write locks. For more information, see the methods System >> waitForRcWriteLock: and System >> waitForApplicationWriteLock:queue:autoRelease:.

Runtime equivalent: #StnObjLockTimeout

Default: 0 (stone waits forever)

Min: 0

Max: 86400

#### STN\_PAGE\_REMOVAL\_THRESHOLD

Minimum batch size for the page manager system gem. When the number of pages waiting to be processed by page manager is greater than this value, then the page manager will request the pages from the stone and process them. Otherwise the page manager will wait until this threshold is exceeded before requesting pages from the stone. The stone cache statistic PagesNeedRemovingThreshold reflects the current value of this parameter.

Runtime equivalent: #StnPageRemovalThreshold

Default: 40

Min: 0

Max: 1792

## Changes in Cache Statistics

### The following statistic has been renamed:

**SmcQueueSize** is now **LastSmcQueueSize** (Stone)

This statistic is measured slightly differently, and has therefore been renamed. The **LastSmcQueueSize** is the **SmcQueueSize** at the time the stone main thread last transferred **smcQueue** contents to the **runQueue**.

### The following statistics have been added:

**CommitRecordsDisposable** (Stone)

The number of commit records queued for disposal. These are no longer being referenced by any session.

**LdiThreadOperations** (Stone)

The number of wakeups from semaphore wait in stone **NetLdiCall** Thread.

**Lock1WaitQueueSize** (Stone)

The number of sessions waiting for the Application 1 object lock. (System `waitForApplicationWriteLock:queue:autoRelease:`)

**Lock2WaitQueueSize** (Stone)

The number of sessions waiting for the Application 2 object lock. (System `waitForApplicationWriteLock:queue:autoRelease:`)"

**NetWriteThreadSocketWrites** (Stone)

The number of socket write operations attempted in stone **NetWrite** thread.

**NetWriteThreadWakeups** (Stone)

The number of wakeups from semaphore wait in stone **NetWrite** thread.

**NumCacheWarmers** (Stone)

The number of cache warmer gem processes currently operating on the shared cache.

**NumInLdiQueue** (Stone)

The number of uncompleted requests in the stone **NetLdiCall** Thread input list.

**NumInMainInpQueue** (Stone)

The number of requests from other threads in the stone **Main** thread input list.

**NumInNetReadWorkList** (Stone)

The number of tasks in the **NetRead** thread work list, produced by **NetPoll** action functions and not yet processed by the **NetRead** thread.

**NumInNetWriteQueue** (Stone)

The number of uncompleted requests in the input list to the stone **NetWrite** thread.

**PgsvrCheckpointState** (Stone)

The state of checkpoint processing within an AIO **pgsvr**; 0=not active, 1=writing dirty pages, 2=finished write dirty, 3=in fsync.

**RcRetryQueueSize** (Stone)

The number of sessions waiting for the **RcRetry** object lock. (System `waitForRcWriteLock:`)

**SlotsCrashedCount** (Stone)

The total number of slots for which the shared cache monitor has attempted recovery because a client process shutdown abnormally.

**StnAioCompletionFailures** (Stone)

The number of aio\_write() operations for which either aio\_error() or aio\_return() reported that the AIO failed.

**StnAioFsyncFailures** (Stone)

The number of fsync() operations in AioWait thread which failed.

**StnAioSuspendEAGAIN** (Stone)

The number of times that aio\_suspend() in AioWait thread returned EAGAIN error. Non zero value indicates that some aio\_write() are failing to complete within 15 seconds.

**StnAioSuspendPrematureReturn** (Stone)

The number of times that aio\_suspend() returned prematurely with EINPROGRESS (has been seen on AIX).

**StnAioSyncWritesAfterCancel** (Stone)

The number of AIOs initiated by aio\_write() that failed to complete within 60 seconds, and were cancelled and written synchronously by AioWait thread. Should normally be zero.

**StnAioWaitsForWork** (Stone)

The number of times stone AioWait thread waited on semaphore for more work.

**StnAioWriteEAGAIN** (Stone)

The number of times aio\_write() returned EAGAIN error due to lack of resources. Stat should normally be zero.

**StnAioWriteFailures** (Stone)

The number of aio\_write() calls by stone main thread which failed with other than EAGAIN. Should be normally be zero.

**StnMainWaitsForFreeAio** (Stone)

The number of times the stone Main thread waited for free AIO buffers to become available.

**TimerThreadWakeup**

The number of wakeups from sleep in stone Timer thread.

**TranlogsFull**

A flag indicating if all configured tranlog partitions or directories are full; 1 means full.

**WaitsForOtherReader**

The number of PageRead operations avoided by waiting for read already in progress by another process.

**The following cache statistics have been removed:**

- AioPollCount
- AioPollTime
- AioWaitCount
- AioWaitTime
- AioWaitTimedoutCount
- AsyncWritesCount

AsyncWritesInProgress  
FailedAioCount  
RecoverCrBacklog  
SocketsPolledCount  
SocketsPolledOobCount  
SocketsReadyCount  
SocketsReadyOobCount  
StnLogState  
StnLoopAioWaitCount  
StnLoopAioWaitTime  
StnLoopAioWaitTimedoutCount  
StnLoopFifoWakeupBytes  
StnLoopFifoWakeupCount  
StnLoopMaxSleepTime  
StnLoopNetPollCount  
StnLoopNetPollOobCount  
StnLoopPollCount  
StnLoopPollInterruptCount  
StnLoopPollNoEventCount  
StnLoopPollNoSleepCount  
StnLoopPollTimeoutCount  
StnLoopsPerNetPoll  
StnLoopTimeInNetPollOob  
StnLoopUpTime  
StnOobSocketPollInterval  
TimeInNetPoll  
TotalSessionsStoppedByStone

## Bugs Fixed

The following bugs in GemStone/S 64 Bit 2.1.4 have been fixed in GemStone/S 64 Bit 2.2.

### Disposing commit records caused slow stone response

Large commit record backlogs were placing an excessive burden on the stone, impacting performance. The internal design has been optimized to avoid this. (#35823)

### User action calls to assert() could have hung

Calling assert() within a useraction results in a send of abort(). This is the equivalent of sending kill -ABRT to the process. Handling this similarly to a kill -TERM was not possible due to the way the OS handles this signal.

The session will now handle SIGABRT by printing the C stack to stdout and, if GEMSTONE\_CHILD\_DEBUG environment variable is defined, waiting for a debugger to be attached. The process exit does entail the risk of a stuck spin lock.

**Production user action code should not call assert().** (#36493)

## Duplicate FreePages cache statistics

The FreePages cache statistic for Gems has been renamed to GemFreePages. The cache statistic FreePages now only applies to the Stone process (#35657)

## Out-of-memory error during indexing even with autocommit=TRUE

The IndexManager's autocommit setting is designed to avoid out of memory problems when indexing a large collection. Some internal structures were not added to the root collection that would force them to be committed and released from memory; which resulted in the out of memory error. (#36147)

## Statmonitor could get stuck in loop creating files

Under particular circumstances - large cache, using statmonitor with the -r option, and shutting down the Stone while statmonitor is running - the statmonitor process could create small sample files in an infinite loop. (#35813)

## Methods remained where abort could lose data

Most methods that perform an abort now will report an error and not execute if that abort would cause unsaved changes to be lost. However, the methods

```
Object >> findReferences  
Object >> findReferencesWithLimit:
```

did not. These methods will now return an error if invoked when unsaved changes would be lost. (#36630)

## Time>>asMillisecondsGmt broken

This method used the wrong rounding constant, returning incorrect results. (#36155)

## asFloat incorrect results for some large Integers

For some large Integers, that are outside the range that can be exactly represented as Float values, on conversion the rounding was incorrectly rounded down rather than up, as required per IEEE 754. (#36251)

## Very small Floats converted to Fractions produced incorrect results

Very small Floats (Floats that are very close to zero), when converted to Fractions either produced zero, or a Fraction that was not exactly equivalent to the original Float. (#36173)

## copyFrom:to: returned error

The implementation of #copyFrom:to: in CharacterCollection and SequenceableCollection returned an error if the stop value was larger than start value. The ANSI specification, which is now followed, requires that this case return a result of size zero. (#35799)

## String>>replaceFrom:to:with:startingAt: incorrect results when replace was performed using self

If the third argument to String>>replaceFrom:to:with:startingAt: was the same object as the receiver, this method produced incorrect results. (#36177)

## **Object>>\_primitiveSize reported physical rather than logical size for NSCs**

The method `Object >> _primitiveSize` returned the internal implementation root size for `NonSequenceableCollections` (NSCs), although this information was not usable with any structural access private primitives. This has been made consistent and now reports the logical size. (#36469)

## **Stone log entries did not appear immediately**

Entries in the stone log appeared when the log was flushed, which may be some time after the event occurred that should appear in the log. The stone log will now flush at least every 15 seconds; this timing limits the delay without impacting performance. (#36527)

## **AIO errors in opening or writing to stone caused stone shutdown**

If AIO writes did not complete promptly, the stone shut down. While the root problem would be the AIO failures at the OS level, GemStone will now attempt to cancel the asynchronous write and write synchronously, start a new log, and report details to the stone log. It is still possible that the stone will need to shut down if the AIO system is unresponsive and the asynchronous writes cannot be cancelled. (#35940)

## **Short stone freeze at checkpoint start**

At the beginning of a checkpoint, the stone may have paused up to 300ms while waiting for a response from an AIO Page Server. The checkpoint code has been modified to avoid socket communications at checkpoints. (#36063)

A new page server stat has been added, `PgsvrCheckpointState`; see page 1-15.

## **makeusers script had several problems**

The script `$GEMSTONE/install/makeusers` generates topaz code that allows you to create multiple users programmatically. This script had errors both in running the script, default values, and in execution of the generated topaz code. (#35772)

## **GsSocket>>connectTo:on: may have failed to connect**

This method may have failed under circumstances when it should have connected. (#36410)

## **Topaz crash on interrupt**

On some previous server versions of GemStone/S 64 Bit, on HP-UX only, it was possible for topaz to SIGSEGV on attempt to interrupt execution using control-C and enter. (#36354)

Note that on HP-UX and AIX, when topaz is waiting for input, an additional enter is required after the control-c before the interrupt is recognized. This is not required on Solaris or Linux.

## **Soft break did not interrupt Delays**

On certain platform and version combinations, a soft break did not interrupt a Delay (#36106)

**Object >> pageCreationTime returned nil**

This method returned nil in previous 2.x versions.

**Could not shut down Stone when maximum sessions logged in**

When a STN\_MAX\_SESSIONS number of sessions was logged in, it was not possible to shut down the stone using stopstone, (#36604)

**\_shrinkExtents was ineffective**

Attempting to reduce the file sizes of extents using \_shrinkExtents usually had no effect, due to the PageManager retaining references to high pages. (#36546).

The fix for this includes the addition of a new configuration parameter; see “STN\_PAGE\_REMOVAL\_THRESHOLD” on page 1-14.

**yearStartDst errored if no DST**

In time zones that do not have a DST (such as GMT), the method TimeZone >> yearStartDst returned an error. (#36401).

**Method caches not cleared on abort**

After compiling a method and aborting, if no other sessions have committed methods, the method may still have been available. (#35868)

**DoubleByteString >> asNumber returned Integer when should return Float**

asNumber should return an Integer, Fraction or Float, depending on the contents of the receiver, but for DoubleByteStrings it returned an Integer, truncating remaining digits. (#36647)

**OldestCrSessionNotInTrans may show session in transaction**

It was possible for the cache statistic OldestCrSessionNotInTrans to reference the session number of a session that was in transaction. (#36561)

# *Extended Character Set Support*

Past releases of GemStone have provided limited ability to process and collate String data that includes extended Characters. Earlier releases did not handle Characters over 127, more recently this limit was raised to 255. This is still insufficient for many languages.

With GemStone/S 64 Bit 2.2, support for extended character set is added. This allows the various Character test, comparison, and conversion methods to work correctly across the full range of Characters that can be represented in one or two bytes. Users now have control over the contents of the character data tables that drive these methods, and can either use tables based on the Unicode Standard, or can design their own based on their particular application requirements.

Customers who do not require more than the basic 256 Character set do not need to do anything; GemStone by default uses basic 256 Character set tables, which provide the best performance. Extended Character Set support was designed to provide the additional abilities without compromising performance for applications that do not need them.

The new Extended Character Set support includes:

- ▶ **Default Built-in Tables** — by default, GemStone/S 64 Bit 2.2 comes with basic character set tables for the 0-255 Character range, based on the Unicode Standard.
- ▶ **16-bit Unicode Standard** — users can extend the character data tables to support the Unicode Standard up through all Characters that can be represented with two bytes (0-65535).
- ▶ **Flexible upgrade and customization** — users can build their own character data tables, or update their applications when there are new releases of the Unicode Standard.

#### **WARNING**

*The power to modify character tables means it is also possible to corrupt these tables in a way that breaks topaz command line processing. Use caution in using this new feature. If a corrupt character table is installed that renders the system unusable, see. "To fix problems after installing an invalid character data table" on page 2-10*

# Character Set Representation

## Categories

Earlier releases included a Character type, which was represented as a number or a symbol. These types were #alpha, #digit, or #special. These types are deprecated and should no longer be used.

The replacement for this is Character categories. The categories are based on the Unicode standard, which provides categories for all possible Characters. Table 1 lists all the Unicode categories. While some of these Character categories are commonly used, other categories are more rare and not used in English or Latin based languages.

### Titlecase category

In addition to uppercase and lowercase, the Unicode standard provides an additional case related category, Titlecase, category #Lt. This is a special case for composite characters that incorporate multiple individual characters. (Such characters are rare; there are only 12 such characters in the Unicode standard). An example is Unicode Character code 0x01C5 (decimal 453), Dž. Titlecase of these composite character forms consists of an Uppercase first character, followed by lowercase remaining characters. The uppercase would be DŽ, the lowercase would be dž, and the titlecase Dž.

For most letters, Titlecase is the same as Uppercase. Titlecase is an optional features in GemStone's formatted tables.

**Table 1 Character Category Codes and Symbols**

1	#Lu	Letter, Uppercase
2	#Ll	Letter, Lowercase
3	#Lt	Letter, Titlecase
4	#Lm	Letter, Modifier
5	#Lo	Letter, Other
6	#Mn	Mark, Nonspacing
7	#Mc	Mark, Spacing Combining
8	#Me	Mark, Enclosing
9	#Nd	Number, Decimal Digit
10	#Nl	Number, Letter
11	#No	Number, Other
12	#Pc	Punctuation, Connector
13	#Pd	Punctuation, Dash
14	#Ps	Punctuation, Open/Start
15	#Pe	Punctuation, Close/End
16	#Pi	Punctuation, Initial Quote
17	#Pf	Punctuation, Final Quote
18	#Po	Punctuation, Other
19	#Sm	Symbol, Math
20	#Sc	Symbol, Currency
21	#Sk	Symbol, Modifier
22	#So	Symbol, Other
23	#Zs	Separator, Space
24	#Zl	Separator, Line
25	#Zp	Separator, Paragraph
26	#Cc	Other, Control
27	#Cf	Other, Format
28	#Cs	Other, Surrogate
29	#Co	Other, Private Use
30	#Cn	Other, Not Assigned

## Configuring your application to use Extended Character Sets.

If your application uses more than the standard 256 Characters, you will need to define Character Data Tables that include the additional Characters, in addition to the basic 256. We recommend that you only add Characters that you require. The full Unicode allows for 65K Characters, but Character lookups in larger tables will take longer than lookups in smaller tables.

GemStone represents Character data in two ways. For efficient loading during login, the tables are stored as a Primitive Character Table, an Array of ByteArrays, each containing numerically encoded information. For users who need to view or edit the Character data, the same information can be represented as a structured table, the Structured Character Table, consisting of an Array of Arrays with Character instances and Symbols.

Some methods work with structured table information, while others expect the lower-level ByteArray representation.

## #CharacterDataTables

GemStone/S 64 Bit 2.2 adds a new variable in the Globals SymbolDictionary, #CharacterDataTables. This variable is used to hold a user-defined Primitive Character Table. This variable is checked during login for each session; if not set, the session uses the default built-in 256 character tables. If the variable is set, the ByteArray tables stored in it are loaded into the internal tables and used for all subsequent character operations.

### Primitive Character Tables

The Primitive Character Tables consist of three tables:

- a. the Dispatch Table, mapping unicode value to index in the main table.
- b. the Main Table containing the numeric data values for uppercase and lowercase. This table is in collation sequence order.
- c. the TitleCase Table containing information for Characters with titlecase forms, if any.

### Structured Character Table

To assist the user in viewing, updating, and modifying these tables, a more structured format is used for representing character data. A Structured character table is formatted as an Array of elements, arranged according to character collate order. Each element is an Array of 4 or 5 entries:

- a. The character for this entry.
- b. The symbolic character category code.
- c. Uppercase character ( if a letter ) / Numerator ( if numeric ).
- d. Lowercase character ( if a letter ) / Denominator ( if fraction ).
- e. (Optional) Titlecase character

For example, in the default Structured Character Table the 73rd element is the letter \$a (unicode/ascii 97). The array at this index contains the following:

```
#( $a #L1 $A $a )
```

The equivalent line in the printed output of the Primitive Character Table would be:

```
65/65: 1 65 97
```

Corresponding to:

```
(table index)/(unicode) (category Code) (uppercase) (lowercase)
```

## Loading Extended Character Sets

GemStone provides the 65K Unicode Character Data Set, which you can load into your repository to allow you to immediately work with all Characters.

To load this into your image, as SystemUser, execute the following and commit:

```
Character activateCharTablesFromFile:
    '$GEMSTONE/goodies/CharTableUnicode.dat'
```

For a customized version of the Character Set, you can load the Primitive Character Table to a Structured Character Table, and modify the Structured Character Table by adding the new Characters you will be using. You can then export this to a file in passivated format, and load it into the Primitive Character Table.

To do this:

1. Create an instance of Structured Character Table, and place this in a temporary variable. To begin with, you may wish to load the default 256 basic Character Table.

For example:

```
| passivatedData |
passivatedData := PassiveObject fromServerTextFile:
    '$GEMSTONE/goodies/CharTableDefault.dat'.
UserGlobals at: #MyCharacterDataTables put:
    (passivatedData activate)
```

2. Add the required entries to this table. Entries are Arrays formatted according to the description on page 2-4. They must be positioned in the Table (Array of Arrays) in the desired sort order, not added to the end (unless the new Character should be sorted last in any collation sequence).

For example, if you would like to include the characters Š (Unicode 352) and š (Unicode 353), which would follow S and s in collation sequence respectively, you might do the following

```
MyCharacterDataTables
  add: (Array
    with: (Character withValue: 352)
    with: #Lu
    with: (Character withValue: 352)
    with: (Character withValue: 353))
  after: #( $S #Lu $S $s ).
```

```
MyCharacterDataTables
  add: (Array
    with: (Character withValue: 353)
    with: #Ll
    with: (Character withValue: 352)
    with: (Character withValue: 353))
  after: #( $s #Ll $S $s ).
```

3. When you are sure your changes are correct, as SystemUser, execute the following and commit:

```
Character installCharTables: MyCharacterDataTables
```

This converts the Structured Character Table into equivalent byte arrays and places them into Globals variable #CharacterDataTables, to be loaded by all later sessions on login.

Use caution: Note that if the table is incorrectly formatted, it is possible you may render your system unusable; see “To fix problems after installing an invalid character data table” on page 2-10. You may also affect or break indexes that rely on String sequencing. We recommend you drop all indexes in your System, if any, prior to modifying the Character Tables.

4. If you wish to make further additions based on the currently customized Primitive Character Table, you may create the Structured Character Table based on your current image information by executing:

```
Character charTables
```

This reads the Primitive Character Table in #CharacterDataTables and returns the equivalent Structured CharacterTable.

## Image changes - Character

Many existing Character instance methods have been modified to use the new features of the Extended Character Set Support; for example:

```
isDigit
isLetter
isLowercase
isUppercase
isSeparator
isVowel
```

The following methods have been added:

Character >> `isTitleCase`

Returns true if this character is a Titlecase character, false otherwise.

Character >> `asTitleCase`

Returns the Titlecase version of this character, or the Uppercase if there is no Titlecase.

Character Class >> `categoryId: aSymbol`

Given a character category *aSymbol*, returns the numeric ID.

Character Class >> `categorySymbol: anInteger`

Given a character category ID *anInteger*, returns the category symbol.

Character Class >> `charTables`

Returns the structured representation for the character data tables recorded in Globals variable `CharacterDataTables`.

Character Class >> `activateCharTablesFromFile: aFileName`

Installs the character data table information recorded in GemStone passivate format in the designated file to the Globals variable `CharacterDataTables`. See the method `passivateCharTablesToFile:` for the method that generates this file.

Character Class >> `passivateCharTablesToFile: aFileName`

Writes the contents of the Globals variable `CharacterDataTables` to the designated file using the GemStone passivate mechanism. See the method `activateCharTablesFromFile:` for the method that reads this file back into `CharacterDataTables`.

Character Class >> `installCharTables: tableInfo`

Converts a structured character data table into appropriately formatted byte arrays and then places them into Globals variable `CharacterDataTables` for use in this and subsequent sessions. You must be `SystemUser` to execute this method.

**WARNING:** Installing incorrectly formatted character table data will break Character and String test, comparison, and conversion operations, including command line processing, to a point where the system will be impossible to use. If this occurs, see "To fix problems after installing an invalid character data table" on page 2-10.

Changing the collation order of Characters will break any indexes that are keyed off of Strings/DoubleByteStrings. Before changing the tables, remove all such indexes, install the new tables, and then reconstruct the indexes.

## Files

There are two new files supplied in the `$GEMSTONE/goodies` directory. These files are in GemStone passivate format.

`CharTableDefault.dat`

Character table data for the default table, in structured form. This file can be used for viewing the contents of the default built-in tables.

`CharTableUnicode.dat`

Character table data for 16-bit Unicode, in low-level ByteArray form. This file is used to install the extended Unicode tables as described below.

## The Unicode Database

The Unicode Consortium is an international standards organization that produces the Unicode Database, which provides unique codes for all Character in all Character Sets. The database continues to develop; GemStone's implementation is using version 4.1.0.

For more information on this database, refer to:

```
http://www.unicode.org/Public/UNIDATA/UCD.html
```

The Unicode Consortium provides code charts by script as well as a single master list of all characters, presented in an ASCII-only, comma delimited version. Version 4.1.0 of this file has been used to create the ByteArray tables, and is provided in passivated object form.

This table can be found at

```
http://www.unicode.org/Public/UNIDATA/UnicodeData.txt
```

If the standard file changes, and you need to recreate the internal tables, GemStone provides the following utility:

```
$GEMSTONE/goodies/UnicodeData.gs
```

The file UnicodeData.gs contains GemStone Smalltalk code that can be filed in as SystemUser to create the UnicodeData class, which includes various utilities for working with the Unicode Character Database.

Methods of interest in the UnicodeData class include:

```
UnicodeData Class >> loadFromFile: fileName
```

Load the contents of the Unicode Database from the specified file.

```
UnicodeData >> generateTables
```

Generate structured character data tables from the Unicode data. The result from this method can then be installed using Character>>installCharTables:.

## Procedures

The following section describes several common procedures for using the new extended character set features of GemStone/S 64 Bit 2.2 .

### View current character data tables

To view the contents of the internal character data tables currently in use:

1. Login a linked topaz session.
2. Execute:

```
Character _dumpCharTables
```

This writes output to stdout (standard output), that is, the linked topaz session output.

### View structured version of current

To get the structured version of the character data tables:

When using the default built-in 256 character data tables:

```
(PassiveObject fromServerTextFile:
    '$GEMSTONE/goodies/CharTableDefault.dat') activate
```

When using Globals variable CharacterDataTables to override the default:

```
Character charTables
```

### To install the 16-bit Unicode Standard character data table

1. As SystemUser, execute:

```
Character activateCharTablesFromFile:
    '$GEMSTONE/goodies/CharTableUnicode.dat'
```

2. Commit

### To install a specially tailored character data table

1. Construct your table as required, using formatting information provided earlier. Store someplace accessible, for example:

```
UserGlobals at: #MyCharacterDataTables
```

2. As SystemUser, execute:

```
Character installCharTables: MyCharacterDataTables
```

3. Commit

### To install updated standard character data table

To install an updated 16-bit Unicode Standard character data table: (for when new releases of the Unicode Standard come out)

1. Download the updated Unicode Database at:

```
http://www.unicode.org/Public/UNIDATA/UnicodeData.txt
```

to a file (note that this URL may change). We'll call this file "unicode.dat"

2. Login a topaz session as SystemUser
3. File-in and commit \$GEMSTONE/goodies/UnicodeData.gs
4. Execute the following:

```
| tables |
UnicodeData loadFromFile: 'unicode.dat'.
tables := UnicodeData generateTables.
Character installCharTables: tables.
```

5. Commit

### To save the current CharacterDataTable to a file

To save the current contents of the Globals variable CharacterDataTables to a passivated file, execute:

```
Character passivateCharTablesToFile: 'MyFileName'
```

### To load a file into the current CharacterDataTables

To install the saved contents of a passivate file to the Globals variable CharacterDataTables:

1. As SystemUser, execute:  

```
Character activateCharTablesFromFile: 'MyFileName'
```
2. Commit

### To reset CharacterDataTables to use the default 256 character set

1. Login as SystemUser
2. Execute:  

```
Globals removeKey: #CharacterDataTables.
```
3. Commit

### To fix problems after installing an invalid character data table

1. At OS level, set the host environment variable GS\_DISABLE\_CHARACTER\_TABLE\_LOAD to TRUE (the particular value of this environment variable does not matter; its existence is the critical factor)
2. Login a new topaz session as SystemUser.
3. Execute Globals at: #CharacterDataTables put: nil.
4. Commit