GemStone®

GemStone/S 64 Bit™ Release Notes

Version 3.0

June 2011

vmware®

GEMSTONE S™ 64

INTELLECTUAL PROPERTY OWNERSHIP

This documentation is furnished for informational use only and is subject to change without notice. VMware, 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 VMware, 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 VMware, 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 VMware, Inc.

This software is provided by VMware, 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 VMware, 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

PATENTS

GemStone software is covered by U.S. Patent Number 6,256,637 "Transactional virtual machine architecture", Patent Number 6,360,219 "Object queues with concurrent updating", Patent Number 6,567,905 "Generational garbage collector with persistent object cache", and Patent Number 6,681,226 "Selective pessimistic locking for a concurrently updateable database". GemStone software may also be covered by one or more pending United States patent applications.

TRADEMARKS

VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions.

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

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

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

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

Microsoft, **MS**, **Windows, Windows XP, Windows 2003**, **Windows 7** and **Windows Vista** 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.

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

AIX, **POWER5**, and **POWER6** are trademarks or registered trademarks of International Business Machines Corporation.

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

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. VMware 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.

VMware, Inc.
15220 NW Greenbrier Parkway
Suite 150
Beaverton, OR 97006

Preface

About This Documentation

These release notes describe changes in the GemStone/S 64 Bit™ version 3.0 release. Read these release notes carefully before you begin installation, conversion testing, or development with this release.June 2011

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 3.0.

Terminology Conventions

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

Technical Support

GemStone Website

http://support.gemstone.com

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

- **Documentation** for released versions of all GemStone products, in PDF form.
- **Downloads** and **Patches**, including past and current versions of GemBuilder for Smalltalk.
- **Bugnotes**, identifying performance issues or error conditions that you may encounter when using a GemStone product.
- **TechTips**, providing information and instructions that are not in the documentation.

- **Compatibility matrices**, listing supported platforms for GemStone product versions.

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

Help Requests

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

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

Website: http://techsupport.gemstone.com

Email: techsupport@gemstone.com

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

When submitting a request, please include the following information:

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

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

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, for issues impacting a production system. For more details, contact your GemStone account manager.

Training and Consulting

Consulting is available to help you succeed with GemStone products. Training for GemStone software is available at your location, and training courses are offered periodically at our offices in Beaverton, Oregon. Contact your GemStone account representative for more details or to obtain consulting services.

Contents

Chapter 1. GemStone/S 64 Bit 3.0 Release Notes

Chapter 3. Changes in Topaz

Chapter 4. Bugs Fixed

Chapter 1

GemStone/S 64 Bit 3.0 Release Notes

1. Overview

GemStone/S 64 Bit version 3.0 is a new version of the GemStone/S 64 Bit object server. This release is a substantial redesign and optimization of the GemStone/S 64 Bit product, with many changes and new features.

Due to these extensive changes, customer application code require changes. You may also find new features that you can take advantage of by updating your code. Please read these Release Notes carefully to ensure that the appropriate changes are made.

These Release Notes describe changes between the previous version of GemStone/S 64 Bit, version 2.4.4.6, and version 3.0. They do not include changes that were introduced in version 2.4.4.6 or before. If you are upgrading from a release prior to 2.4.4.6, you must also review the Release Notes for each intermediate release to see the full set of changes.

New keyfiles are required with Version 3.0. Contact GemStone Technical Support or email keyfiles@gemstone.com for an evaluation keyfile.

For details about installing GemStone/S 64 Bit 3.0 or upgrading from earlier versions of GemStone/S 64 Bit or other GemStone server products, see the platform-specific version 3.0 *GemStone/S 64 Bit Installation Guide*.

Ruby and MagLev

GemStone/S 64 Bit version 3.0 has extensive internal redesign, both for performance and to support the Ruby programming language. GemStone's implementation of Ruby with persistence is MagLev.

The changes to the virtual machine, class hierarchy, and bytecodes to support Ruby are extensive, but for the most part are transparent to the user. Smalltalk and Ruby can coexist in an image, but while the complete Ruby environment is included in the distribution, the Smalltalk extents do not include the bulk of the Ruby classes. For more information on MagLev, go to **`http://maglev.gemstone.com/`**.

In the image, you will see additional Ruby classes, methods, and variables. These are not used in regular Smalltalk execution, and are not enumerated in these release notes. The $GEMSTONE/bin directory includes a ruby extent.

Seaside and GLASS

GLASS, GemStone's Seaside framework, continues to undergo development and improvement. New functionality has been added to version 3.0 that is primarily designed to support Seaside or for the specific needs of Seaside applications.

While the Seaside installation provided in the distribution is fully functional, upgrade of Seaside applications from earlier GemStone/S 64 Bit versions to 3.0 is not available with the version 3.0 release.

Further information about Seaside and Glass, see the website:

`http://seaside.gemstone.com/`

and blog:

`http://gemstonesoup.wordpress.com/`

Upgrade/conversion process

New keyfiles are required with GemStone/S 64 Bit version 3.0. Keyfiles for GemStone/S 64 Bit 2.x will not work with version 3.0. If you need a keyfile for version 3.0, contact GemStone Technical Support, or email keyfiles@gemstone.com.

GemStone/S 64 Bit version 3.0 uses a new bytecode implementation, as well as structural and base class changes from 2.4.x. As a result, in addition to conversion, all application code must be recompiled. Any stored executable blocks in your repository must also be recreated. This includes the sort blocks stored in instances of SortedCollection. The conversion process includes tools to support the automatic conversion of the sort blocks in persistent SortedCollections.

There also may be specific code changes required as part of conversion, many of which can be automated. Other changes may also require modifications to your application.

- Any uses of #[ ] Array constructor syntax must be modified. GemStone provides tools to allow this during application code file-in, and by calls to process external Topaz scripts. See "Array and ByteArray literals – change in semantics for #[ ]" on page 64.
- ScaledDecimal has been entirely redesigned. The old ScaledDecimal is still available as the FixedPoint class. You may modify your code to use FixedPoint to keep the same behavior; otherwise, references in your code to the new ScaledDecimal class and literals containing the s syntax (e.g. 2.3s2) will use the new ScaledDecimal implementation. See "ScaledDecimal reimplemented" on page 47.
- The fileout form of class and method definitions has been changed to use new subclass creation methods, and to include the **`environment`** keyword. Code filed out from v3.0 cannot be filed into earlier versions, although code filed out from older versions in many cases will file in to 3.0. If you are using a code management tool, you should file out all application code and reload into your code management system. See "Subclass creation change" on page 22.
- Exception signaling and handling has been redesigned to use native ANSI exceptions. If you have custom legacy exceptions, you will need to update these to use ANSI

exceptions. The existing legacy handling protocol is, for the most part, still available, although it is deprecated. See "Exception Changes" on page 42. Exception handing behavior may be different under specific conditions, particularly where GemStone did not previously adhere to ANSI specifications.

- If you use the description instance variable of Class, this data is lost during conversion, since the description is no longer defined as an instance variable in Class, and the slot is used for other purposes. See "Class description" on page 22.
- Protocol for removing users, and in some cases for creating users, has changed. See "Removing users" on page 39 and "Creating and adding users" on page 38.
- The tag mechanism is deprecated, and the method names have changed. You will need to change calls to **`tagAt:`** and **`tagAt:put:`** to call **`_tagAt:`** and **`_tagAt:put:`**, or preferably change your code to use dynamic instance variables. For details, see "Tag changes" on page 26 and "Dynamic instance variables" on page 25.

The conversion process is only supported directly from version 2.4.4.x to version 3.0. If you wish to upgrade from a version earlier than 2.4.4 or from 32-bit GemStone/S, you must first upgrade or convert to 2.4.4.x, then convert to 3.0.

GBS, GemConnect, and GBJ

In addition to upgrading the server, if you use GemBuilder for Smalltalk (GBS), you must upgrade to GBS version 7.4. The changes in v3.0 require corresponding changes to GBS.

You must also upgrade GemConnect to version 2.2.3, and GemBuilder for Java to version 3.0.1. Existing releases of these products will not file in correctly to GemStone/S 64 Bit version 3.0, primarily due to the Array constructor syntax change.

2. Supported Platforms and GBS Versions

Platforms

GemStone/S 64 Bit version 3.0 is supported on the following platforms:

- Solaris 10 on SPARC
- Solaris 10 on x86
- HP-UX 11.31 on Itanium
- AIX 5.3, TL7, SP2 and AIX 6.1, TL1, SP1
- SuSE Linux ES 10 SP1 and Red Hat Linux ES 5.0 and 5.5, on x86
- Mac OSX 10.6.4 (Snow Leopard), with Darwin 10.4.0 kernel, on x86

Note that v3.0 will not run on Solaris 9 or earlier.

Native code is not supported on Itanium; processing on Itanium will use interpreted code as in previous versions.

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

GBS version/s

If you use GemBuilder for Smalltalk (GBS), you should upgrade to GBS version 7.4 for use with GemStone/S 64 Bit v3.0. Earlier versions of GBS may log in to GemStone/S 64 Bit v3.0 and perform some operations, but can expect errors.

GBS version 7.4

VW 7.8 32-bit with 32-bit 7.8 OE	**VW 7.8 64-bit** with 64-bit 7.8 OE	**VW 7.7.1 32-bit** with 32-bit 7.7.1 OE	**VW 7.7.1 64-bit** with 64-bit 7.7.1 OE
▸ Windows 7, Windows 7 64-bit, Windows Vista, Windows 2003 Server, and Windows XP ▸ Solaris 10 on SPARC ▸ SuSE Linux ES 10 and RedHat Linux ES 5.5 and 5.0	▸ Solaris 10 on SPARC ▸ SuSE Linux ES 10 and Red Hat Linux ES 5.5 and 5.0	▸ Windows 7, Windows 7 64-bit, Windows Vista, Windows 2003 Server, and Windows XP ▸ Solaris 10 on SPARC ▸ SuSE Linux ES 10 and RedHat Linux ES 5.5 and 5.0	▸ Solaris 10 on SPARC ▸ SuSE Linux ES 10 and Red Hat Linux ES 5.5 and 5.0

For more details on supported platforms with GBS version 7.4, refer to the Installation Guide for that version.

3. Internal Changes and Performance Improvements

GemStone/S 64 Bit version 3.0 has substantial internal redesign for optimization and to support native code and Ruby functionality.

For v3.0, the bytecode instruction set has been rewritten. Methods and executable blocks have been redesigned to use these new bytecodes. While all methods and executable blocks in an upgraded application must be recompiled to use the new bytecodes, these changes should not greatly impact applications after the upgrade process is completed.

In addition, primitives have been renumbered and internal handling has changed. Again, this should have no impact on customer applications. However, if you call any primitives by number (including the protected mode primitive 901), you will need to update your code.

Native code support

GemStone/S 64 Bit v3.0 includes support for native code generation, allowing faster performance. This is enabled via a configuration parameter, GEM_NATIVE_CODE_ENABLED. By default, native code is enabled. For details, see "GEM_NATIVE_CODE_ENABLED" on page 103.

Native code is not supported on Itanium.

GsNMethod

A new class, GsNMethod, replaces GsMethod. Existing instances of GsMethod are not executable.

Executes or GCI sends can be specified to execute entirely in native code or entirely in interpreted code. Native code is generated when the GsNMethod is faulted in for execution (if native code is enabled), regardless whether current execution is native or interpreted. Native code resides in the code_gen subspace of the temporary object cache.

Breakpoints disable native code; after all breakpoints are removed, native code is reenabled.

ExecBlock

A new class, ExecBlock, replaces ExecutableBlock. Each ExecBlock has its own instance of GsNMethod with the code for that block. Stored instances of ExecutableBlock are not usable; any stored blocks must be recompiled and recreated in version 3.0. This includes the sort blocks stored in instances of SortedCollection, which must be recreated.

Classes ExecBlock0...5 and ExecBlockN are also new in v3.0. These classes are instantiated in-memory only for use by the VM.

Environment Id

Execution now takes place in one of up to 256 execution environments specified by an environmentId. The Smalltalk base image is environmentId 0; Ruby uses environmentId 1 and 2. EnvironmentId is used for method lookup; lookup is based on both selector and environmentId.

Smalltalk applications do not need to know anything about environmentId. However, for non-Ruby applications, environmentIds 2 through 255 are available for Smalltalk

applications, and have potential value for controlling application behavior. The special syntax @env and @ruby is available to invoke methods in specific environments.

Version 3.0 introduces new versions of many GCI calls, Topaz functions, and Smalltalk methods that take an environmentId argument. The corresponding call, function, or method without this keyword by default uses the environmentId 0.

Method dictionary changes

The methodDicts instance variable of Behavior may now be a Dictionary or an Array of Dictionaries. The environmentId into which a method is compiled determines the method dictionary into which the new method is installed.

Unless other environmentIds are explicitly used, methods will be compiled in environmentId 0 and stored in a Dictionary in #methodDicts, as in previous releases.

Internal process scheduling improvements

The virtual machine now has eight stack memory areas, so eight instances of GsProcess can be active in stack memory at once. Process switching between these processes is very fast. An LRU algorithm chooses the process to evict when a new stack area is needed. It is possible to lock up to four GsProcess instances in stack memory, using **`GsProcess >> lockInMemory`**.

These changes include a reimplementation of GsProcess. The old GsProcess class has been renamed ObsoleteGsProcess. You must examine any code that has been added to the old GsProcess and, if appropriate, move it to the new class.

`GsProcess >> terminate` behavior has also changed. In version 3.0, terminating the process behaves correctly in relation to the stack of the terminated GsProcess. The old behavior is available as a new method **`GsProcess >> terminate9`**.

Page lookup improvement

In previous versions, all page lookups by Gems required serialized access to a hash table row (via a spin lock). In version 3.0, only writes need to be serialized; operations to read a page that is not currently being written can just read-lock the hash table row. This results in faster lookup of pages in the shared page cache.

Abort requires fewer round trips to Stone

For most aborts, the number of round trips to the Stone has been reduced from 2 to 1.

New communication channel for Gem requests to Stone

In previous versions, when the Gem made a request to the Stone, it was serialized on the Shared Memory Queue (SMC queue) using a spin lock. In version 3.0, the Gem atomically sets a bit in shared memory to request service from the Stone. This avoids any need to block, and can improve Gem-to-Stone communication performance by 30%. The SMC queue no longer exists.

Session Priority

Due to this mechanism, the Stone may receive multiple simultaneous requests, so requests no longer have a strict time sequence. The Stone will process requests from lowest to highest via session id. To allow more important Gem requests to be serviced first, sessions

now include a priority. Higher priority sessions are added to the run queue ahead of lower priority sessions. However, any session that has the commit token, or is about to get the commit token, is automatically at the head of the queue.

This priority is distinct from the ProcessScheduler priority, which controls priorities of threads within a session, not between sessions.

Session priorities may be:

0 - lowest
1 - low
2 - medium; the default
3 - high
4 - highest

The default is 2, except for the Reclaim GcGem and the Admin GcGem, both of which have priority 0.

To set session priority, the following methods have been added:

`System class >> setSessionPriority:` *anInt* **`forSessionId:`** *aSessionId*

`System class >> priorityForSessionId:` *aSessionId*

These operations require the #SessionAccess privilege. In addition, the operation to update a session's priority requires the new #SessionPriority privilege.

The session's priority is now included in the information returned by **`System class >> descriptionOfSession:`**. For more about this, see "Updated descriptionOfSession: result" on page 80.

Segment name change to GsObjectSecurityPolicy

The term "Segment" has always been a misleading and confusing term for GemStone's object security management. To correct this source of confusion, Segment has been renamed to GsObjectSecurityPolicy for version 3.0. It is referred to in addition as "security policy" or just "policy" in the documentation.

The names of the classes and references to Segment in method names have been changed. Privileges have also been renamed, and named Segments (that is, GsObjectSecurityPolicies) have new names. The associations for the old names using "Segment" remain, so that references to Segment and named Segments can be resolved. Similarly, most methods that included the term "Segment" remain for backwards compatibility, but are now deprecated. For example:

Version 2.x method name	Version 3.0 method name
`changeToSegment:`	**`objectSecurityPolicy:`**
`assignToSegment:`	**`assignToObjectSecurityPolicy:`**
`currentSegment`	**`currentObjectSecurityPolicy`**
`currentSegment:`	**`currentObjectSecurityPolicy:`**

Other methods have been renamed similarly.

References to the Segment class, and calls to these methods, should be corrected at some future time in customer applications.

Other than the name change, there are no significant changes in Segment (now GsObjectSecurityPolicy) behavior. However, users created without specifying a default security policy (Segment) are now created with a nil security policy, rather than having a new Segment created as a side effect, which may impact your application. See "Creating and adding users" on page 38 for details.

POSIX AIO system removed

The use of POSIX asynchronous I/O for writing to new transaction logs has been removed. This has been a source of intermittent problems over the years. It has been replaced with our own thread-based I/O subsystem. GemStone/S 64 Bit installation no longer requires you to enable asynchronous I/O.

This is configured via the new configuration parameter STN_NUM_AIO_WRITE_THREADS. For more information, see "STN_NUM_AIO_WRITE_THREADS" on page 105.

A number of cache statistics related to POSIX AIO have been removed. In their place, version 3.0 introduces the new cache statistics StnAioWriteQueueSize, StnAioWritesQueuedCount, StnAioNumWriteThreads, and StnAioWriteQueueHighWaterSize. For details, see "Cache Statistic Changes" on page 96.

Improved stack trace utility for Linux/Mac

The gstack utility, which was previously distributed in $GEMSTONE/bin on the Linux and Mac platforms, has been replaced by a custom pstack script, which provides more information. You should now use pstack instead of gstack to get stack trace information on Linux or Darwin.

Reserved selectors

Version 3.0 expands the list of reserved and optimized selectors. This is the complete list of reserved selectors:

```
-                          _isInteger
*                          _isNumber
~~                         _isScaledDecimal
==                         _isSmallInteger
and:                       _isArray
_and:                      _isExceptionClass
bitAnd:                    _isExecBlock
or:                        _isFloat
_or:                       _isOneByteString
ifFalse:                   _isSymbol
ifFalse:ifTrue:            _isRange
ifTrue:                    _isRegexp
ifTrue:ifFalse:            _isRubyHash
untilFalse                 _isRubyString
untilFalse:                repeat
untilTrue                  timesRepeat:
untilTrue:                 to:do:
whileFalse                 to:by:do:
whileFalse:                _downTo:by:do:
whileTrue                  _downTo:do:
whileTrue:                 _gsReturnNoResult
isKindof:                  _gsReturnNothingEnableEvents
ifNil:                     _stringCharSize
ifNil:ifNotNil:            _rubyKindOf:
ifNotNil:                  _leaveProtectedMode
ifNotNil:ifNil:            __inProtectedMode
                           __threadRubyEnvId
```

4. Class Structural and Management Changes

Class structural changes

The position of Class and Behavior within the class hierarchy has been changed, to allow support for Ruby classes.

```
Behavior
   ObsoleteMetaclass
   Module
      Metaclass3
         Class
```

The new class Metaclass3 (for version 3.0) replaces the v2.x Metaclass. The v2.x Metaclass has been renamed to ObsoleteMetaclass in v3.0. During the conversion process, v2.x instances of Metaclass are automatically converted to instances of Metaclass3.

There are no instances of Module in a Smalltalk repository; instances of Module are only created by Ruby code.

Code that explicitly checks **`== Metaclass`** should send **`isMeta`** instead.

If you have added or modified behavior of Metaclass, you will need to manually move this code to Metaclass3.

In version 2.x, both Metaclass and Class were subclassed from Behavior, so Class could not inherit from Metaclass. In version 3.0, Class is now a subclass of Metaclass3, and as such will inherit its behavior. Code in Metaclass3 may need to take into account invocation on an instance of Class.

Class description

Class no longer includes an instance variable **`#description`**. The methods **`description`** and **`description:`** now access an entry in the Class instance variable **`#extraDict`**. Conversion to version 3.0 will cause data in the Class instance variable **`#description`** to be lost. If you use this instance variable, you will need to store the values elsewhere, such as a separate dictionary indexed by class, to preserve the information through the conversion process.

The Class's description previously held the class comment. This is now handled differently; see "Class documentation changes" on page 25.

Subclass creation change

In previous releases, class creation methods included keywords such as **`instanceInvariant:`** and **`isModifiable:`**. In version 3.0, many of these subclass creation variants are specified via the **`options:`** keyword, which is an Array of symbols.

The options can include these:

#subclassesDisallowed
#disallowGciStore
#modifiable
#traversalByCallback
#logCreation

In addition, options can include only one of the following mutually exclusive options:

#dbTransient
#instancesNonPersistent
#instancesInvariant

#disallowGciStore and #traversalByCallback are designed for internal use by GBS. #dbTransient and #instancesNonPersistent were previously set by sending messages to the class, rather than via class definition. #logCreation is a new option that indicates that class creation should be logged to the gem log (using **`GsFile >> gciLogServer:`**).

Sending Class definition messages that would result in a class identical to the existing class are now do not creates a new class instance (do not create a new version of the class).

With the addition of the options: keyword, the default fileout form has changed. Class definitions filed out from version 3.0 by default use the **`options:`** keyword, and will not file in to earlier versions of GemStone without modification.

New class creation protocol

Version 3.0 introduces the following subclass creation methods:

```
byteSubclass:classVars:classInstVars:poolDictionaries:
   inDictionary:

byteSubclass:classVars:classInstVars:poolDictionaries:
   inDictionary:newVersionOf:description:options:

byteSubclass:classVars:classInstVars:poolDictionaries:
   inDictionary:options:

indexableSubclass:instVarNames:classVars:classInstVars:
   poolDictionaries:inDictionary:

indexableSubclass:instVarNames:classVars:classInstVars:
   poolDictionaries:inDictionary:newVersionOf:description:
   options:

indexableSubclass:instVarNames:classVars:classInstVars:
   poolDictionaries:inDictionary:options:

subclass:instVarNames:classVars:classInstVars:poolDictionaries:
   inDictionary:

subclass:instVarNames:classVars:classInstVars:poolDictionaries:
   inDictionary:newVersionOf:description:options:

subclass:instVarNames:classVars:classInstVars:poolDictionaries:
   inDictionary:options:

subclass:instVarNames:inDictionary:options:
```

Deprecated or obsolete class creation protocol

To avoid complications for customers filing in code from older versions of GemStone, or for programmatic class creation, existing public class creation protocol has been retained, including protocol that was made obsolete in earlier versions or products. We encourage review and update of the subclass creation methods used.

Any class creation methods that include the following keywords are now obsolete:

Keyword	Status
`inClassHistory:`	Long obsolete; has been officially deprecated in v3.0. Use a method containing the **`newVersionOf:`** keyword instead.
`isInvariant:`	Long obsolete. Include #instanceInvariant in the **`options:`** argument.
`constraints:`	Not used in GemStone/S 64 Bit, which does not use constraints, and made obsolete in v2.1. Use a method without the **`constraints:`** keyword.
`isModifiable:`	Made obsolete in GemStone/S 64 Bit v3.0. Include #modifiable in the **`options:`** argument.
`instanceInvariant:`	Made obsolete in GemStone/S 64 Bit v3.0. Include #instanceInvariant in the **`options:`** argument.

Class documentation changes

Earlier GemStone products and versions have provided class documentation in the form of an instance of GsClassDocumentation, stored in the #description instance variable of a class. This information was provided in the GBS browser, and by sending **`#description`** to a Class; but was in general inconvenient to access.

In version 3.0, each class has a class method **`comment`** in category #Documentation. The method answers a String containing the same information as provided by the classes' GsClassDocumentation. GBS browsers now return the results of executing the **`comment`** method, and no longer use the GsClassDocumentation.

The GsClassDocumentation is still available programmatically for many classes via the #description method, but is not created for new classes.

Class name length limit raised

Previously, Class names were limited in length to 64 Characters. The new limit is 1024 Characters.

Session method dictionaries interface changer

The mechanisms that allow session-specific method dictionaries to be added to classes has changed in this release.

Dynamic instance variables

Previous releases provided up to two slots that could be added to any object, accessed via **`tagAt:`** and **`tagAt:put:`**.

In version 3.0, tags are replaced by dynamic instance variables. Dynamic instance variables are implemented per object, as a sequence of key-value tuples in the object's tag area.

Dynamic instance variables are also handled differently by **`become:`** and **`copy`** than tags were in previous versions. In previous releases, **`become:`** did not affect tags, so the tag contents remained with the identity (OOP); and **`copy`** did not include tag contents in the copied object. In version 3.0, the **`become:`** and **`copy`** operations treat dynamic instance variables as if they were regular instance variables.

The maximum number of dynamic instance variables that can be added to an object is 255. However, the maximum may be lower for classes with many instance variables, since an object cannot be changed to a large object by adding dynamic instance variables. The actual limit for the number of dynamic instance variables is:

```
(255 min: ((2034 - self class instSize) / 2)
```

Access to dynamic instance variables is provided via the following new methods:

```
Object >> dynamicInstVarAt:
Object >> dynamicInstVarAt:put:
Object >> removeDynamicInstVar:
Object >> dynamicInstanceVariables
```

And similar new GCI calls:

```
GciFetchDynamicIv
GciStoreDynamicIv
GciFetchDynamicIvs
```

Tag changes

The tag mechanism remains but is deprecated, and method names and semantics have changed. The methods **`tagAt:`** and **`tagAt:put:`** have been replaced by **`_tagAt:`** and **`_tagAt:put:`**. The tag fields now move with the object during a **`become:`** operation, and are copied by **`copy`** methods, as are dynamic instance variables.

Tags and dynamic instance variables should not be used on the same object, as they both access the same storage, which may result in unexpected behavior.

Obsolete classes

A number of classes have been reimplemented in version 3.0. To avoid problems with older code, the previous versions of these classes have been renamed to include "Obs" or "Obsolete". In most cases, instances of the 2.x class will automatically convert to use the new class. If you have modified any of the reimplemented classes, you should review your code and move updates to the new class.

A new dictionary, ObsoleteClasses, has been added to the Globals of DataCurator. Some obsolete classes, particularly classes that were obsolete in previous releases, have been moved to this dictionary.

Version 2.x class	Reimplementation/change in v3.0
Block	Moved to the ObsoleteClasses dictionary.
ClampSpecification	Converted to **Obsolete23ClampSpecification**, and moved to the ObsoleteClasses dictionary.
DoubleByteString, DoubleByteSymbol	Converted to **ObsDoubleByteString** and **ObsDoubleByte-Symbol**, respectively. See "Multi-byte string instances now stored in-memory in CPU byte order" on page 65.
Exception	Converted to **ObsoleteException**. See "Exception Changes" on page 42.
Float	Converted to **ObsFloat**. See "Float class reimplemented" on page 49.
GsFile	Converted to **Obsolete23GsFile**, and moved to the Obsolete-Classes dictionary. See "GsSocket and GsFile Changes" on page 54.
GsProcess	Converted to **ObsoleteGsProcess**. See "Internal process sched-uling improvements" on page 18.
GsSocket	Converted to **ObsoleteGsSocket**. See "GsSocket and GsFile Changes" on page 54**.**
LargeNegativeInteger, LargePositiveInteger	Converted to **ObsLargeNegativeInteger** and **ObsLargePosi-tiveInteger**, respectively. See "LargeInteger replaces LargePosi-tiveInteger / LargeNegativeInteger" on page 47

Version 2.x class	Reimplementation/change in v3.0
Metaclass	Converted to **ObsoleteMetaclass**. See "Class structural changes" on page 22.
MethodContext	Moved to the ObsoleteClasses dictionary.
QuadByteString	Converted to **ObsQuadByteString**. See "Multi-byte string instances now stored in-memory in CPU byte order" on page 65.
ReenterBlock	Moved to the ObsoleteClasses dictionary.
SelectionBlock	Moved to the ObsoleteClasses dictionary.
SmallFloat	Converted to **ObsSmallFloat**. See "Float class reimplemented" on page 49.
VariableContext	Converted to **ObsoleteVariableContext**, and moved to the ObsoleteClasses dictionary.

The following classes, which were prefixed Obsolete in version 2.x, have been moved to the new ObsoleteClasses dictionary. Their OOPs are unchanged.

ObsoleteClampSpecification
ObsoleteDateTime
ObsoleteDateTime50
ObsoleteDictionary
ObsoleteGsFile
ObsoleteIdentityCollisionBucket
ObsoleteIdentityDictionary
ObsoleteLanguageDictionary
ObsoleteRcCollisionBucket
ObsoleteSymbol
ObsoleteSymbolAssociation
ObsoleteSymbolDictionary
ObsoleteSymbolKeyValueDictionary
ObsoleteSymbolListDictionary
ObsoleteSymbolSet
ObsoleteTimeZone

5. OS Process Level Changes

Change in process identification and cache naming

In previous releases, GemStone hosts were identified using IP address, which had some issues in cases where the IP address changed, as well as being problematic for multi-homed machines.

In version 3.0, each host running any GemStone process will be assigned a unique 64-bit hostId, obtained by reading eight bytes from **`/dev/random`**. This hostId is created the first time a GemStone process is started on this host, and is stored in a file named **`gemstone.hostid`** in the locks directory. This file will be created with permission 444 if it does not exist. The locks directory is normally **`/opt/gemstone/locks/`**, but may be modified by a setting for GEMSTONE_GLOBAL_DIR.

If this file is deleted, the shared page cache monitor will regenerate it.

In previous releases, shared caches were identified by a name of the form "stoneName@hostName". In version 3.0, shared caches have names of the form "stoneName~hostId", with the Stone name now followed by the tilde character and the hostId in hex.

While the hostId is used in some NRS strings, this change should have minimal impact. Remote login parameters are unchanged.

While the (new) cache name may be used with statmonitor, it is no longer necessary to specify the cache by name when monitoring a remote cache. On a remote host, starting statmonitor, using the name of the Stone with which the remote cache is associated, will monitor the remote shared cache for that Stone.

The GemStone C Statistics Interface, GCSI, has been updated appropriately for this change.

A new method has been added to get the host id programmatically:

`System class >> hostId`
Returns an Integer which represents a unique 64-bit identifier for the host where the session is running.

Specifying well-known ports

In version 3.0, it is now possible to specify the port numbers that the Stone, shared page cache monitor, and netldi will listen on as their well-known port.

To specify the port for the Stone and shared page cache monitor, set the STN_WELL_KNOWN_PORT_NUMBER and SHR_WELL_KNOWN_PORT_NUMBER, respectively, to a port number. The specified port must not be in use by another process. A port number of 0 means the system will select an unused port number.

For details, see "STN_WELL_KNOWN_PORT_NUMBER" on page 106 and "SHR_WELL_KNOWN_PORT_NUMBER" on page 104.

To specify the well-known port for the netldi, use the new -P option on startnetldi. For more information, see "startnetldi added -P option" on page 91.

Handling OS low memory conditions on Linux

The Linux default behavior on low operating system memory is to kill processes that are using large amounts of memory. Since Linux considers that the shrpcmon owns the shared memory cache and therefore owns more memory than most other processes, Linux will first kill the shrpcmon, if low in virtual memory.

To avoid shutting down the system entirely when low in memory, if the Linux user has sufficient privilege, we now make the stone, shrpcmon, and the system gems (specifically the pagemanager) less sensitive to the OOM killer. If the user does not have privileges, we make non-system processes (gem and topaz -l) more sensitive to OOM killer, in the hope that they will be killed before the shared cache monitor if the machine runs out of virtual memory.

Messages are written to stdout about what is being done, and will show up in the linked topaz session (except in quiet mode), or in the gem log.

The Linux user privilege required is CAP_SYS_RESOURCE; we recommend granting this privilege to the user who owns the stone and shrpcmon processes. Changing process sensitivity to the OOM killer is accomplished by writing to **`/proc/`**<*pid*>**`/oom_score_adj`** if running on Linux kernel 2.6.36 or newer, or by writing to **`/proc/`**<*pid*>**`/oom_adj`** on older kernels.

6. Improvements for Distributed Configurations

GemStone/S 64 Bit version 3.0 includes changes to improve performance for distributed systems using remote caches, including systems in which some Gem processes are on nodes of a Wide Area Network (WAN), distant from the machine on which Stone is running. These changes include a number of enhancements that have been validated under heavy production load in the 32-bit GemStone/S product.

Mid-level caches

The page manager's management of remote caches now may use subordinate caches to reduce communications between the Stone machine and machines with remote caches. If a mid-level cache is in use, then for each Gem process using the mid-level cache, all the shared caches to which the Gems are attached are subordinate to that mid-level cache.

The page manager's pgsvr aggregates the responses from the pgsvrs on each of the gemSCs, and returns a combined response to the page manager. This reduces the number of round trips from the page manager to distant nodes.

It is assumed that if Gem processes are running on more than one machine far from the Stone, then all Gems physically close to each other relative to the WAN topology will be using a mid-level cache close to those Gems.

Added methods in System class include:

```
midLevelCacheConnect:
midLevelCacheConnect:cacheSizeKB:maxSessions:
midLevelCachesList
midLevelCachesReport
```

For more detailed documentation on using mid-level caches, see the *System Administration Guide for GemStone/S 64 Bit*, along with method comments in the image.

On Solaris and HP/Itanium, processes use /dev/poll

`/dev/poll` is an optimization over **`poll()`** available on the Solaris and HP/Itanium platforms, providing significantly improved performance as the number of connections increases. The Stone, shrpcmonitor and page manager have been modified to use **`/dev/poll`** rather than **`poll()`** for polling sockets for all network communications, on the platforms that support it.

Compression for lists of pages sent to remote caches for removal

The page manager sends lists of pages to be removed to each remote shared page cache. A new configuration parameter, STN_PAGE_MGR_COMPRESSION_ENABLED, has been added to allow this list of pages to be compressed, reducing network traffic. For details, see “STN_PAGE_MGR_COMPRESSION_ENABLED” on page 105.

More control over batch size for pages for page manager

The page manager requests pages from the Stone to be processed in batches of between 1 and 16384 pages. A configuration parameter controls the batch size; the page manager waits until this threshold is reached before requesting pages. As this value gets larger, the page manager will make less frequent requests for pages from the Stone, and process a larger number of pages when it does.

In previous releases, only the minimum batch size was controlled by the parameter STN_PAGE_REMOVAL_THRESHOLD.

In v3.0, this configuration parameter has been renamed to STN_PAGE_MGR_REMOVE_MIN_PAGES, and a related parameter, STN_PAGE_MGR_REMOVE_MAX_PAGES, has been added. The maximum has also been increased from 1792 to 16384.

For details, see "STN_PAGE_REMOVAL_THRESHOLD" on page 102 and "STN_PAGE_MGR_REMOVE_MAX_PAGES" on page 106.

Remote cache timeout is configurable

In previous releases, the timeout on a remote cache was fixed at 60 seconds. This could cause unnecessary timeouts and cache shutdown on slow networks.

In version 3.0, the timeout on response from a remote cache is configurable. For details, see "STN_REMOTE_CACHE_PGSVR_TIMEOUT" on page 106.

Warnings on slow responses from remote caches

When the page sends a request to a remote cache, after the time period defined by STN_REMOTE_CACHE_PGSVR_TIMEOUT is exceeded, the remote cache is considered dead and all Gems on that remote cache are terminated.

A new configuration parameter, STN_PAGE_MGR_PRINT_TIMEOUT_THRESHOLD, allows you to log warnings when the responses from the remote cache are taking a significant amount of time, but not as long as the remote cache timeout. This allows you to configure a long remote cache timeout but still monitor the response times of remote caches. For details, see "STN_PAGE_MGR_PRINT_TIMEOUT_THRESHOLD" on page 105.

Compressed page transfers

Two new configuration parameters have been added to enable compressed page transfers.

- GEM_PGSVR_COMPRESS_PAGE_TRANSFERS configures compression of page transfers between Gem and the Gem's pgsvr on the Stone's machine. The same configuration value is applied between the Gem's pgsvr on a mid-level cache and the Gem's pgsvr on the Stone's machine. Page transfers between a Gem and a pgsvr on a mid-level cache are not compressed, since it is assumed that mid-level cache is on a node close to the Gem process. For details, see "GEM_PGSVR_COMPRESS_PAGE_TRANSFERS" on page 103.
- STN_PAGE_MGR_COMPRESSION_ENABLED configures compression of the lists of pages that the page manager sends to remote shared page caches for removal. It only applies in system using remote caches. For details, see "STN_PAGE_MGR_COMPRESSION_ENABLED" on page 105.

More free pages per round trip

The new configuration parameter GEM_FREE_PAGEIDS_CACHE allows you to configure Gems to get more free pages from Stone each time the Gem needs free pages, to reduce number of round trips to Stone. For details, see "GEM_FREE_PAGEIDS_CACHE" on page 103.

Recycle buffers sent to remote caches

Rather than build a new, similar buffer to be sent to each remote cache, now the buffer is updated before sending to each cache. This is especially significant if compression is enabled (see "GEM_PGSVR_COMPRESS_PAGE_TRANSFERS" on page 103).

Cache information

The following methods have been added:

`System class >> remoteCachesList`
Returns an Array describing all shared caches that the Stone process is managing, not including the cache on the Stone machine. The result contains one element per cache. Each element of the result is a six-element Array containing:
hostName
a Boolean - true if cache was created as a mid-level cache
a SmallInteger - total number of sessions connected to the cache
a SmallInteger - max number of sessions for the cache
a SmallInteger - size of cache in KB
a SmallInteger - number of sessions using cache as a mid-level cache

`remoteCachesReport`
Returns the result of **`remoteCachesList`** formatted as a string, one line per cache.

7. Changes to Authentication and User Management

New login authentication using UNIX or LDAP

In addition to login authentication via the GemStone userId and GemStone password, version 3.0 includes the ability to authenticate login using UNIX user ID and password, or via an LDAP server. By default, login is via GemStone userId and GemStone password, as in previous releases.

The authentication scheme is configured for individual UserProfiles; the repository may contains UserProfiles using a mix of authentication schemes. However, special accounts, including SystemUser, DataCurator, GcUser, SymbolUser, and Nameless, must always use GemStone authentication.

#OtherPassword privilege is required to modify any UserProfile's authentication scheme.

LDAP authentication

An LDAP (Lightweight Directory Access Protocol) server consolidates and centralizes user authentication, and can now be used to authenticate logins to the GemStone server.

For LDAP authentication, in most cases the API requires the fully qualified name -- DN (distinguished name). You can either provide the complete DN directly, or perform a pattern-based search based on the basic information and the userId. The information is provided at the time the authentication is configured, not at login time, so a search option is particularly useful if the LDAP structure is likely to be modified.

When configuring LDAP authentication, there are two additional arguments: baseDn and filterDn. Setting the filterDn configures authentication to search for the DN; a nil filterDN disables the search.

To configure authentication to use a specific fully qualified DN, for example:

```
baseDn: 'uid=%s,ou=gss,dc=gemstone,dc=com' filterDn: nil.
```

To configure authentication to do a search for the given user:

```
baseDn: 'dc=gemstone,dc=com' filterDn: '(uid=%s)'
```

Methods to set up authentication

The following new instance methods on UserProfile specify authentication details for the given user account.

`enableUnixAuthenticationWithAlias:` *aUnixUserIdOrNil*

Modifies the UserProfile so that UNIX authentication is used to validate logins to GemStone. After this method has been run and successfully committed, all future logins of this UserProfile must specify the password for the given UNIX user ID. GemStone will then validate the password using UNIX password calls. If *aUnixUserIdOrNil* is nil, then the UserProfile's userId is used for the UNIX user ID.

You can determine if the UNIX userId exists using the new method **`System class >> unixUserIdExists:`**.

enableLDAPAuthenticationWithAlias: *aUserIdOrNil* **baseDn:** *baseDn* **filterDn:** *filterDn*
Modifies the UserProfile so that LDAP authentication is used to validate logins to GemStone. After this method has been run and successfully committed, all future logins by this UserProfile must specify the password for the given LDAP user ID. GemStone will then validate the password using an LDAP bind.

If *aUserIdOrNil* is nil, then the UserProfile's userId is used for the LDAP user ID.

Most LDAP binds require a fully qualified DN (distinguished name) to be specified in order to get a successful bind. There are two ways to get the fully qualified DN: by specifying it as baseDn, or by having GemStone perform a query to resolve the DN before attempting the bind.

If *filterDn* is a String, this filter will be used to search for the fully qualified DN, and the string must include the character sequence '%s' in place of the user ID. In this case, the *baseDn* argument must not contain the '%s' sequence.

If no DN search is to be performed, then nil should be passed in the *filterDn* argument and the *baseDn* string must include the character sequence '%s' where the userId is to be inserted in the string.

For more information on configuring GemStone for LDAP authentication, refer to the *System Administration Guide for GemStone/S 64 Bit*.

enableGemStoneAuthentication
Modifies the receiver so that GemStone authentication is used to validate logins to GemStone by the receiver. After this method has been run and successfully committed, all future logins of the receiver must specify a valid GemStone password.

Other useful methods to manage authentication

The following public instance methods on UserProfile have been added to query and manage authentication:

ldapSearchFilterDn
If the receiver is using LDAP authentication, returns a copy of the search filter string used to resolve the receiver's complete distinguished name during LDAP authentication; otherwise returns nil. Generates an error if the receiver is not your UserProfile and you do not have the #OtherPassword privilege.

ldapBaseDn: *aString* **searchFilterDn:** *stringOrNil*
Specifies the base distinguished name (DN) and optionally the search filter DN for LDAP logins done by the receiver. Has no meaningful effect unless LDAP authentication is enabled for the receiver. Requires the #OtherPassword privilege.

ldapBaseDn
If the receiver is using LDAP authentication, returns a copy of the base distinguished name (DN) string. Otherwise returns nil. Generates an error if the receiver is not your UserProfile and you do not have the #OtherPassword privilege.

authenticationSchemeIsUNIX
Answers true if the receiver uses UNIX for password authentication. Otherwise answer false. Requires the #OtherPassword privilege for any UserProfile other than your own.

authenticationSchemeIsLDAP
Answers true if the receiver uses LDAP for password authentication. Otherwise answers false. Requires the #OtherPassword privilege for any UserProfile other than your own.

authenticationSchemeIsGemStone
Answers true if the receiver uses GemStone for password authentication. Otherwise, answers false. Requires the #OtherPassword privilege for any UserProfile other than your own.

authenticationScheme
Returns a Symbol representing the authentication scheme for the receiver: either #GemStone, #UNIX, or #LDAP. Generates an error if the receiver is not your UserProfile and you do not have the #OtherPassword privilege.

Additional Password Control

In version 3.0, the following attributes may be set for each UserProfile:
passwordAgeLimit
passwordAgeWarning
staleAccountAgeLimit

These same attributes continue to exist in the AllUsers collection (UserProfileSet). The attributes in AllUsers now serve as system defaults, which can be overridden by attributes set in individual UserProfiles. If a given attribute is not set in a given UserProfile, then the global setting in AllUsers will apply. If a given attribute is set in a given UserProfile, the attribute in the UserProfile will be used, no matter what the same attribute is set to in AllUsers.

The special users, SystemUser, DataCurator, GsUser, SymbolUser, and Nameless, are not subject to password aging; these new features have no effect on them. (As in previous releases, the System-wide attributes do not apply to special users.) Also, if you are not using GemStone for password authentication, password aging is not relevant. Stale account aging, however, is valid when using non-GemStone password authentication.

A setting of nil in the UserProfile means to use the AllUsers value for that attribute, while a setting of 0 means that attribute is disabled.

To support this, the following new methods have been added to UserProfile:

staleAccountAgeLimit: *numberOfHours*
Sets the maximum number of hours for which the receiver will remain enabled without a login. Once the user logs in, he or she has up to this number of hours to login again, or the account is disabled. A disabled account requires another account with the #OtherPassword privilege to reset the password.

staleAccountAgeLimit
Returns the maximum number of hours for which the receiver will remain enabled without a login. Returns nil if the repository-wide staleAccountAgeLimit will be used to suspend stale accounts. Returns zero if the stale account age limit is disabled for the receiver.

setPasswordNeverExpires: *aBoolean*
Makes the receiver immune from all forms of password expiration. Once set, the

receiver is never required to change the password and will never be denied access because too much time has passed since the last successful login.

This method does not modify any password validity restrictions for the receiver such as maximum repeating characters, disallowed passwords, etc.

`passwordAgeWarning:` *numberOfHours*
Sets the maximum number of hours prior to a password expiration time for which a login as the receiver can succeed without a warning. If the user logs in to GemStone within this number of hours before the password is due to expire, GemStone issues a warning about the impending expiration. This feature grants a user the opportunity to change the password conveniently and to prevent the account from being disabled. Has no effect if a password age limit is not set for the receiver.

Setting this value to nil causes the receiver to use the repository-wide passwordAgeWarning setting. Setting this value to zero disables the password age warning for the receiver.

`passwordAgeWarning`
Returns the maximum number of hours prior to a password expiration time for which a login as the receiver can succeed without a warning. Returns nil if the repository-wide passwordAgeWarning will be used to generate password age warnings. Returns zero if password age warning is disabled for the receiver.

`passwordAgeLimit:` *numberOfHours*
Sets the maximum number of hours for which the receiver can retain a password. When a password is set, it expires this number of hours later. The user must change the password before it expires, or the account is disabled. A disabled account requires another account with the #OtherPassword privilege to reset the password.

Setting this value to nil causes the receiver to use the repository-wide passwordAgeLimit setting in AllUsers. Setting this value to zero disables password aging for the receiver.

`passwordAgeLimit`
Returns the maximum number of hours for which the receiver can retain a password before the account expires. Returns nil if the repository-wide passwordAgeLimit will be used to enforce the password age limit. Returns zero if password aging is disabled for the receiver.

Additional password management

Version 3.0 includes several new ways to manage the user's choice of passwords. These features are repository-wide, managed in AllUsers (UserProfileSet), not on a per-UserProfile basis.

Disallow specific number of previous passwords

The existing disallowUsedPasswords feature, which prevents users from using any of an unlimited number of previous passwords, has been enhanced. In version 3.0, a specific count of previously-used passwords can be disallowed, so for example, the last 10 passwords are disallowed, but passwords used earlier than that may be reused.

numberOfUsedPasswordsDisallowed: *aSmallInt*
Sets the number of previous passwords that may not be reused by users when changing their password. *aSmallInt* must be a SmallInteger in the range of 0 to 65535. A value of zero means users may not reuse any previous password. This method only has effect if disallowUsedPasswords is also set to true.

numberOfUsedPasswordsDisallowed
Returns a SmallInteger between 0 and 65535 which is the number of previous passwords which may not be reused by users when changing their password. A value of zero means users may not reuse any previous password. This method only has effect if disallowUsedPasswords is also set to true.

Passwords can be required to include multiple character types

In version 3.0, you can now require passwords to contain at least one uppercase character, symbol, lowercase character, and/or digit. For GemStone's password management, a symbol is any character that returns false to the message #isAlphaNumeric; in practice, this includes whitespace and control characters.

The following new methods on UserProfileSet support these new features:

passwordRequiresUppercase: *aBoolean*
Enables or disables the requirement that new passwords contain at least one uppercase character.

passwordRequiresUppercase
Answers true if passwords changed by users are required to contain at least one uppercase character.

passwordRequiresSymbol: *aBoolean*
Enables or disables the requirement that new passwords contain at least one symbol character. In the scope of passwords, a symbol character is any character that answers false to the message #isAlphaNumeric.

passwordRequiresSymbol
Answers true if passwords changed by users are required to contain at least one symbol character. In the scope of passwords, a symbol character is any character that answers false to the message #isAlphaNumeric.

passwordRequiresLowercase: *aBoolean*
Enables or disables the requirement that new passwords contain at least one lowercase character.

passwordRequiresLowercase
Answers true if passwords changed by users are required to contain at least one lowercase character.

passwordRequiresDigit: *aBoolean*
Enables or disables the requirement that new passwords contain at least one digit.

passwordRequiresDigit
Answers true if passwords changed by users are required to contain at least one digit.

Creating and adding users

In previous releases, much user creation protocol included the **`defaultSegment:`** keyword, which is now replaced by **`defaultObjectSecurityPolicy:`** (see "Segment name change to GsObjectSecurityPolicy" on page 19). In addition, some protocol included the **`compilerLanguage:`** keyword, which has long been obsolete. Some old user creation protocol remains for backwards compatibility, but is deprecated.

Changes in methods for creating and adding users

v2.x protocol	v3.0 protocol that replaces v2.x protocol
`UserProfileSet >>` **`addNewUserWithId:`** **`password:`**	This method exists in 3.0, but the behavior has changed. Instead of creating a new default Segment for the new user, the new user is created with a nil default GsObjectSecurityPolicy. To get behavior of previous releases, use: **`UserProfileSet >>`** **`addNewUserWithId:`** **`password:`** **`createNewSecurityPolicy:`** *true*
`UserProfileSet >>` **`addNewUserWithId:`** **`password:`** **`defaultSegment:`**	Exists but is deprecated in 3.0. Replace with: **`UserProfileSet >>`** **`addNewUserWithId:`** **`password:`** **`defaultObjectSecurityPolicy:`**
`UserProfileSet >>` **`addNewUserWithId:`** **`password:`** **`defaultSegment:`** **`privileges:`** **`inGroups:`**	Exists but is deprecated in 3.0. Replace with: **`UserProfileSet >>`** **`addNewUserWithId:`** **`password:`** **`defaultObjectSecurityPolicy:`** **`privileges:`** **`inGroups:`**
`UserProfileSet >>` **`addNewUserWithId:`** **`password:`** **`defaultSegment:`** **`privileges:`** **`inGroups:`** **`compilerLanguage:`**	Removed in 3.0; **`compilerLanguage:`** is long obsolete.
	Method added in 3.0, which creates a UserProfile with a nil default GsObjectSecurityPolicy. **`UserProfile class >>`** **`newWithUserId:`** **`password:`**

Changes in methods for creating and adding users (Continued)

v2.x protocol	v3.0 protocol that replaces v2.x protocol

	Method added in 3.0, which creates a UserProfile with a given default GsObjectSecurityPolicy.
	`UserProfile class >>`
	`  newWithUserId:`
	`  password:`
	`  defaultObjectSecurityPolicy:`

`UserProfile class >>`
`  newWithUserId:`
`  password:`
`  privileges:`
`  inGroups:`
	Unchanged in 3.0.

`UserProfile class >>`
`  newWithUserId:`
`  password:`
`  defaultSegment:`
`  privileges:`
`  inGroups:`
	Exists but is deprecated in 3.0. Replace with:
	`UserProfile class >>`
	`  newWithUserId:`
	`  password:`
	`  defaultObjectSecurityPolicy:`
	`  privileges:`
	`  inGroups:`

`UserProfile class >>`
`  newWithUserId:`
`  password:`
`  defaultSegment:`
`  privileges:`
`  inGroups:`
`  compilerLanguage:`
	Removed in 3.0; `compilerLanguage:` is long obsolete.

Removing users

The process of removing a UserProfile from AllUsers has changed significantly.

In previous releases, instances of UserProfile were removed from AllUsers by using regular Collection protocol such as `remove:`, `remove:ifAbsent:`, etc. This, however, did not properly clean up the removed user. Any Segments owned by the otherwise removed user kept a reference to the UserProfile, and therefor any objects references by that UserProfile (see "Risk of garbage when UserProfile removed" on page 132).

As a result, version 3.0 introduces replacement methods for removing users. These replacement methods reassign Segments (now GsObjectSecurityPolicies) owned by the removed user to either the current UserProfile, or another user, depending on the specific method used to remove the user.

Also new in version 3.0, when a user is removed, an instance of the new class DeletedUserProfile is created. The DeletedUserProfile includes the userId, the symbolList of the user, and the current DateTime. This instance of DeletedUserProfile is put in an IdentitySet #AllDeletedUsers in Globals. The classes and references in the deleted user's SymbolList should be reviewed and cleaned up, as a separate manual process, by an administrative user.

A similar process for removing users was previously performed by GBS code, but was not performed by server code.

To support the new process for removing UserProfiles, all Collection **`remove:`** protocol is now disallowed for AllUsers. If your application includes code for removing users, you will need to update your application.

The following new UserProfileSet instance methods replace the previous ways of removing UserProfiles from AllUsers. These methods require write access to the GsObjectSecurityPolicies for both DataCurator and the UserProfile instance to be removed, and require the #OtherPassword privilege.

`removeAndCleanup:` *aUserProfile*
Removes the given UserProfile from the receiver. Transfers ownership of any GsObjectSecurityPolicy objects owned by *aUserProfile* to the UserProfile for the current session. A new DeletedUserProfile with information related to *aUserProfile* is added to the AllDeletedUsers collection in Globals.

`removeAndCleanup:` *aUserProfile* **`migrateSecurityPoliciesTo:`** *anotherUserProfile*
Removes the given UserProfile from the receiver. Transfers ownership of any GsObjectSecurityPolicy objects owned by *aUserProfile* to *anotherUserProfile*. A new DeletedUserProfile with information related to *aUserProfile* is added to the AllDeletedUsers collection in Globals.

`removeAndCleanupUserWithId:` *aUserId*
`migrateSecurityPoliciesToUserWithId:` *anotherId* **`ifAnyAbsent:`** *aBlock*
Removes the UserProfile for *aUserId* from the receiver. Transfers ownership of any GsObjectSecurityPolicy objects owned by the UserProfile for *anotherId*. A new DeletedUserProfile with information related to the removed UserProfile is added to the AllDeletedUsers collection in Globals. If the UserProfile for either *aUserId* or for *anotherId* is not found, executes *aBlock* and returns.

`removeAndCleanupUserWithId:` *aUserId* **`ifAbsent:`** *aBlock*
Removes the UserProfile for *aUserId* from the receiver. Transfers ownership of any GsObjectSecurityPolicy objects owned by the UserProfile being removed to the UserProfile for the current session. A new DeletedUserProfile with information related to the removed UserProfile is added to the AllDeletedUsers collection in Globals. Executes *aBlock* if the UserProfile could not be found.

New feature to disable accounts

Accounts can now be disabled manually, resulting in a state similar to that resulting from password expiration. These methods require #OtherPassword privilege.

To re-enable a disabled account, another account with the #OtherPassword privilege must send the disabled account the **`password:`** message to reset the password, thus re-enabling the account.

`disable`
Disables the receiver when the transaction is committed.

`disableWithReason:` *aString*
Disables the receiver when the transaction is committed and specifies *aString* as the reason.

Updating lastLoginTime

The UserProfile's lastLoginTime is not updated automatically by the system unless password aging or stale account aging is enabled. This avoids the overhead of an extra commit at each login.

However, initially enabling password or stale account aging can mean that existing accounts are disabled. To avoid that, you can manually execute the following method for each account that is subject to password or stale account aging, using the current DateTime (**`DateTime now`**), or nil. Accounts with a nil lastLoginTime do not fail aging tests.

`UserProfile >> lastLoginTime:` *aVal*
Updates the lastLoginTime for the receiver to be *aVal*, which must be an instance of DateTime or nil.

Modifying privileges from Special accounts is now disallowed

It is no longer allowed to add or delete privileges for any of the special accounts: SystemUser, SymbolUser, GcUser, DataCurator, and Nameless.

Users disabled immediately rather than on next login

Previously, users that were due to be disabled - due to password expiration, for example - were not identified as disabled until they logged in. Messages such as **`findDisabledUsers`** did not return these potentially-disabled users. Now, potentially disabled users are detected before the user logs in, and returned by queries for disabled status.

8. Exception Changes

GemStone includes mechanisms to allow applications to signal exceptions and define custom exceptions, and mechanisms to catch exceptions generated by GemStone or by application code.

In previous versions of GemStone, the exception framework consisted of a code-based legacy exception framework, with an extension to support ANSI class-based exceptions. The legacy protocol distinguished more than 500 GemStone errors by category and number. Exception handlers could be defined using either the legacy or ANSI protocol, and Exceptions could be signaled using either legacy or ANSI protocol.

In GemStone/S 64 Bit v3.0, there is native support for ANSI Exceptions, avoiding some of the problems caused by dependency on the previous underlying legacy framework. The ANSI Exception framework now extends the previous class hierarchy by defining further subclasses of Exception, to match more closely the granularity of errors that GemStone users may want to handle.

The legacy handler protocol has been deprecated with version 3.0. In v3.0, all exceptions are ANSI exceptions. Methods that previously signaled legacy exceptions now signal ANSI exceptions. For backwards compatibility, version 3.0 supports the use of legacy handlers to catch ANSI exceptions.

We strongly encourage the use of ANSI protocol for handling all errors. Nonetheless, you may continue to use the legacy protocol for handling GemStone system errors.

In order to define, create, or signal exceptions, you must use the ANSI protocol. If you have defined custom exceptions based on the legacy interface, you will need to redefine these in version 3.0.

NOTE

In previous GemStone/S 64 Bit versions, the legacy exception framework included error categories. In version 3.0, categories are no longer supported. If you signaled and handled your own error categories in a previous GemStone version, you must rewrite your application to use ANSI exceptions in version 3.0.

The diagram on the following pages shows the new ANSI exception handler class hierarchy. Note that Exception is the highest class that customer Smalltalk code should ever handle.

Exception Class Hierarchy

```
AbstractException ( gsResumable gsTrappable gsNumber currGsHandler
                gsStack gsReason gsDetails tag messageText gsArgs )
   Exception
        ControlInterrupt
           Break
           Breakpoint  ( context stepPoint )
           ClientForwarderSend  ( receiver clientObj selector )
           Halt
           TerminateProcess
        Error
           CompileError
           EndOfStream
           ExternalError
              IOError
                    SocketError
              SystemCallError  ( errno )
           GciError  ( errorDescription externalSession )
              GciCompileError  ( list )
              GciEventError
              GciPause
              GciRuntimeError
              GciApplicationError
              GciDoesNotUnderstand
           ImproperOperation  ( object )
              ArgumentError
              ArgumentTypeError  ( expectedClass actualArg )
              CannotReturn
              LookupError  ( key )
              OffsetError  ( maximum actual )
              OutOfRange  ( minimum maximum actual )
                    FloatingPointError
              RegexpError
           IndexingErrorPreventingCommit
           InternalError
              GciTransportError
           InvalidUtf8Error
           LockError  ( object )
           MarkerNotFound
           NameError  ( selector )
              MessageNotUnderstood  ( envId receiver )
           NumericError
              ZeroDivide  ( dividend )
           RepositoryError
           SecurityError
           SignalBufferFull
           ThreadError
           TransactionError
           UncontinuableError
           UserDefinedError
```

```
      ExceptionSample
      ExceptionSampleNotResumable
      ExceptionSampleResumable
Notification
   Admonition
      AlmostOutOfMemory
      AlmostOutOfStack
      RepositoryViewLost
   Deprecated
   FloatingPointException
   InterSessionSignal  ( sendingSession signal )
   ObjectsCommittedNotification
   TransactionBacklog  ( inTransaction )
   Warning
      CompileWarning
TestFailure
   ResumableTestFailure
```

Default exception handlers

The GemStone/S 64 Bit legacy exception handling framework provides for zero or more "dynamic" handlers (on the process execution stack), zero or more "static" handlers (**`Exception class installStaticException:category:number:`**), and a GCI handler.

The version 3.0 ANSI exception handling framework provides for zero or more stack-based handlers and a list of zero or more default handlers, ordered in the sequence they were installed.

Since ANSI does not provide a direct API for adding and removing default handlers at runtime, version 3.0 provides the following methods to deal with default handlers in the context of the ANSI framework.

`Exception class >> addDefaultHandler:` *aOneArgumentBlock*
Creates a GsExceptionHandler that understands the message **`remove`** and adds the new handler to the beginning of the **`defaultHandlers`** list. After *aOneArgumentBlock* (equivalent to the second argument to **`on:do:`**) is invoked, the argument (an instance of Exception or one of its subclasses) responds appropriately to **`pass`** and **`outer`** seamlessly between stack-based and default handlers.

`Exception class >> defaultHandlers`
Returns a SequenceableCollection (or subclass) of GsExceptionHandler instances that will catch instances of the receiver (typically, a subclass of Exception). The result does not include any legacy static handlers. This collection may be empty and typically is a subset of the installed default (static) handlers.

`GsExceptionHandler >> remove`
Searches both the Smalltalk stack and the list of default handlers and removes the receiver if found. Returns receiver if found, otherwise generates an error

Identifying Exception class corresponding to Error

You can determine the Exception class corresponding to an error number or symbol using the method **`Exception class >> legacyNumberToClasses:`** *anInteger*.

For example,

```
Exception legacyNumberToClasses: 6009
```

or

```
Exception legacyNumberToClasses:
   (ErrorSymbols at: #rtErrSignalAbort)
```

Changes in Common GemStone Event Exceptions

Version 3.0 exception class Legacy symbol (and number)	Description
TransactionBacklog *#rtErrSignalAbort (6009)* *#rtErrSignalFinishTransaction (6012)*	When System inTransaction returns false (running outside a transaction), Stone requested Gem to abort. This error is generated only if you have executed the method **`enableSignaledAbortError`**. When System inTransaction returns true (the session is in transaction), Stone has requested the session to commit, abort, or continue (with continueTransaction) the current transaction. This error is received only if you have executed the method **`enableSignaledFinishTransactionError`**.
ObjectsCommittedNotification *#rtErrSignalCommit (6008)*	An element of the notify set was committed and added to the signaled objects set. This error is received only if you have executed the method **`enableSignaledObjectsError`**.
InterSessionSignal *#rtErrSignalGemStoneSession (6010)*	Your session received a signal from another GemStone session. This error is received only if you have executed the method **`enableSignaledGemstoneSessionError`**. Arguments: 1. The session ID of the session that sent the signal. 2. An integer representing the signal. 3. A message string.
AlmostOutOfMemory *#rtErrSignalAlmostOutOfMemory (6013)*	Temporary object memory for the session is almost full. The error is deferred if in user action or index maintenance.
RepositoryError *#rtErrTranlogDirFull (2339)*	All available transaction log directories or partitions are full. This error is received if you are DataCurator or SystemUser, otherwise only if you have executed the method **`enableSignalTranlogs-Full`** in this session.
RepositoryViewLost *#abortErrLostOtRoot (3031)*	While running outside a transaction, Stone requested Gem to abort. Gem did not respond in the allocated time, and Stone was forced to revoke access to the object table.

Behavior change in message not understood retry handling

In previous releases, there was an implicit retry in the error handling for a message not understood exception (in the absence of an explicit return from the handler block).

For example, in previous releases, the following code returned 43:

```
| x |
Exception
   category: GemStoneError
   number: 2010
   do: [:ex :cat :num :args |
      Object
         compileMethod: 'foo' , Character lf asString , '^42'
         dictionaries: #()
         category: 'test'.
      999.
   ].
x := Object new foo + 1.
x
```

In version 3.0, this returns 1000. If your application code depends on this automatic retry from previous releases, you will need to modify the handlers to accommodate the new behavior.

On a return from a GciContinue() call, the does-not-understand processing will automatically retry.

Changes in nested block variable scoping

The fix for bug #38359 introduces a change in scoping that may break existing code that relied on the existing, incorrect variable scope. You may need to declare variables used within nested blocks in an outer scope.

9. Changes in Number classes

GemStone/S 64 Bit version 3.0 incorporates changes to the Number classes for ANSI compliance and for optimization.

LargeInteger replaces LargePositiveInteger / LargeNegativeInteger

There is a new implementation for large Integers. A single class, LargeInteger, is used for both large positive and negative integers, replacing LargePositiveInteger and LargeNegativeInteger. As with other reimplemented classes, in-memory instances are now in CPU native byte order, regardless of the byte order of the repository, by swizzling when they are loaded into memory.

During conversion, the classes LargePositiveInteger and LargeNegativeIntegers are converted to ObsoleteLargePositiveInteger and ObsoleteLargeNegativeInteger. Instances are auto-mutated when they are faulted into the VM. To explicitly convert instances, execute **`Number >> convert`**.

The maximum absolute value for a LargeInteger is $(2^{130144} - 1)$. Attempts to create a LargeInteger that exceeds this maximum will fail with an Integer overflow error.

ScaledDecimal reimplemented

ScaledDecimal has been reimplemented in version 3.0. The new format is composed of an integer mantissa (which may be a LargeInteger) and a power-of-10 scale. This new format represents decimal fractions to the precision specified. The old fraction-based format represented rational numbers with unbounded precision, but did not always represent any decimal fraction precisely.

The new ScaledDecimal implementation conforms to the ANSI specification.

The maximum scale is 30000.

There is a new instance creation method for ScaledDecimal:

`ScaledDecimal class >> with:` *aNumber* **`scale:`** *aScale*
Returns an instance of the receiver, having value *aNumber* and scale *aScale*.

Per ANSI, ScaledDecimals use a literal notation using s, such as 1.23s2. In version 3.0, this s notation refers to a literal of the new ScaledDecimal class.

FixedPoint

The previous implementation of Scaled Decimal has been renamed to FixedPoint, and remains available. FixedPoint uses the reserved OOP of the previous ScaledDecimal class. The FixedPoint class allows close compatibility with the VisualWorks FixedPoint class, which has similar behavior.

FixedPoint literals are defined with a new literal syntax that uses the p character instead of the s, for example 1.23p2. This notation otherwise follows the same rules as the s syntax.

Converting numbers to ScaledDecimal and FixedPoint

To convert numeric types to FixedPoint and the new ScaledDecimal, the following methods are provided:

`asFixedPoint:` *aScale*
`asScaledDecimal:` *aScale*

Scale of ScaledDecimal results

ANSI does not precisely specify the scale of a ScaledDecimal that is returned by mathematical operations. The implementation of ScaledDecimal in version 3.0 uses the following rules:

- For unary messages, the scale of the result should equal the scale of the receiver.
- For a one-argument message, the scale of the result should be the greater of the scale of the receiver and argument. An integer receiver or argument coerced to a ScaledDecimal should effectively have a scale of zero, meaning the result will have the scale of the non-coerced ScaledDecimal argument or receiver.

Rounding of ScaledDecimal results

For some mathematical operations, the returned value type is a ScaledDecimal, but the returned value cannot be exactly represented as a ScaledDecimal with the correct scale. In these cases, the results are rounded using the following rules:

- Following the example of IEEE488 float rounding, the ScaledDecimal that is answered is selected as though we computed the numerically exact value and then chose the closest representable ScaledDecimal of the scale specified by the rules.
- If the numerically exact value falls exactly halfway between two adjacent representable ScaledDecimal values of the scale specified by the rules, the ScaledDecimal with an even least significant digit is answered.

Upgrade and conversion of ScaledDecimals

During conversion to 3.0, references in application code to “ScaledDecimal” and literal references using the s notation (for example, 1.23s2) will be converted to the new ScaledDecimal implementation during the recompilation of application code (required by conversion).

Note, however, that persistent instances of the old ScaledDecimal will not be converted; they will remain instances of the class that is now named FixedPoint. A manual step is required to replace the persistent instances of FixedPoint with new instances of ScaledDecimal. Contact GemStone Technical Support if you require scripts to assist with this post-conversion step.

If you wish to continue to use the same class, now named FixedPoint, rather than the new ScaledDecimal, you must modify your application code. All references to “ScaledDecimal” must be changed to “FixedPoint”, and any references using the s notation must be changed to the p notation (for example, 1.23s2 must be changed to 1.23p2).

ScaledDecimal for GBS for VisualWorks

The new GemStone FixedPoint class is similar to the VisualWorks FixedPoint class. The server FixedPoint class will be mapped to the VisualWorks FixedPoint class, as in previous versions.

There is no equivalent in VisualWorks to the new ScaledDecimal class. GBS adds a GbsScaledDecimal class on the client, similar to the new ScaledDecimal implementation, and maps this to the server ScaledDecimal.

There is no s literal syntax for GbsScaledDecimal. VisualWorks s literals are mapped to instances of GemStone FixedPoint.

ScaledDecimal for GBS for VA Smalltalk

VA Smalltalk includes a ScaledDecimal class that does not closely match either the old or new GemStone/S 64 ScaledDecimal class. In future GBS releases that support both version 3.0 and VA Smalltalk, the server FixedPoint class will be mapped to the VA Smalltalk ScaledDecimal class, using the replication mapping as in previous versions. GBS adds a GbsScaledDecimal class on the client, similar to the new ScaledDecimal implementation, and maps this to the server ScaledDecimal.

Float class reimplemented

Float and SmallFloat, along with several MultiByteString subclasses, have been reimplemented for efficiency in byte storage. In-memory instances are in CPU native byte order, regardless of the byte order of the repository, by swizzling when they are loaded into memory. The v2.x class is renamed to ObsFloat and ObsSmallFloat.

In upgraded repositories, instances of Float or SmallFloat are auto-mutated into instances of the new class when they are loaded into memory. The new method **`convert`** can be sent to such in-memory instances, in which case the a subsequent commit will make the mutation persistent.

SmallFloat remains deprecated.

Additional ANSI compatibility for Float

ANSI specifies globals FloatE, FloatD, and FloatQ, as well as Float which is equivalent to either FloatE, FloatD, or FloatQ. The GemStone/S 64 Globals SymbolDictionary now includes mappings from FloatE, FloatD and FloatQ to the Float class.

In addition, literal floats now accept the exponent letter q as well as e and d.

Note that SmallDouble is now a subclass of Float, so Float behavior is inherited; the location of methods may have changed in this release.

The following methods are required by ANSI and have been added to Float class:

`denormalized`
Answers whether the floating point representation allows denormalized numbers.

`e`
The mathematical constant e as representable in an 8-byte IEEE Float.

`emax`
The largest base-10 exponent supported by Float.

emin
The smallest base-10 exponent supported by Float.

epsilon
Smallest Float such that (1.0 + epsilon) = 1.0) == false

fmax
The largest Float smaller than PlusInfinity.

fmin
The smallest positive Float. (Same as **fminDenormalized**.)

fminDenormalized
Smallest positive Float.

fminNormalized
Smallest positive Float that is not a subnormal Float.

precision
The number of radix digits in the floating point representation. This is the number of bits in the mantissa of a Float plus one.

radix
The base of the floating point representation.

Float additional mathematical operations

The following instance methods for mathematical functions have been added to Float:

arcCosh	**erf**	**log2**
arcSinh	**erfc**	**modf**
arcTanh	**frexp**	**modulo:**
arcTan2	**hypot:**	**sinh**
cosh	**ldexp:**	**tanh**

For details about any of these methods, see the image comments.

Change to Integer >> bitAt: for ANSI compliance

The GemStone implementation of **Integer >> bitAt:** has always returned the 1-based offset. The ANSI standard specifies that it should be 0-based. This method has been changed to be ANSI compliant, and now returns on 0-based offset.

Any application use of this method are likely to need adjustment.

Other added ANSI methods

The following methods are implemented for all Number types:

integerPart
Returns an instance of the receiver class whose fraction part has been discarded. (If the receiver is an instance of Fraction, returns an integer.)

fractionPart
Returns the fraction remaining after the receiver is truncated toward zero.

The following method has been added to Integer:

bitAt:put:

ANSI-compliant operation return types

ANSI specifies the type of result returned from a numeric operations, where the operands are of different types. While some improvements have been made over past releases, an effort has been made to ensure that the correct type of result is returned for all operations and operand types.

Printing of DecimalFloats

DecimalFloat do not have a specific literal notation, and previously they were printed in binary float notion, using the **`E`** character. Now, DecimalFloats are printed using the ScaledDecimal notation, which is more compatible, using the **`F`** character; for example, **`1.234F5`**

asFraction now returns Integer

Previously, **`asFraction`** would always return an instance of Fraction, for example, 4/1. The ANSI standard allows the result of **`asFraction`** to be a rational. Now, if the receiver of asFraction can be reduced to an Integer, this method returns that Integer.

Random number generator

The old, unsupported random number generators included under the examples directory -- the classes Random and FastRandom -- are no longer available. These are replaced by an improved set of classes to generate random numbers for various purposes.

The class hierarchy of the new Random classes are:

```
Object
   Random (abstract)
      HostRandom
      SeededRandom (abstract)
         Lag1MwcRandom
         Lag25000CmwcRandom
```

Random

The abstract superclass; you will use a subclass of this for most purposes. In most cases, the recommended subclass is Lag25000CmwcRandom. For convenience, interface methods on Random will create instances of Lag25000CmwcRandom:

`new`
Answers a Lag25000CmwcRandom with an initial random seed generated from the host OS **`/dev/urandom`**.

`seed:` *aSmallInteger*
Answers a Lag25000CmwcRandom with an initial seed generated from the given SmallInteger.

Once you have an instance of any subclass of Random, you can generate random numbers with these messages:

`float`
Answers a random Float in the range [0,1)

floats: *n*
Answers a collection of *n* random floats in the range [0,1)

integer
Answers a random non-negative 32-bit integer

integers: *n*
Answers a collection of *n* random non-negative 32-bit integers

integerBetween: *l* **and:** *h*
Answers a random integer in the given range

integers: *n* **between:** *l* **and:** *h*
Answers a collection of *n* random integers in the given range

smallInteger
Answer a random integer in the full SmallInteger range, [-2**60..2**60)

It is recommended to create an instance of a Random subclass and use that instance to generate many random numbers. Creating multiple instances of Random subclasses and using each one only a few times is much more expensive.

HostRandom

HostRandom allows access to the host operating system's **/dev/urandom** random number generator.

HostRandom is much slower than the other subclasses of Random. However, **/dev/urandom** on some platforms may be intended to be cryptographically secure random number generator, which none of the other subclasses are. It also has the advantage of not needing an initial seed, and so is good for generating random seeds for the faster Random subclasses.

HostRandom uses a shared singleton instance, which is accessed by sending #new to the class HostRandom. Sending #new has the side effect of opening the underlying file **/dev/urandom**. This file normally remains open for the life of the session, but if you wish to close it you can send #close to the instance, and later send #open to reopen it. If you store a persistent reference to the singleton instance the underlying file will not be open in a new session and you must send #open to the instance before asking for a random number.

Since HostRandom is a service from the operating system, it cannot be seeded, and should not be used when a repeatable random sequence of numbers is needed.

SeededRandom

SeededRandom is an abstract superclass for classes that generate sequences of random numbers that can be generated repeatedly by giving the same initial seed to the generator.

In addition to creating new instances using the class methods new and seed:, the following instance methods allow repeatable sequences to be generated:

seed: *aSmallInteger*
Sets the seed of the receiver from the given seed, which can be any SmallInteger. The subsequent random number sequence generated will be the same as if this generator had been created with this seed.

`fullState`, **`fullState:`** *stateArray*

The internal state of a generator is more than can be represented by a single SmallInteger. These messages allow you to retrieve the full state of a generator at any time, and to restore that state later. The random number sequence generated after the restoration of the state will be the same as that generated after the retrieval of the state. You might, for instance, allow a generator to get its initial state from /dev/urandom, then save this state so the random sequence can be repeated later.

Lag1MwcRandom

Lag1MwcRandom is a seedable random number generator with a period of about 10**18. It is a lag-1 generator using the multiply-with-carry algorithm to generate random numbers. It is slower than Lag25000CmwcRandom, its period is much shorter, it is of limited use in generating random n-tuples, and it has been shown that the random bits of the MWC algorithm are not quite perfectly fair, so for most uses Lag25000CmwcRandom is superior to this class.

The primary advantage of Lag1MwcRandom is that it can be seeded by a single 61-bit SmallInteger, whereas Lag25000CmwcRandom requires a seed of more than 800000 bits. Lag25000CmwcRandom>>seed: uses a Lag1MwcRandom to generate all the seed bits from the given SmallInteger seed. Other uses are not recommended.

Lag25000CmwcRandom

Lag25000CmwcRandom is a seedable random generator with a period of over 10**240833. It is a lag-25000 generator using the complementary multiply-with-carry algorithm to generate random numbers. It is the fastest subclass of Random. Its period is so long that every possible sequence of 24994 successive 32-bit integers appears somewhere in its output, making it suitable for generating random n-tuples where n<24994. Its output is fair in that the number of 0 bits and 1 bits in the full sequence are equal.

This generator is recommended for most uses, but it is *not* a cryptographically secure generator, so for applications like key generation you should consider using HostRandom, once you satisfy yourself that HostRandom is secure enough on your operating system.

You can also allow the seed bits to be initialized from the HostRandom, then retrieve that state by sending #fullState. That state can later be restored by sending the retrieved state as an argument to #fullState:.

10. GsSocket and GsFile Changes

GsSocket and GsFile have been reimplemented in version 3.0. The old classes GsSocket and GsFile are renamed ObsoleteGsSocket and ObsoleteGsFile. The new GsSocket and GsFile classes are subclasses of the new class IO.

The way in which these classes retain information within the C code has been rewritten. This change should be transparent to the user; the new classes should be more robust.

The IO class includes a new instance variable, fileDescriptor, which provides the OS file descriptor for the file.

Hash change

The GsFile hash computation has changed, and the **`GsFile >> id`** method has been removed. In previous release, the hash used this id. In version 3.0, the pathname is the basis for the hash value.

The undocumented method **`GsFile >> newWithFilePtr:pathname:mode:onClient:`** has been removed.

Directory creation with mode

Using GsFile to create a directory now creates a directory that has no permissions for world; the default umask is xor-ed with 770. To allow directory creation with specific permissions, the following class method has been added

`createServerDirectory:` *aPathName* **`mode:`** *modeInt*
Creates the named directory on the server machine. Returns the receiver if the directory was created, nil if an error occurs. *modeInt* should be a SmallInteger, the value of which will be used as the second arg to mkdir(). If *modeInt* == nil, a value of 8r770 will be used.

This is only available for server directory creation, not for client directory creation.

Added methods on GsFile

The following methods have been added to GsFile:

`copy`
Creates a copy of the receiver. The copy will be closed.

`nextLineTo:` *eolValue*
Reads the next line from the receiver, up to and including the *eolValue*. *eolValue* is either 0...255 representing the value of a Character, or a String with a size between 1 and 128.

`nextLineTo:` *eolValue* **`prompt:`** *promptString*
This method is intended to be used with the new Topaz line editor. (See "Line editor" on page 121.) If the prompt is not nil, and the receiver is stdin, and stdin is a terminal device, then the Topaz line editor is used to read the input.

`nextLineToChar:` *eolCharacter*
Reads the next line from the receiver, up to and including the *eolCharacter*. *eolCharacter* is a Character.

read: *numberOfBytes* **into:** *byteObj*
Reads up to the given number of bytes into the given byte object (for example, a String) starting at index 1. *byteObj* is grown if necessary.

space
Appends a space character to the receiver.

tab
Appends a tab to the receiver.

write: *amount* **from:** *byteObj*
Writes the given number of bytes from the given byte object.

Other methods have also been added for ANSI compatibility.

GsSocket changes

GsSocket has significant implementation changes in version 3.0.

The way that instances of GsSocket wait has changed. The process scheduler is now involved in managing GsSocket waits. Instance variables now track waits for read and waits for write for each socket. The following related methods have been added: **hasReadWaiter:**, **hasWaiters**, **hasWriteWaiter:**, **readWaiters**, and **writeWaiters**.

The following additional methods have been added to GsSocket:

GsSocket >> copy
Creates a copy of the receiver.

GsFile >> newUdp
Creates a new socket of type User Datagram Protocol (UDP)

The following obsolete methods have been removed:

GsSocket >> listen:

GsSocket >> listen:acceptingWith:

GsSocket class >> newWithId:.

11. New Multi-Threaded Operations

Most operations that scan the entire repository can now be performed by multiple threads running in parallel. These operations include markForCollection (MFC), findDisconnectedObjects (FDC), listInstances, listReferences:, objectAudit, and other operations. By scanning in parallel, these operations can be completed much more quickly than in earlier versions of GemStone, using all system resources to the maximum. These operations can also be tuned to limit use of resources, allowing the operation to run in the background while other application activity takes precedence.

The base multi-threaded scan operations include two new arguments that provide the initial settings for the scan:

- **`withMaxThreads:`** the maximum number of slave sessions (threads) that can be used during a scan operation. This is an upper limit; the actual number of threads may vary or be modified manually during the course of the scan. This maximum value is recorded in the cache statistic **MtMaxThreads**.
- **`maxCpuUsage:`** the maximum CPU usage, which limits percentage of total CPU activity at which GemStone automatically deactivates threads to prevent overload of system resources. This may be modified manually during the course of the scan. The value is recorded in the cache statistic **MtPercentCpuActiveLimit**.

The master thread of the multi-threaded scan uses the CPU usage limit, along with the threads limit, to determine how many threads to run. The number of threads running will be between 0 and the current thread limit (**MtThreadsLimit**). The scan begins with the **MtThreadsLimit** set to the maximum specified in the argument (*mtMaxThreads*). If this results in CPU usage that is over the specified maximum, threads are deactivated until the number of threads is low enough to keep CPU usage under the specified limit. The actual number of threads running at any given time is recorded in the cache statistic **MtActiveThreads**.

Two new cache statistics are involved in tuning the multi-threaded operations:

- **MtThreadsLimit** —The current upper limit on the number of threads performing the scan.
- **MtPercentCpuActiveLimit** —The current maximum CPU usage.

These are "configurable statistics" – they appear in the cache statistics for the session performing the operation, and they can be set explicitly while the operation is in progress to modify the behavior of the operation.

You can manually decrease the upper limit on the number of threads that may be running by modifying MtThreadsLimit. Once per second, the master thread checks the current values of MtThreadsLimit, MtActiveThreads, MtPercentCpuActiveLimit and percentCpuActive, to determine how to set the number of threads running (MtActiveThreads). The master thread will shut down or activate slave threads as needed, up to the MtThreadsLimit ceiling, to keep the CPU usage under the MtPercentCpuActiveLimit.

Since the multi-threaded scans manage load by controlling the number of threads, the most straightforward tuning involves specifying a maximum CPU usage limit. For example, for a scan operation that is being performed on a system that is running a live application, the maximum threads can be set to a high value and the maximum CPU usage can be used to control the CPU resources used. If the scan pushes the CPU

utilization above the specified percentage, the main thread automatically deactivates slave threads until the load is again below the specified limit. If necessary, if the CPU is heavily utilized for other processes, the scan operation will pause until CPU usage is again below the limit.

The multithreaded scans must be performed by Gems running locally to the Stone. When executed from a Gem that is running on a machine remote from the Stone, the scan will be performed by one thread, regardless of the number of threads requested. The new method **`System class >> sessionIsOnStoneHost`** allows you to confirm whether the gem is local.

Existing interface methods perform the multi-threaded operations with a moderate but not minimal default. The default is a maximum of 2 threads, and 90% CPU percentage. Functionally equivalent "fast" methods have been added to perform the operations with the most aggressive settings. For these methods, MtThreadLimit is estimated based on the number of CPU cores and the number of current sessions, and MtPercentCpuActiveLimit is set to 95%.

Memory impact

Multi-threaded operations may require considerable heap memory. This memory requirement is **not** part of temporary object cache memory; you can configure your GEM_TEMPOBJ_CACHE_SIZE per other application gem requirements, or even configure the sessions performing repository scan operations with a very small temporary object cache size.

The amount of memory space that is needed depends primarily upon the current oopHighWater value, the number of threads, and the page buffer size. Most operations default to use a pageBufSize of 8, with the exception of markForCollection and findDisconnectedObjects operations, which use a default pageBufSize of 128.

- The overhead associated with the oopHighWater value can be computed as:

```
(stnOopHighWater + 10M) / 2
```

- The memory cost per thread is:

```
50K + (180K * pageBufSize)
```

For example, a system with an oopHighWater mark of 500M running eight threads with a page buffer size of 128 would require about 440 MB of free memory to start up.

Persistent vs. transient instances

The multi-threaded scan methods locate only persistent objects in the repository. Objects that exist only in memory for the current session are not found or returned.

The exception to this is **`Class >> allInstances`**. For consistency with previous releases, this methods collects both persistent and in-memory instances and returns a result containing both.

The following methods, which operate on objects in memory only and do not use multi-threaded scan, have been added:

```
Repository >> listInstancesInMemory: anArray

Repository >> listReferencesInMemory:anArray

Object >> findReferencesInMemory

Object >> findNonStubbedReferencesInMemory
```

Functionality using multi-threaded scanning operations

The following Repository methods all use multi-threaded scan code.

The existing interface methods now perform multi-threaded scanning with a default limit of 2 threads and 90% of CPU.

New "fast" methods use aggressive scanning to complete as quickly as possible, using most or all system resources.

Additional new methods are also provided, with arguments that allow you to explicitly specify the thread and CPU limits.

For more details on these methods, see method comments in the image.

Object Audit and Repair

Object audit has been redesigned and is substantially different in version 3.0. In addition to being much faster, the requirements are relaxed and the output is limited to reporting the audit results, rather than information about the contents of the repository.

Object audit no longer needs to run in single-user mode, and does not need to do a mark-sweep/reclaim cycle prior to running. Object audit and repair no longer require GarbageCollection privilege, but do require SystemControl privilege. Instead of details on the contents, the following line is reported:

```
Object Audit: Audit successfully completed; no errors were
detected.
```

To get the details that were previously reported by objectAudit, you can use methods in the class GsObjectInventory (see page 61).

The following are the new objectAudit protocols:

objectAudit

repair

fastObjectAudit

objectAuditWithMaxThreads: *maxThreads* **percentCpuActiveLimit:** *aPercent*

The methods **auditWithLimit:**, **repairWithLimit:**, and **quickObjectAuditWithLevel:** are deprecated. These methods invoke the new objectAudit protocol; code that performed the objectAudit in previous releases is no longer present.

MarkForCollection (MFC)

markForCollection

fastMarkForCollection

markForCollectionWithMaxThreads: *maxThreads* **waitForLock:** *waitTimeSeconds* **percentCpuActiveLimit:** *aPercent*

FindDisconnectedObject (FDC)

With the improved performance of MFC multi-threaded scans, it is no longer recommended to run FDC. FDC may be deprecated in a future release.

For methods that have a **pageBufferSize:** keyword, the argument to this must now be a power of 2. This value impacts the memory availability requirements for the session. Methods without a specific argument default to use a page buffer size of 128.

findDisconnectedObjectsAndWriteToFile: *aFileNameString* **pageBufferSize:** *pageBufSize* **saveToRepository:** *saveToRep*
This method defaults to use an aggressive setting that may consume all system resources, in order to complete as quickly as possible.

basicFindDisconnectedObjectsAndWriteToFile: *aFileNameString* **pageBufferSize:** *pageBufSize* **saveToRepository:** *saveToRep* **withMaxThreads:** *maxThreads* **maxCpuUsage:** *aPercentage*

findDisconnectedObjectsAndWriteToFile: *aFileNameString* **withMaxThreads:** *maxThreads* **maxCpuUsage:** *aPercentage*

Note that the old "fast" FDC functionality has been removed; see "Fast findDisconnectedObjects removed" on page 63.

Count Instances

countInstances: *anArray*

fastCountInstances: *anArray*

countInstances: *anArray* **withMaxThreads:** *maxThreads* **maxCpuUsage:** *aPercentage*

List Instances

listInstances: *anArray*

listInstances: *anArray* **limit:** *aSmallInt*

listInstances: *anArray* **toDirectory:** *aString*

listInstancesInPageOrder: *anArrayOfClasses* **toFile:** *aString*

listInstancesToHiddenSet: *aClass*

fastListInstances: *anArray*

fastListInstances: *anArray* **limit:** *aSmallInt*

fastListInstances: *anArray* **toDirectory:** *aString*

fastListInstancesInPageOrder: *anArrayOfClasses* **toFile:** *aString*

fastListInstancesToHiddenSet: *aClass*

listInstances: *anArray* **limit:** *aSmallInt* **toDirectory:** *aStringOrNil* **withMaxThreads:** *maxThreads* **maxCpuUsage:** *aPercentage*

listInstancesToHiddenSet: *aClass* **withMaxThreads:** *maxThreads* **maxCpuUsage:** *aPercentage*

listInstancesInPageOrder: *anArrayOfClasses* **toFile:** *aString* **withMaxThreads:** *maxThreads* **maxCpuUsage:** *aPercentage*

The method **Class >> allInstances** invokes multi-threaded scan code using the default limits. This method also invokes **listInstancesInMemory:** to find non-persistent instances of the class.

List References

listReferences: *anArray*

listReferences: *anArray* **withLimit:** *aSmallInt*

fastListReferences: *anArray*

listReferences: *anArray* **withLimit:** *aSmallInt* **withMaxThreads:** *maxThreads* **maxCpuUsage:** *aPercentage*

The methods **Object >> findReferences**, **Object >> findReferencesWithLimit:**, and **Object >> findAllReferences** invoke multi-threaded scan code using the default limits.

List Objects in ObjectSecurityPolicies (Segments)

listObjectsInObjectSecurityPolicies: *anArray*

listObjectsInObjectSecurityPolicies: *anArray* **toDirectory:** *aString*

listObjectsInObjectSecurityPolicies: *anArray* **withLimit:** *aSmallInt*

listObjectsInObjectSecurityPolicyToHiddenSet: *anObjectSecurityPolicyId*

fastListObjectsInObjectSecurityPolicies: *anArray*

fastListObjectsInObjectSecurityPolicies: *anArray* **toDirectory:** *aString*

fastListObjectsInObjectSecurityPolicies: *anArray* **withLimit:** *aSmallInt*

fastListObjectsInObjectSecurityPolicyToHiddenSet: *anObjectSecurityPolicyId*

Note that the existing methods **listObjectInSegments:**, **listObjectsInSegments:toDirectory:**, **listObjectsInSegments:withLimit:**, and **listObjectsInSegmentToHiddenSet:** are deprecated, and are replaced by equivalent methods listed above.

Cache warming using configuration parameters

Cache warming on stone startup using the new configuration parameters STN_CACHE_WARMER and STN_CACHE_WARMER_SESSIONS uses multi-threaded

scanning algorithms. The sessions specified by STN_CACHE_WARMER_SESSIONS are slave threads, rather than regular GemStone sessions. See the section "Cache Warming Changes" on page 89.

GsObjectInventory

The class GsObjectInventory includes protocol to scan the repository and report details of the number of classes and instances. The existing interface methods now perform multi-threaded scanning with a default limit of 2 threads and 90% of CPU. New "fast" methods use aggressive scanning to complete as quickly as possible, using most or all system resources.

The following methods perform an abort; you may not have uncommitted changes in your repository prior to executing these methods. These methods run in transaction, but the implementation does not require a commit record, so they avoid causing a commit record backlog.

GsObjectInventory class >> profileRepository

GsObjectInventory class >> fastProfileRepository

GsObjectInventory class >>
profileRepositoryAndSkipHiddenClasses

GsObjectInventory class >>
fastProfileRepositoryAndSkipHiddenClasses

GsObjectInventory class >> profileGarbageFromFile: *aFilename*
includeHiddenObjects: *aBoolean*

GsObjectInventory class >> fastProfileGarbageFromFile: *aFilename*
includeHiddenObjects: *aBoolean*

GsObjectInventory class >> profileObjectsInHiddenSet: *anInt*
showHiddenClasses: *aBool*

GsObjectInventory class >> fastProfileObjectsInHiddenSet: *anInt*
showHiddenClasses: *aBoolean*

The following methods operate on a specific collection of objects which may be committed or uncommitted. If the session is in transaction, no abort is performed, and you may profile a collection of temporary objects. If you are not in transaction, the method begins a transaction and aborts at completion.

GsObjectInventory class >> profileObjectsIn: *aCollection*

GsObjectInventory class >> fastProfileObjectsIn: *aCollection*

Find Objects larger than

System >> findObjectsLargerThan: *aSize* **limit:** *aLimit*
This method returns an Array of all objects larger than the given size, returning up to *aLimit* objects. This method performs the scan using two thread, with a 90% CPU usage limit.

System >> fastFindObjectsLargerThan: *aSize* **limit:** *aLimit*
Similar to **System >> findObjectsLargerThan:limit:**, but uses an aggressive number of threads and up to 95% of CPU.

New tuning methods

GemStone/S 64 Bit v3.0 provides new public instance methods on Repository that you can use to monitor and tune your multi-threaded application. This allows you to modify the impact on your system of a multi-threaded scan, while the operation is underway. For example, you can set your scan to use all system resources during off hours, but reduce impact during the business day.

These operations require #SystemControl and/or #SessionAccess privileges, and must be executed from a Gem running on the same host that the Gem performing the multi-threaded scan is running on.

To perform simple tuning of any multi-threaded scans currently in progress, you can invoke one of the following Repository methods:

`setMultiThreadedScanMaxAggressive`
`setMultiThreadedScanMediumAggressive`
`setMultiThreadedScanMinAggressive`
Sets the thread and CPU usage limits per common possible usages.

You can temporarily pause a scan entirely using the following method:

`pauseMultiThreadedScan`
Pauses the operation by setting the number of threads to use to 0. To resume, send any method that resets the number of threads used by the scan, such as any of the **`setMultiThreadedScan*`** methods.

To specifically tune, you can set both maximum thread and CPU usage limits using:

`setMultiThreadedScanThreads:` *numThreads* **`maxCpu:`** *aPercent*
Modifies the number of threads active for a multi-threaded scan, and the maximum CPU usage. *aPercent* may be an Integer representing a percentage between 0 and 100; or it may be nil, meaning do not change the existing value.

The following methods are also available to access and update cache statistics:

`getMultiThreadedScanMaxThreads`

`mtMaxThreads:` *sessionId*

`mtPercentCpuActiveLimit:` *sessionId*

`mtPercentCpuActiveLimit:` *sessionId* **`setValue:`** *newVal*

`mtThreadsLimit:` *sessionId*

`mtThreadsLimit:` *sessionId* **`setValue:`** *newVal*

Removed methods

In addition to the methods listed above, the following instance methods on Repository have been removed in version 3.0:

auditDependencyMap, repairDependencyMap
In GemStone/S 64 Bit 3.0, the dependencyList is only audited as part of objectAudit.

readObjectTableAndDataPagesForGem:of:useSharedCache:
cacheFullLimit:

readObjectTableForGem:of:useSharedCache:cacheFullLimit:

readObjectTableForGem:of:useSharedCache:loadDataPagesOpCode:
cacheFullLimit:

readPageRangeForGem:of:pageBufSize:

Fast findDisconnectedObjects removed

The old "fast" FDC functionality, which used cache warmers to pre-load objects for improved performance, is obsolete in this release. It is replaced by the multi-threaded operations, which are much faster.

The following methods have been removed:

basicFindDisconnectedObjectsAndWriteToFile:pageBufferSize:
saveToRepository:numCacheWarmers:

findDisconnectedObjectsAndWriteToFile:pageBufferSize:
saveToRepository:numCacheWarmers:

findDisconnectedObjectsAndWriteToFileUsingCacheWarmers:

shrinkExtents deprecated

Repository >> shrinkExtents, which we have recommended to avoid using for some time, is deprecated in this release.

To safely and reliably reduce the size of the repository to its minimum size, take a programmatic backup and restore this into clean extents, per the instructions in the *System Administration Guide*.

12. Collection and String Changes

Array and ByteArray literals – change in semantics for #[ ]

Previous releases used the syntax #(a b c) as an Array literal, and #[a, b, c] as an Array constructor (also known as Array builder). An Array literal is evaluated at compile time, while an Array constructor is evaluated at runtime and can include variables or statements. In v2.2.4, the syntax { a . b . c } was added as an alternate syntax for Array constructor, for compatibility with Squeak.

Version 3.0 replaces the use of square brackets for ByteArray literals, which could not previously be defined in GemStone/S 64. ByteArray literals are defined using syntax of the form #[0 1 254 255]. This is compatible with the ByteArray literal syntax of other Smalltalk dialects.

All Array constructors should now use the syntax { a . b . c }. This differs from the previous Array constructor syntax:

- Curly braces rather than square brackets
- Elements separated by periods rather than commas
- No initial # required

This may have significant impact on existing code. To file in code created in previous versions of GemStone/S 64, you can configure your system to translate the Array constructor syntax during compilation using a new runtime-only configuration parameter, #GemConvertArrayBuilder. This process is used during upgrade, which requires application code file-in to recompile methods. It may also be set temporarily to load code at other times.

The new configuration parameter #GemConvertArrayBuilder is runtime-only, and must be set using:

```
System gemConfigurationAt: #GemConvertArrayBuilder put: true
```

When this is set, the source strings of methods and executions are passed through a pre-processor to convert legacy-style Array constructor syntax to the updated Array constructor syntax. With #GemConvertArrayBuilder set to true, compilation of methods that include the old square bracket Array constructor syntax will succeed, and the resulting compiled method will have updated correct source using curly braces, as well as the correct compiled code. This allows code following the legacy syntax to be filed in or re-compiled correctly.

- If #GemConvertArrayBuilder was true during a file-in, the source strings of the compiled methods reflect the changes made by the preprocessor. You will need to file out the compiled methods if you want the modified source code to be present in an external source control system.

Since this is a runtime-only configuration, it is set to false at each login.

To convert code stored in Topaz methods, the following GsNMethod class methods are provided:

c**`convertArrayBuildersInString:`** *aString*
Converts any ArrayBuilder in *aString* to CurlyArrayBuilder syntax per $GEMSTONE/doc/bnf.txt. Returns aString if no ArrayBuilder found, otherwise returns a modified copy of aString.

convertArrayBuildersInTopazScript: *aFileName*
Converts any ArrayBuilder in source strings within specified file to CurlyArrayBuilder syntax per $GEMSTONE/doc/bnf.txt. The file is overwritten in place. Return true if file changed, false otherwise.

convertArrayBuildersIn: *directoryName* **list:** *listName*
Given *directoryName* and a file named *listName* within that directory, assuming that *listName* has one fileName per line, converts each file in the list. Within the list, leading/trailing whitespace on each line is ignored, and lines starting with # are ignored.

For updated BNF, see the *GemStone/S 64 Bit Programming Guide* for version 3.0.

Multi-byte string instances now stored in-memory in CPU byte order

DoubleByteString, DoubleByteSymbol, and QuadByteString (along with Float and SmallFloat) have been reimplemented for efficiency in byte storage. In-memory instances are in CPU native byte order, regardless of the byte order of the repository, by swizzling when they are loaded into memory. The old classes remain and are renamed ObsDoubleByteString, ObsDoubleByteSymbol, and ObsQuadByteString.

In upgraded repositories containing instances of these classes, the instances are auto-mutated into instances of the new class when they are loaded into memory. The new method **convert** can be used to commit the in-memory state of instances of any of these classes to make the mutation persistent.

Full support for QuadByteString/Symbol

QuadByteStrings and QuadByteSymbols no longer have usage limitations, and can be compiled and otherwise handled the same as DoubleByteString/Symbol and String/Symbol.

Older non-standard String and Character classes deprecated

With the full support for QuadByteString, and extended character set support, classes that historically were used to support complex characters and strings are no longer necessary. The following classes, and methods specific to these classes, are deprecated in v3.0:

EUCString
EUCSymbol
JapaneseString
JISCharacter
JISString
InvariantEUCString

Collection >> accompaniedBy:do: required

There have been a number of optimizations and changes to Collection methods. One of these changes is that several methods now rely on a new method, **accompaniedBy:do:**, to handle certain cases. If you have customized Collection classes, ensure that your class implements or responds correctly to **accompaniedBy:do:**.

copyFrom:to:into:startingAt: replaced by replaceFrom:to:with:startingAt:

The method **copyFrom:to:into:startingAt:** is deprecated in v3.0. It is replaced by method **replaceFrom:to:with:startingAt:**, which performs the equivalent function with slightly different argument ordering and semantics. Implementation of **copyFrom:to:into:startingAt:** now invoke **replaceFrom:to:with:startingAt:**.

If you have reimplemented **copyFrom:to:into:startingAt:** in subclasses, then you should review and update your code; this method is used internally in the implementation of other functionality.

The previous implementations of **copyFrom:to:into:startingAt:** placed limitations on the types of collections that could be used as arguments. **replaceFrom:to:with:startingAt:**, and therefore **copyFrom:to:into:startingAt:** in v3.0, does not have these restrictions.

KeyValueDictionary subclass optimizations

The classes IdentityKeyValueDictionary, StringKeyValueDictionary, and IntegerKeyValueDictionary have updated implementations that do not use **compareKey:with:** and **hashFunction:**. This should not impact your application unless you have customized subclasses.

In addition, the classes StringKeyValueDictionary and IntegerKeyValueDictionary use a new class EqualityCollisionBucket as their collisionBucket.

SortedCollection additional methods

Persistent instances of SortedCollection (or a custom subclass) include a sort block. Due tot the reimplementation of ExecutableBlock (see page 17), all such blocks must be recreated. This is done by the postconv step of conversion.

To support this conversion, a number of methods have been added to SortedCollection. These should not affect non-conversion use.

atAllPut: change

Previously, **SortedCollection >> atAllPut:** returned the receiver, rather than the argument. This was inconsistent with other GemStone implementations of **atAllPut:**. **SortedCollection >> atAllPut:** has been changed to return the argument.

ReadStream >> nextOrNil

The following method has been added on ReadStream:

ReadStream >> nextOrNil
: Returns the next object that the receiver can access for reading. Returns nil if an attempt is made to read beyond the end of the stream. For use with Streams whose collections do not contain nil.

CharacterCollection addAll: and addLast:

Previously, these method return the receiver rather than the argument, unlike most **add:** protocol. Now, they return the argument. Note that this does not affect String or MultiByteString (DoubleByteString and QuadByteString), which override the behavior.

Indexing Changes

Improvements to auditIndexes

auditIndexes has had performance improvements for most audit situations.

In addition, progress counts and statistics have been added to allow monitoring of the progress of auditIndexes.

A new gem cache statistic has been added:

IndexProgressCount (Gem)
Used to monitor the status of certain index operations:
0 - Inactive
1 - Identity index audit in progress.
2 - Equality index audit - auditing root terms.
3 - Equality index audit - auditing NSC counts.
4 - Equality index audit - auditing btree counts.
5 - IndexManager>>removeAllIndexes in progress.

For some of the status reported by IndexProgressCount, the ProgressCount statistic is also updated to indicate the progress of some of these operations. In these cases, it starts at the number of path terms to be checked and is decremented each time the audit of a path term has completed, or incremented when the audit switches to audit a different index.

The following methods have been added to support the update of these cache statistics by the audit operations performed in Smalltalk:

```
UnorderedCollection class >>
    statValueForAuditingIdentityIndexes
UnorderedCollection class >> statValueForAuditingRootTerms
UnorderedCollection class >> statValueForAuditingNscCounts
UnorderedCollection class >> statValueForAuditingBtreeCounts
UnorderedCollection class >> statValueForRemovingAllIndexes
System class >> incrementProgressCountBy: aSmallInt
System class >> decrementProgressCountBy: aSmallInt
System class >> setProgressCountTo: aSmallInt
System class >> incrementIndexProgressCountBy: aSmallInt
System class >> decrementIndexProgressCountBy: aSmallInt
System class >> setIndexProgressCountTo: aSmallInt
```

Default values for dirtyObjectCommitThreshold and percentTempObjSpaceCommitThreshold

The default for IndexManager's percentTempObjSpaceCommitThreshold has been changed from 75 to 60.

Previously, the default for the IndexManager dirtyObjectCommitThreshold was 20000, which resulted in frequent commits (when autoCommit is enabled). The default has been changed, and is now SmallInteger maximumValue, disabling dirtyObjectCommitThreshold. Now, by default, autoCommit will commit when the percentTempObjSpaceCommitThreshold limit is reached.

Methods added to CharacterCollection or String

`CharacterCollection >> containsDigit`
Answers true if the receiver contains at least one digit, otherwise answers false. Added for password management.

`CharacterCollection >> containsLowercase`
Answers true if the receiver contains at least one lowercase character, otherwise answers false. Added for password management.

`CharacterCollection >> containsPasswordSymbol`
Answers true if the receiver contains at least non-alphanumeric character, otherwise answers false. Added for password management.

`CharacterCollection >> containsUppercase`
Answers true if the receiver contains at least one uppercase character, otherwise answers false. Added for password management.

`CharacterCollection >> copyReplaceChar:` *aCharacter* **`with:`** *secondCharacter*
Returns a copy of the receiver with all occurrences of *aCharacter* replaced with *secondCharacter*.

`CharacterCollection >> findLastSubString:` *subStr* **`startingAt:`** *startIndex*
startIndex should be >= 1 and <= self size. Search is backwards through the receiver. Returns 0 if no match found.

`CharacterCollection >> hasLdapWildcardPattern`
Answers true if the receiver contains one and only one '%' character, and the character after the '%' is 's'.

`String >> indexOfByte:` *anInt* **`startingAt:`** *startIndex*
Returns the index of the first occurrence of (**`Character withValue:`** *anInt*) in the receiver, not preceding *startIndex*. If the receiver does not contain the specified Character, returns zero. The search is case-sensitive.

`String >> indexOfLastByte:` *anInt* **`startingAt:`** *startIndex*
Returns the index of the last occurrence of (**`Character withValue:`** *anInt*) in the receiver, starting from *startIndex* and searching backwards. If the receiver does not contain the specified Character, returns zero. The search is case-sensitive.

invalidReferencesAudit and invalidReferencesAuditWithRepair: removed

These methods fixed conditions resulting from a bug that was fixed long ago, and are no longer necessary.

Interval displayed as code rather than iteratively

Previously, an Interval such as (1 to: 7 by: 3) displayed by listing the calculated contents (1, 4, 7), with no indication of the arguments defining the Interval. Now, Interval display includes the size of the collection and the arguments used to create it, for this example:

```
anInterval of size 3 (1 to: 7 by: 3)
```

#inject:keysAndValuesInto: and #inject:valuesInto: renamed

The methods #inject:keysAndValuesInto: and #inject:valuesInto: were added in v2.2 to optimize certain types of iterations to avoid complex blocks and improve performance. These method names, which imply inject: semantics, were misleading; the methods have been renamed.

Old name	New Name
inject:keysAndValuesInto:	accompaniedBy:keysAndValuesDo:
inject:valuesInto:	accompaniedBy:do:

IdentityBag class >> elementKind removed

The method **`IdentityBag class >> elementKind`** has been removed. This method was made obsolete many years ago. It may be replaced with **`Behavior >> varyingConstraint`**, which it previously called, but note that constraints are deprecated in v3.0.

PositionableStream changes to allow ANSI and Legacy to coexist

GemStone's implementation of PositionableStream is not ANSI compliant, and includes instance variables that are not portable to other Smalltalk dialects.

In order to support both instances of PositionableStream in upgraded repositories, and new development that should be portable, GemStone now includes several hierarchies of PositionableStream. These hierarchies all exist in the image, but only a subset are visible at any time.

There are four separate sets of these classes,. The following two are always visible:

```
PositionableStreamLegacy
   ReadStreamLegacy
   WriteStreamLegacy
```

These are the same classes as PositionableStream in previous versions, and instances of subclasses of PositonableStream in upgraded repositories will be instance of these classes.

```
PositionableStreamPortable
   ReadStreamPortable
   WriteStreamPortable
      ReadWriteStreamPortable
         FileStreamPortable
```

This is a new, ANSI compatible and portable implementation of PositionableStream (including additional subclasses, mentioned on page 87). Much of the interface is the same, but note the position is 0-based.

`PositionableStream` (with Legacy definition and methods)

```
   ReadStream
   WriteStream
```

These are distinct class instances, with behavior identical to the equivalent PositionableStreamLegacy. These classes will be visible in upgraded repositories, but not visible in new 3.0 repositories.

```
PositionableStream (with Portable definition and methods)
   ReadStream
   WriteStream
      ReadWriteStream
         FileStream
```

These are distinct class instances, with behavior identical to the equivalent PositionableStreamPortable. These classes will be visible in new 3.0 repositories, but invisible by default in upgraded repositories.

The portable classes are stored in Globals under #GemStone_Portable_Streams, and can be installed using:

```
Stream class >> installPortableStreamImplementation.
```

The legacy classes are also stored in Globals under #GemStone_Legacy_Streams, and can be installed using:

```
Stream class >> installLegacyStreamImplementation
```

To check what is currently installed, use the following messages, which can be sent to any PositionableStream class or subclass:

```
isLegacyStreamImplementation
isPortableStreamImplementation
```

GBS replication of PositionableStreams

GBS version 7.4 does not support replication of **`PositionableStreamPortable`** and its equivalent **`PositionableStream`**.

Repositories that were upgraded from 2.x will by default have PositionableStreamLegacy and its equivalent **`PositionableStream`** installed, and will be able to replicate PositionableStreams normally, as in previous releases. Do not install the portable classes.

If new 3.0 repositories will be used with GBS, you should install the Legacy classes by invoking **`Stream installLegacyStreamImplementation`**.

PositionableStreamLegacy ANSI position

Previous versions of GemStone have defined **`PositionableStream >> position`** and **`PositionableStream >> position:`** as a 1-based offset, while the ANSI standard requires a 0-based offset. In version 3.0, these methods by default now follow the ANSI protocol and use a 0-based offset.

Note that is applies to new instances of **`PositionableStreamLegacy`** and **`PositionableStream`** (with Legacy behavior). **`PositionableStreamPortable`** and its equivalent **`PositionableStream`** always used 0-based position offset.

Version 2.4 and later allowed you to install ANSI-compatible **`position`** and **`position:`** methods; you could also install legacy or warning implementations of these methods. This functionality remains, and upgraded repositories are not affected and will continue to use either Legacy or ANSI, according to options installed in the v2.4.x repository.

13. Changes for Developers

ProfMonitor enhancement

ProfMonitor now supports sample intervals less than the resolution of an OS timer interrupt. In previous releases, the minimum sampling interval was 10 milliseconds. In version 3.0, the sample interval can be specified in nanoseconds, in which case a high resolution clock is checked at each point where interrupts would be checked. The minimum interval is now 1000 nanoseconds.

While you can monitor with much smaller intervals, the code to be profiled must take at least 10 milliseconds to execute, and the total CPU time reported will be in increments of 10 ms.

Note that 1 millisecond (ms) = 1000 microseconds (us) = 1,000,000 nanoseconds (ns).

Intervals less than 10 milliseconds use high resolution sampling, while intervals >= 10 milliseconds use the previously available timer interrupt implementation.

The default interval when using methods such as **`spyOn:`** and **`monitorBlock:`** is now 1 millisecond. To get the interval behavior in previous releases, use **`monitorBlock:downTo:interval:`** with an interval argument of 10.

New version of the profiling methods have been added with the additional **`intervalNs:`** keyword:

`monitorBlock:` *aBlock* **`downTo:`** *hits* **`intervalNs:`** *nanosecondsPerSample*

`monitorBlock:` *aBlock* **`intervalNs:`** *nanosecondsPerSample*

When using high resolution sampling, code executes 20% to 30% slower, and profiling may use much more memory. To determine an appropriate interval to use to avoid out of memory issues, a new method is provided.

`ProfMonitor Class >> computeInterval:` *unprofiledCpuTimeSeconds*
Returns an estimated profiling interval in nanoseconds for an execution taking the specified amount of CPU time, to yield approximately 100000 samples. For 100000 samples, GEM_TEMPOBJ_CACHE_SIZE should be set to at least 300MB to avoid OutOfMemory errors during analysis phase.

Coding style changes

For performance, calls to **`perform:with:`**, **`perform:with:with:`**, and **`perform:with:with:with:`** should be replaced by calls to **`with:perform:env:`**, **`with:with:perform:env:`**, **`with:with:with:perform:env:`**, using 0 as an argument to the **`env:`** keyword.

The call **`with:with:with:with:perform:env:`** (allowing one further argument) is now also available.

New optimized selector methods have been added and may be used for performance. The complete list of optimized selectors is on page 21. For example:

`Object >> ifNil:` *nilBlock*
`Object >> ifNotNil:` *notNilBlock*

Object >> ifNotNil: *notNilBlock* **ifNil:** *nilBlock*
Object >> ifNil: *nilBlock* **ifNotNil:** *notNilBlock*

ExecBlock >> untilTrue: *aBlock*
ExecBlock >> untilFalse: *aBlock*

The block argument *notNilBlock* can have either zero or one argument.

Deprecation mechanism

A deprecation mechanism has been added in version 3.0. Methods may be deprecated by including within the source code a call to one of the following methods:

Object >> deprecated

Object >> deprecated: *aString*
aString specifies a message that should include the version in which the deprecation was added, and alternative code to use.

Object >> deprecatedNotification: *aString*

Methods that are deprecated are supported until the next major release, for at least a year after the date of the release in which is it deprecated, but may be removed from the image after that point.

By default, the **deprecated** and **deprecated:** methods do nothing. To turn on notification of deprecated methods, uncomment the code in the **deprecated:** method.

Many deprecated or obsolete GemStone server methods now invoke **deprecated:**; however, there may be deprecated or obsolete methods that do not use this mechanism.

For a list of methods that use the deprecated mechanism, execute:

```
ClassOrganizer new sendersOfReport: #deprecated:
```

Support for Monticello

GemStone/S 64 Bit version 3.0 introduces the ability to load .mcz files for Monticello, a distributed version control source code storage system. Monticello is used extensively in the Seaside product; limited features are now available in GemStone/S 64 Bit.

This feature is now used by GBS to load server code specific to the GBS product into the server.

The following methods have been added:

GSPackageLibrary class >> loadMczFile: *mczFilename*
fromRepositoryPath: *repositoryPath*

Loads the given .mcz file *mczFilename* into the current user's UserGlobals symbol dictionary from the repository located in the directory *repositoryPath* on the server.

GSPackageLibrary class >> loadLastVersionOf: *packageName*
fromRepositoryPath: *repositoryPath*

Loads the latest version of the package *packageName* into the current user's UserGlobals symbol dictionary from the repository located in the directory *repositoryPath* on the server.

NOTE

The classes that are used to load Monticello code are in a symbol dictionary that is not in the current user's symbolList, and thus are not visible in the image.

For more information about Monticello – for example, how to save code in Monticello – refer to the online reference Pharo By Example (http://www.pharobyexample.org).

Methods added to ClassOrganizer

The following new instance methods are available in ClassOrganizer. Note that some of these methods have identical variants with abbreviated names, for convenience when typing in Topaz.

`implementorsOfReport:` *aSelector*, **`impls:`** *aSelector*
Returns a String describing the methods that are senders of the selector *aSelector*.

`implsC:` *aSelector*
A variant of **`implementorsOfReport:`** sorted by method categories.

`methodCategories`
Returns a String containing a report of all the method categories.

`methodsInCategory:` *aString*, **`methsInCat:`** *aString*
Returns a String containing a report of all methods in the category with name *aString*.

`sends:` *aSelector*
A variant of **`sendersOfReport:`**. Provided for convenience when typing in Topaz.

`sendsC:` *aSelector*
Senders report sorted by method categories.

`sentButNotImplemented`, **`sendsNotImpl`**
Returns an Array of selectors that are sent but not implemented.

`sentButNotImplementedReport`
Returns a report of selectors sent but not implemented.

`stringsC:` *aString*
A variant of **`substringReport:`** sorted by method categories.

`stringsIc:` *aString*
A case-insensitive variant of **`substringReport:`**, easier to type.

`stringsIcC:` *aString*
A case-insensitive variant of **`substringReport:`** sorted by method categories.

`subclassesReport:` *aClass*
Returns a report of all subclasses of the class *aClass*.

`substringReport:` *aString*, **`strings:`** *aString*
Returns a String describing the methods whose source contains *aString*.

New Behavior methods

Methods have been added to Behavior to take advantage of implementation changes that are designed to support code management.

`authorInitials`

`changeStamp`
Answers a string to be pasted into source code to mark who changed it and when.

`commentStamp`
Returns the comment timestamp. Both a Class and its metaclass share a single comment timeStamp.

`commentStamp:` *aStamp*
Sets the receiver's comment timestamp to *aStamp*. Both a Class and its metaclass share a single comment stamp.

`setStamp:` *aStamp* **`forMethod:`** *selector*
Sets the timestamp for the given method. This code is shared by Classes and metaclasses, parameterized by methodStampDictName.

`stampForMethod:` *selector*
Gets the timestamp for the given method.

14. Changes in Cache Statistics Programmatic Interface

In addition to the changes documented below, there are changes in cache statistics names and other details; see "Cache Statistic Changes" on page 96.

There are minor changes in the reporting of processName to be more consistent.

New API to retrieve statistics

Previously, each type of process that recorded statistics (stone, gem, spcmon, and pgsvr) used the same set of statistics descriptions and slots, although the actual information recorded by each process was not identical. In some cases it was ambiguous which statistics applied in which way to which process.

This older interface is still available, but deprecated. A new API has been added that returns a different set of statistics descriptions and Array of statistics values for each process type, as well as general methods to return information based on slot, pid, or process type.

Statistics Descriptions

`System class >> cacheStatisticsDescriptionForType:` *aProcessKind*
Returns an Array of Strings describing the result of the method #'cacheStatisticsAt:'. The argument <aProcessKind> is the StatType (as returned as element #4 by #'cacheStatisticsAt:') and must be one of the following:

1= Shared Page Cache Monitor
2 = Stone
4 = Page Server
8 = Gem (including Topaz, GBS, and other GCI applications).

The result includes only those appropriate to the specific executable (common and unique). Note that Statmonitor will capture additional statistics (e.g., on itself and the system) that are not actually stored in shared memory.

`System class >> cacheStatisticsDescriptionAt:` *aSlot*
Returns an array of strings which describe the cache statistics for the kind using the cache slot with index anInt. Returns an empty array if the slot is not in use.

`System class >> cacheStatisticsDescriptionForStone`
Returns an Array of Strings describing cache statistics applicable to the stone.

`System class >> cacheStatisticsDescriptionForGem`
Returns an Array of Strings describing cache statistics applicable to a gem process.

`System class >> cacheStatisticsDescriptionForMonitor`
Returns an Array of Strings describing cache statistics applicable to the shared page cache monitor.

`System class >> cacheStatisticsDescriptionForPageServer`
Returns an Array of Strings describing cache statistics applicable to a page server process.

Statistics for all processes

`System class >> cacheStatisticsForAllSlots`
Return an Array with one entry for each process attached to the shared cache. Each

element in the result contains the cache statistics for that slot. The last element in each array is the slot ID of the slot. The subarrays will be different sizes depending on what type of process is using the slot.

System class >> cacheStatisticsForAllSlotsShort
Return an Array with one entry for each process attached to the shared cache. Each element in the result is an array containing 5 elements. The first 4 elements are the same first 4 elements returned when fetching cache statistics for a session, slot, or process. The fifth element is the process slot for the process. Each sub-array of 5 elements contain the following elements:

1 - ProcessName (String)
2 - ProcessId (SmallInteger)
3 - SessionId (SmallInteger)
4 - StatType (SmallInteger)
5 - Cache Slot (SmallInteger)

All statistics for a process

System class >> cacheStatisticsAt: *aProcessSlot*
Returns an array of statistics appropriate to the specific executable (common + unique). The argument *aProcessSlot* is the same as described in #'cacheStatistics:' method. The contents are described by the result of sending element 4 of the result to the #cacheStatisticsDescriptionFor: method.

System class >> cacheStatisticsProcessId: *aPid*
Search the cache for a process id matching *aPid*. If found, return the statistics appropriate for that process. Returns nil if the process was not found.

System class >> stoneCacheStatistics
Return the cache statistics for the stone if this session is located on the same host as the stone process. Only cache statistics applicable to the Stone process are returned. The description of these statistics can be determined by evaluating: 'System cacheStatisticsDescriptionForStone'

System class >> gemCacheStatisticsForSessionId: *aSessionId*
Same as the cacheStatisticsAt: method except the argument is the session ID of a session currently connected to the shared memory cache. In systems that use multiple shared memory caches, the session must exist on the same cache as the session invoking this method. Only cache statistics applicable to a Gem process are returned. The description of these statistics can be determined by evaluating: 'System cacheStatisticsDescriptionForGem'

System class >> cacheStatisticsForProcessWithCacheName: *aString*
Search the cache for a process with cache name matching *aString*., and return statistics for that process, or nil if no such process is found. This method is useful in conjunction with naming gems using **System cacheName:**.

System class >> sharedPageCacheMonitorCacheStatistics
Return the cache statistics for the SPC monitor process. Only cache statistics applicable to the SPC monitor process are returned. The description of these statistics can be determined by evaluating: 'System cacheStatisticsDescriptionForMonitor'

System class >> cacheStatsForPageServerWithSessionId: *anInt*
Return the cache statistics for the page server with the given session ID. Page servers assume the same session ID as their client gem. Return -1 if the page server with the given session ID was not found.

Specific statistic by name for a process

System class >> stoneCacheStatisticWithName: *aString*
Return the value of the cache stat with the given name, which must match an element of the Array returned by the #cacheStatisticsDescription method applicable to the stone process. The UserTime and SysTime statistics cannot be accessed using this method. Returns nil if a statistic matching *aString* was not found for the stone.

System class >> primaryCacheMonitorCacheStatisticWithName: *aString*
Return the value of the cache stat with the given name for the cache monitor process running on the stone's host. The name must match an element of the Array returned by the #cacheStatisticsDescription method applicable to the cache monitor process. The UserTime and SysTime statistics cannot be accessed using this method. Returns nil if a statistic matching *aString* was not found for the cache monitor.

System class >> myCacheStatisticWithName: *aString*
Return the value of the cache stat with the given name for the current session. The name must match an element of the Array returned by the #cacheStatisticsDescription method applicable to a gem process. The UserTime and SysTime statistics cannot be accessed using this method.Returns nil if a statistic matching *aString* was not found for the cache monitor.

Determining slots and types

System class >> cacheSlotForProcessId: *anInt*
Return the cache slot at being used by the given process ID, or -1 if the slot was not found.

System class >> processIdForSlot: *anInt*
Return the process ID of the process using the cache slot at index anInt. Return -1 if the slot is not in use or out of range.

System class >> processKindForSlot: *anInt*
Return the process kind occupying the given cache slot as follows:

1 = Shared Page Cache Monitor
2 = Stone
4 = Page Server
8 = Gem (including Topaz, GBS, and other GCI applications).

A return value of -1 means the slot index was out of range or the slot is not in use.

Host CPU statistics

The statistics for UserTime and SysTime require an OS call, which can cause cache statistics to impact performance. To allow finer control, these statistics are now not part of the information returned by regular cache statistics interface methods. To get this information, use the following new methods.

hostCpuStatsForProcessId: *anInt*
Return an Array of 2 integers as follows:

array[1] - user mode CPU milliseconds
array[2] - system mode CPU milliseconds

Both array elements will be -1 if the process slot is out of range or not in use or if this method is not supported for the host architecture.

It is not required to be attached to the shared page cache or even be a GemStone process. The method will success for any process for which the gem session has permission to view its CPU usage statistics.

hostCpuStatsForProcessSlot: *anInt*
For the process using the cache process slot anInt, return an Array of 2 integers as follows:

array[1] - user mode CPU milliseconds used
array[2] - system mode CPU milliseconds used

Both array elements are set to -1 if the process slot is out of range or not in use, or if this method is not supported for the host architecture.

New public API for session cache statistics

The method in System class to get and set session cache statistics (_sessionCacheStatAt:, etc.) have officially-public method interface, named without the leading underscore. The new methods are:

```
System class >> sessionCacheStatAt:
System class >> sessionCacheStatAt:put:
System class >> sessionCacheStatAt:decrementBy:
System class >> sessionCacheStatAt:incrementBy:
System class >> sessionCacheStatsForProcessSlot:
System class >> sessionCacheStatsForSessionId:
```

Setting name in cache

In addition, you there is now public API for setting the name of a session in the cache. Note that it has been disallowed to set the session's cacheName to a name that is in use by another session.

```
System class >> cacheName:
```

Although existing methods with the leading underscore are unchanged, we recommend using these public selectors.

15. Other New and Changed Methods

Reimplemented copy

Previously, during the copy operation, any post-processing or cleanup of the new object was done by code in the original object, requiring use of setter methods. Now, copy is implemented via:

```
self shallowCopy postCopy
```

which allows the copied object to perform cleanup on itself.

The methods **shallowCopy** and **postCopy** have been added to kernel classes to implement appropriate copy behavior.

This change should be transparent, but may impact the way you implement copy for you application objects.

Added and changed System methods

The method **System class >> sleep:** does not use the ProcessScheduler, and no other waiting GsProcesses will execute during execution of *sleep*. In order to sleep while allowing other processes to run, use Delay or Semaphore.

The method **System class >> flushAllExtents** now has no effect for file systems as well as raw partitions, and is deprecated.

The method **System class >> clusterBucket** has been removed. This method name is misleading, and it has been obsolete for many years. Replace with **System class >> currentClusterId**, which it previously called, or **System class >> currentClusterBucket**.

The method **findObjectsLargerThan:limit:** now finds only persistent objects, not temporary objects.

The following methods have been added:

System class >> commitsSinceCurrentView
Returns a SmallInteger, the number of commits that have occurred since the session obtained its current view. If the session has no current view due to a lost OT root error, then -1 is returned. A result of 0 indicates that the session has the most current view of the repository.

This value is also now available via **System class >> descriptionOfSession:**, in position 16 of the result Array. For more about this, see "Updated descriptionOfSession: result" on page 80.

System class >> hostCpuCount
Returns a SmallInteger, the number of CPUs on the host system. In most cases, this number represents the total number of CPU cores on the host system.

System class >> readHiddenSet: *hiddenSetSpecifier* **fromSortedFile:** *aString*
Load a file assumed to be a sorted bitmap file of objects produced by one of the Repository>>listInstancesInPageOrder: methods. The objects are loaded into the given hidden set, but the ordering by page ID is lost because hidden sets are always ordered by object ID.

System class >> sessionIdHoldingGcLock
Returns a SmallInteger, the session ID of the session currently holding the garbage collection lock. Returns 0 if the GC lock is free.

System class >> sessionIsOnStoneHost
Answer true if this session is running on the same host computer as the stone process.

System class >> sessionPerformingBackup
Returns the session ID of the session currently performing a full backup, or 0 if a full backup is not in progress.

UNIX password validation

The following methods have been added on System to allow a UNIX user id and password to be validated or encrypted. These methods are not related to the authentication for login to GemStone/S 64 via GemStone, UNIX, or LDAP authentication.

System class >> validatePasswordForUser: *uid* **password:** *pw* **isEncrypted:** *encrypted*
Validates the password *pw* for the user id *uid* using the getpwnam(), getspnam(), and crypt() C functions. If encrypted is true, then *pw* is assumed to be encrypted and is compared with the encrypted password returned by getpwnam() or getspnam(). If encrypted is false, then pw is assumed to be the clear-text password. Returns true if the password was successfully authenticated, or false otherwise. Requires the #UserPassword privilege.

System class >> encryptPassword: *pw* **withSalt:** *salt*
Calls the C function crypt() to encrypt pw, which is assumed to be the clear text password to be encrypted. Returns a new String which contains the encrypted password.

Updated descriptionOfSession: result

The Array returned by **System class >> descriptionOfSession:**, which was previously an 11-element array, now contains 16 elements. The added elements are:

12. The priority of the session (a SmallInteger).
13. Unique host ID of the host where the session is running (an Integer).
14. Time of the session's most recent request to Stone (from **System timeGmt**).
15. Time the session logged in (from **System timeGmt**).
16. Number of commits that have occurred since the session obtained its view.

Methods added to Date and DateTime

The following new convenience methods have been added:

Date >> next: *dayName*
Returns the next Date whose weekday name is *dayName*.

Date >> nextMonth: *monthName*
Returns the next Date that describes a date later than the receiver and has month named *monthName*.

Date >> previousMonth: *monthName*
Returns the next Date that describes a date earlier than the receiver and has month named *monthName*.

Date >> weekdayIndex
Returns a SmallInteger between 1 and 7 representing the day of the week of the receiver, where Monday is 1 and Sunday is 7.

Date class >> indexOfMonthName: *monthName*
Returns the index of the month named *monthName*. Returns a SmallInteger between 1 and 12 inclusive, where 1 signifies January.

DateTime >> addMs: *aNumber*
Returns a DateTime that describes a moment in time *aNumber* milliseconds later than that of the receiver.

DateTime class >> newGmtWithYear: *yearInt* **month:** *monthInt* **day:** *dayInt* **hours:** *hourInt* **minutes:** *minuteInt* **seconds:** *secondInt* **ms:** *msInt*
Creates and returns an instance of the receiver from the specified values, expressed as Greenwich Mean Time.

DateTime class >> newGmtWithYear: *yearInt* **month:** *monthInt* **day:** *dayInt* **milliseconds:** *millisecondsInt*
Creates and returns an instance of the receiver from the specified values, expressed as Greenwich Mean Time.

High priority waits

To allow waits that will interrupt the current thread, the following methods have been added:

Semaphore >> highPriorityWaitForMilliseconds: *millisecondCount*
If signalCount > 0, this decrements the count. Otherwise, this will suspend the active process and add it to the end of the receiver's list of suspended processes. It will wait for at least *millisecondCount* milliseconds for the receiver to be signaled. The priority of the active process is raised for the duration of the wait so that expiration of the wait will interrupt any infinite loop or infinite wait for socket activity in other processes. Returns true if the semaphore was signaled for the caller. Returns false if the timeout expires without a signal for the caller.

Semaphore >> highPriorityWaitForSeconds: *secondCount*
Same as **Semaphore >> highPriorityWaitForMilliseconds:**, but the wait is in seconds rather than milliseconds.

Delay >> highPriorityWait
Suspends the active process until the millisecond clock reaches the appropriate value. The active process priority is raised for the duration of the wait, so that expiration of the delay will cause the active process to resume, thus interrupting any infinite loop or infinite wait for socket activity in other processes.

Change in Time now

Now, the results of executing **Time now** are in the TimeZone of the gem (**TimeZone current**). Previously, the current OS system time, regardless of TimeZone, was returned. This only has impact if the Gem's timezone is not the same as that of the platform it is running on.

Session holding GC Lock

A method has been added to retrieve the session ID of the session that is holding the GC Lock.

`System class >> sessionIdHoldingGcLock`
Returns the session ID of the session currently holding the garbage collection lock, or 0 if the GC lock is free.

GemStone/S 64 Bit v3.0 - New Features

16. Ephemerons

An ephemeron is a type of object that provides a way to associate properties to an application object, allowing customized finalization on a potentially large structure of properties when the application object is no longer needed.

Any object with at least two instance variables may be made into an ephemeron, using the **`Object >> beEphemeron:`** method. Normally this would be a customized subclass. The first instance variable, which is used as the "key", references the application object and acts as the weak reference, while all other instance variables of the ephemeron may hold properties. When in-memory garbage collection determines that the only references to an object are from an (itself unreferenced) ephemeron, this ephemeron is "fired". It stops being an ephemeron, and is sent the message **`mourn`**, which it may override to perform appropriate finalization.

The properties referenced by other slots in the ephemeron may reference the object without preventing this finalization process.

The following methods provide ephemeron support:

```
Object >> beEphemeron:
Object >> isEphemeron
Object >> mourn
```

17. Foreign Function Interface (FFI)

In addition to UserActions, GemStone/S 64 Bit v3.0 supports a new interface to call out to C code, the Foreign Function Interface (FFI). Using the FFI, you can access C functions in external libraries without the need to write UserActions.

The core FFI defines six new classes: CLibrary, CFunction, CPointer, CByteArray, CCallout, and CCallin.

NOTE

With UserActions, your code is checked against function prototypes of the external library that you're calling. With the FFI, no such checking takes place.

CLibrary

An instance of CLibrary corresponds to a C compiled library. Instances of CLibrary are created using:

`CLibrary class >> named:`*libraryName*

passing in the path and name of the C shared library to be loaded. The platform-specific extension (such as .so) is optional.

CCallout

Individual functions within a CLibrary are represented by instances of CCallout. To create a CCallout, the following class methods are available:

`library:` *aCLibrary* **`name: result:`** *resType* **`args:`** *argumentTypes*

`library:` *aCLibrary* **`name: result:`** *resType* **`args:`** *argumentTypes* **`varArgsAfter:`** *varArgsAfter*

`name: result:` *resType* **`args:`** *argumentTypes*

`name: result:` *resType* **`args:`** *argumentTypes* **`varArgsAfter:`** *varArgsAfter*

aCLibrary may be an instance of CLibrary, an Array of CLibraries, or nil. Nil will cause search of the loaded libraries for a function of this name. *aName* is a String providing the name of the specific function. *resType* is the return type of the function, and *argumentTypes* is an array of zero or more symbols describing the types of the argument for this function.

The following instance method is used to invoke the function described by the instance of CCallout:

`callWith:` *argsArray*

C type symbols

The following table lists the symbols used for creating *resType* (result type) and *argumentTypes* arguments when creating CCallouts.

C type symbols

	Return type	Argument type
#int64	Integer. The C function returns an int64, or any unsigned C integer smaller than 64 bits.	Integer
#uint64	Integer. The C function returns a uint64.	Integer
#int32	Integer. The C function returns a signed C integer 32 bits or smaller.	Integer
#uint32	Integer. The C function returns an unsigned C integer, 32 bits or smaller.	Integer

C type symbols (Continued)

	Return type	Argument type
#int16	Integer	Integer
#uint16	Integer	Integer
#int8	Integer	Integer
#uint8	Integer	Integer
#double	SmallDouble or Float. The C function returns a C double.	SmallDouble or Float; and the function is limited to a maximum of four arguments.
#float	SmallDouble or Float. The C function returns a C float.	SmallDouble or Float
#'char*	nil or a String	The corresponding arg must be a String. The body is copied to C memory before call and copied from C memory (and possible grown/shrunk) after call. C memory will not be valid after the call finishes.
#void	nil	
#ptr	nil or a CPointer	The corresponding arg must be nil, a CByteArray or a CPointer. If nil, a C NULL is passed. If CByteArray, address of body is passed. If CPointer, the encapsulated pointer is passed.
#'&ptr'		The corresponding arg must be a CPointer. The CPointer's value will be passed and updated on return.
#'&int64'		The corresponding arg must be a CByteArray of size 8. A pointer to body will be passed.
#'&double'		The corresponding arg must be a CByteArray of size 8. A pointer to body will be passed.
#'const char*'		The corresponding arg must be nil (to pass NULL) or a String (body is copied to C memory before call) C memory will not be valid after the call finishes.

Platform-specific limitations

Not all platforms support native code generation.

On platforms supporting native code:

- Functions using varArgs may have a maximum of 20 variable arguments.
- Functions with one or more args of C type double are limited to a maximum of four arguments.

On platforms that do not support native code:

- Functions using varArgs may have a maximum of four fixed and 10 total arguments.
- Functions not using varArgs are limited to a maximum of 15 total arguments.
- Arguments and results of C type float are not supported.
- Functions with one or more args of C type double are limited to a maximum of four arguments.

CCallin

A CCallin represents a signature for a C function to be called by C code. The resulting CCallin may be used as a type within the *argumentTypes* array when defining a CCallout.

CByteArray

A CByteArray represents an allocation of C memory. When objects such as pointers or strings are passed to or from C functions, creating a CByteArray, with memory malloc'ed, ensures that the memory will be valid following the call.

CFunction

CFunction is an abstract superclass representing the type signature of a C function. It has two subclasses, CCallout and CCallin.

CPointer

CPointer encapsulates a C pointer that does not have auto-free semantics. New instances are created by CFunction calls with result type #ptr, and are also used for certain arguments of CFunctions.

FFI Wrapper Utilities (CHeader)

In addition to the core classes described above, the FFI provides the class CHeader, along with three internal classes that are used by CHeader: CDeclaration, CPreprocessor, and CPreprocessorToken.

You can use CHeader to generate wrappers for C functions and structures. The CHeader instance method

`wrapperNamed:` *nameString* **`forLibraryAt:`** *pathString* **`select:`** *aBlock*

generates a wrapper for the named function.

For more details about the FFI wrapper utilities, see the "Foreign Function Interface" chapter of the *GemStone/S 64 Bit Programming Guide* for version 3.0.

18. NotTranloggedGlobals

In version 3.0, it is now possible to have persistent objects in the repository for which changes are not recorded in the transaction logs. This can save significant tranlog space; however, these objects cannot be recovered from the tranlogs in the event of a crash.

Objects that are intended to be persistent, but not tranlogged, should be referenced by the new variable in Globals, **NotTranloggedGlobals**. Such objects must have no other references from persistent objects.

Objects that are reachable from AllUsers (the regular root for all persistent objects) may not reference anything that was previously only reachable from NotTranloggedGlobals; this will generate an error on commit.

You can enumerate or roll back the dirty not tranlogged objects using **`System class >> enumerateDirtyList:`** or **`rollbackDirtyList:`** with an argument of 2.

19. New Collection Classes

See also "PositionableStream changes to allow ANSI and Legacy to coexist" on page 69, for the new classes related to PositionableStream.

Global Transcript

As called for by the ANSI standard, GemStone now includes a global Transcript. This is implemented using the new class TranscriptStreamPortable, supporting the ANSI WriteStream protocol.

The Transcript is designed to be conflict-free; any sessions may write to the Transcript. This input is buffered internally, and on **`show:`** or **`flush`**, the contents are written to the Gem's log or topaz output using **`GsFile gciLogServer:`**.

The association for Transcript in Globals is world writable, so any session can install a customized implementation of Transcript.

TransientShortArray

Version 3.0 introduces this new subclass of ByteArray. Instances of TransientShortArray may not be committed. They provide contiguous in-memory physical sizes of up to 65K bytes of 16-bit signed integers.

ReadWriteStream and ReadWriteStreamPortable

The classes ReadWriteStream and ReadWriteStreamPortable have been added for ANSI compliance, with ANSI protocols. ReadWriteStream is part of the portable PositionableStream hierarchy, and only available when the portable/ANSI streams are installed. ReadWriteStreamPortable is available regardless of which PositionableStreams are installed. See "PositionableStream changes to allow ANSI and Legacy to coexist" on page 69 for details.

ReadWriteStream provides a stream with both read and write protocol. For details about ReadWriteStream methods, refer to the image.

FileStream and FileStreamPortable.

The class FileStream and FileStreamPortable have been added for ANSI compliance, with ANSI protocols. It is implemented using GsFile.

FileStream is part of the portable PositionableStream hierarchy, and only available when the portable/ANSI streams are installed. FileStreamPortable is available regardless of which PositionableStreams are installed. See "PositionableStream changes to allow ANSI and Legacy to coexist" on page 69 for details.

BitSet

The new class BitSet is available for efficient operations on Arrays of up to 130144 bits. BitSets will grow as needed up to the maximum size, to accommodate bits being set.

20. WebTools

JSON (JavaScript Object Notation) support

The following methods have been added to the image, and are understood and responded to appropriately by most GemStone classes:

```
printJsonOn:
jsonKeys
asJson
```

WebTools Example

GemStone/S 64 Bit version 3.0 includes prototype code for a lightweight web interface, WebTools, providing the ability to access some GemStone process and statistics information and to browse code. This is a simple proof-of-concept; not all functions in the existing interface are implemented. WebTools uses primarily javascript and web graphics to retrieve information from, and potentially make changes to, the GemStone server.

To use WebTools, follow the instructions in $GEMSTONE/examples/www/readMe.txt. The code is executed in a Topaz session, which provides the interface and must remain running. Object visibility and access in WebTools is that of the user that the Topaz session logged in as.

WebTools currently runs on Firefox and Chrome but does not run successfully on Internet Explorer.

GemStone/S 64 Bit v3.0 - Utility and Environment Changes

21. Cache Warming Changes

GemStone/S 64 Bit v3.0 includes a new interface to perform cache warming, using the new configuration options STN_CACHE_WARMER and STN_CACHE_WARMER_SESSIONS. These configuration parameters specify, respectively, cache warmer configuration and the number of cache warmer sessions to start on Stone startup. The default is to not start cache warmers on Stone startup.

Cache warming makes use of multi-threaded scanning. These cache warmer sessions are slave threads, not complete GemStone sessions.

For more information on these parameters, see “STN_CACHE_WARMER“ and “STN_CACHE_WARMER_SESSIONS“ starting on page page 104.

There are also changes in the startcachewarmer script - see “startcachewarmer uses multi-threaded scan” on page 90. While the startcachewarmer script is still available, we recommend using the configuration parameters instead.

22. Changes to Utility Functions

gslist changes

Now included in Windows distributions

The gslist -m option allows you to list GemStone processes on a different server. To allow this to be done from Windows, the distribution file for Windows now also include the gslist executable.

Note that gslist on Windows will query a compatible netldi on the remote host for this information. The environment for gslist must set %GEMSTONE_NRS_ALL% to the correct netldi, if the remote host does not use gs64ldi as the v3.0 netldi name.

gslist for version 3.0 cannot communicate with version 2.x netldi.

Additional status

In addition to "exists", "running", and similar statuses displayed in the gslist output, the new status "startup" will be displayed for Stone processes that are performing startup and recovery.

pageaudit utility now audits data pages by default

In previous releases, the pageaudit utility only audited non-data pages: object table pages, bitmap pages, etc. Data pages were audited separately by objectAudit.

In version 3.0, pageaudit also checks data pages, reading them into memory and confirming that they are data pages. Data page audit is the new default. To disable audit of data pages and revert to the pageaudit behavior of previous releases, use the new pageaudit **-d** option.

```
Usage: pageaudit [-d] [-h] [-f] [-k logfile ] [-e execfg]
[-z syscfg] [name]
  -d     disables auditing of data pages.
  -e     specifies executable specific configuration file.
  -f     keeps running beyond first error if possible
  -h     prints usage information and exits.
  -l     write output to logfile
  -z     specifies system configuration file.
  name   the name of the stone (default is gs64stone-audit).
```

startcachewarmer uses multi-threaded scan

The startcachewarmer utility now uses the multi-threaded scan (see "New Multi-Threaded Operations" on page 56). As a result, some parameters have changed; there is no longer a need to use the -u and -p options to pass in user name and password.

```
Usage: startcachewarmer [-d|-D][-h][-l limit][-n numSessions]
            [-s stone ][-W][-L path]
  -d     read data pages into the cache (default: only object
         table pages are read)
  -D     read data pages into the UNIX file system buffers.
  NOTE: -d and -D are mutually exclusive. If both -d and -D are
  passed on the command line, the last of these flags is used.
  -h     display this message and exit
  -l     stop cache warming if the free frame count
         falls below <limit>. Use -1 to have system compute a
         default value. Use 0 to force cache warming to continue
         even if the shared cache is full. (default: -1)
  -L     path to a writable log file directory (default: current
         directory)
  -n     number of worker sessions to use (default:
         numberOfCpus + numExtents)
  NOTE: there is always one additional "master" session allocated
  and the warmer will exit with an error if not enough sessions are
  available.
  -s     name of running stone (default: 'gs64stone')
  -W     wait for cache warming gems to exit before exiting this
         script (default: spawn gem in the background and exit
         immediately)
```

startnetldi added -P option

startnetldi now has an additional parameter, **-P**, to specify the well-known port that the netldi will listen on.

```
Usage startnetldi [-h] [-d] [-g|-s] [-n] [-a account] [-l logFile]
         [-p low:high] [-P portNumber] [-t seconds] [name]
  -h  prints usage information and exits.
  -d  cause netldi to print extra information to its log file.
  -g  start netldi in guest mode.
  -s  start netldi in secure mode.
  -n  netldi will not use ad hoc scripts.
  -a  all processes started by the netldi will belong to this
        account.
  -l  specifies the name of the netldi log file.
  -p  the low and high ports in the pool of ports to use in
      creating processes.
  -P  specifies the well-known port number that netldi will use.
  -t  number of seconds netldi will wait for child process to
      start.
  name the name of the netldi to start (default is gs64ldi).
```

stopstone blocks on cache shutdown

By default, stopstone now blocks until the shared page cache is gone.

You can override this using the new option -t, which specifies how long to wait.

```
Usage: stopstone [-h] [-i] [-t timeout]
        [name [account [password]]]
  -h     prints usage information and exits.
  -i     stops the stone immediately even if others are logged in.
  -t     specifies how long to wait for other processes to detach
         the cache. Default is -1 which means wait forever.
  name     is the name of the stone to stop (default is gs64stone).
  account  a privileged GemStone user ID.
  password for the specified account (will prompt).
```

Note that **System class >> shutDown** will also block, waiting for cache shutdown.

topaz changes

A command line option, -u, has been added to topaz. This **-u** option provides a string that has no effect on the executable, but may be useful in tools outside of GemStone that display process information that include commands lines, such as **top**.

The topaz header now includes a line indicating if a **.topazini** file is used, and the name and location if so.

There are many changes in topaz commands; see Chapter 3, "Changes in Topaz" starting on page 121.

printlogs and searchlogs

These utilities have been renamed; the name no longer includes a final **.sh**.

Help text is now available with the -h option, as well as by executing without arguments, for consistency with other utilities.

printlogs and searchlogs now search on additional fields

Tranlog analysis, using **printlogs** and **searchlogs**, permits searching of the transaction logs for exact details on data modifications, which is useful for audits under Sarbanes-Oxley.

In previous releases, logs could be filtered for GemStone user, host name, and client ID address. InGemStone/S 64 Bit v3.0, additional information is written to the logs when a user logs in, providing the following additional filter options:

euid *integer*	Effective UNIX user ID
ruid *integer*	Real UNIX user ID
euidstr *string*	Effective UNIX user name
ruidstr *string*	Real UNIX user name
gempid *integer*	Gem process ID
sessionid *integer*	Gem session ID

These options are available for search on both **printlogs** and **searchlogs**. For **searchlogs**, the search terms must be in the following order:

```
user host client euid ruid euidstr ruidstr gempid sessionid
```

printlogs new keyword

In addition to **full** and **all**, you may also use the new keyword **nouserinfo**, which will skip printing the user information for each record. If you do not need user information for the record this may make the output somewhat smaller and generate faster.

statmonitor changes

Change in behavior with the -r flag

Previously, when the -r flag was used to start statmonitor, if the stone was shut down or died for any reason, statmonitor would wait for up to 120 seconds for the stone to restart, in which case (on some platforms), it would attach and monitor the new cache.

Now, when the stone terminates, any statmonitor monitoring the caches for that stone will shut down and not restart. In order to monitor the restarted stone, statmonitor must be invoked again; we recommend starting the stone using a script which also starts statmonitor.

The -r flag now may only be used in combination with either the -t or -h flags.

statmonitor specification for monitoring remote cache

To avoid the need to type in the new cache name when monitoring a remote cache, statmonitor can now determine the cache name based on a Stone name passed in. You can start statmonitor on a host machine remote from the Stone that is running a remote cache using the Stone name alone, provided that there are not remote caches for multiple Stones with the same name on that machine.

The statmonitor option **-m** *stoneHostName* is no longer necessary and has been removed.

23. Scripts that start system Gems

Previously, the **$GEMSTONE/sys/rungcgem** script was used to start the SymbolGem, as well as both kinds of GcGem and other system Gems. Since one script was used, it was difficult to tune the configuration for specific system Gems.

Now, each system Gem has its own startup script, and the previously used **rungcgem** script has been removed.

runotcachewarmergem
Starts the Object Table (OT) cache warmer Gems.

runsymbolgem
Starts the SymbolGem.

runpagemgrgem
Starts the page manager Gem.

runsymbolgemconv
Starts the SymbolGem, specifically for conversion.

runadmingcgem
Starts the Admin GcGem.

runcachewarmergem
Starts the cache warmer Gems.

runreclaimgcgem
Starts the Reclaim GcGem.

System process log file deletion

The page manager and symbol gem logs are by default not deleted, but you can edit the appropriate script to allow them to be deleted on normal exit (clean shutdown).

24. Environment Variable Changes

Added environment variable GEMSTONE_LIB

Version 3.0 introduces the environment variable GEMSTONE_LIB, to allow specification of the directory for the gem and gem dynamic libraries. This is primarily of use in debugging low-level problems, and is used by the gemnetdebug script to specify the slow gem and gem dynamic libraries.

Added environment variable GS_PAGE_MGR_PRINT_REMOTE_STACKS

If this variable is set, if a remote cache page server becomes stuck, the page manager will request that the remote cache page server print its call stack to its log file.

Added environment variable GS_DEBUG_COMPILE_TRACE

This variable is intended to for gem debugging using gemnetdebug. It is disabled (set to 0) by default. If set, it allows tracing method compiles. The following are valid values:

0 - no tracing
1 - one line (class,selector) of each method compiled
2 - in addition to above, bytecode disassembly
3 - in addition to above, native code assembly listing

25. Distribution File Changes

Changes in required shared libraries

The specific shared libraries that GemStone creates have changed; certain shared libraries are no longer needed. This is for the most part transparent to the user.

As a result of this, the shared libraries required by Windows GCI clients, including GBS, no longer require libgs*.dll; you no longer need to copy this file to GBS client machines. All functionality is statically linked into libgcirpc*.dll. You do need to include the library `msvcr90.dll`.

Additional shared libraries distributed

Additional shared libraries are now included in the distribution for support of new GemStone features.

A new directory, **`libSlow`**, contains slow gem executable and libraries for use in debugging customer problems. See "Added environment variable GEMSTONE_LIB" on page 94.

Example configuration files

The example configuration files are no longer distributed.

gslist for Windows

The **`gslist`** executable is now included in the distribution for Windows; see page 89.

Examples includes testing code

The examples directory includes a new subdirectory, testing, which contains portions of the framework based on SUnit that GemStone uses internally for image level testing.

26. Changes in Globals

The **NativeLanguage** Global is no longer included in the UserGlobals and Globals symbolDictionaries. It is not removed in upgraded repositories, and remains usable by applications.

Instances of GsCurrentSession no longer cache nativeLanguage. If the symbolList does not include a key #NativeLanguage, **`GsCurrentSession >> nativeLanguge`** returns #English.

MinutesFromGMT is also no longer present in new repositories.

BackupLog is no longer used or updated, although it may be present in upgraded repositories. This variable was originally intended to record backups, but since it did not record any subsequent errors in the backup or successful completion, it had little value. The start and successful completion of programmatic backups is recorded in the stone log.

In addition, other new Globals have been added. These are documented in the sections of this document that describe the relevant feature.

27. Cache Statistic Changes

New protocol is available to access cache statics from the image; see “Changes in Cache Statistics Programmatic Interface” on page 75.

VSD changes

Main window additional column

The VSD graphical application main window now includes a column with the sessionId, and the PID column has been relabled "ProcessId".

Cache Statistics Kinds updated

The data structure inside VSD include kinds for each cache statistics; kinds include milliseconds, Operations, Requests, Pages, etc., depending on the specific statistic. These have been cleaned up and corrected.

Cache Statistic Name/Units changes

GemStone/S 64 Bit version 2.4 introduced updates to VSD and statmonitor, allowing values greater than 32-bit to be generated and displayed. In version 2.4, Session and Global cache statistics changed to be 64 bit and could take 64 bit Integer values.

To work around the 32-bit limitation in versions earlier than v2.4, some cache statistics which normally held very large values, such as FreeOopCount, were renamed to FreeOopsK, and reported in units of 1000s. In version 3.0, these statistics have been renamed without the “K”, and are reported exactly rather than in unit of 1000s.

Process type	Pre-3.0 name	Statistics Name in 3.0
ShrPcMon	TotalLocalPageCacheKHits	TotalLocalPageCacheHits
ShrPcMon	TotalLocalPageCacheKMisses	TotalLocalPageCacheMisses
ShrPcMon	TotalKFramesFromFreeList	TotalFramesFromFreeList
ShrPcMon	TotalKFramesAddedToFreeList	TotalFramesAddedToFreeList
ShrPcMon	TotalKPcesRemovedFromFreeList	TotalPcesRemovedFromFreeList
ShrPcMon	TotalKPcesAddedToFreeList	TotalPcesAddedToFreeList
Stone	PossibleDeadKobjs	PossibleDeadObjs
Stone	VoteNotDeadKobjs	VoteNotDeadObjs
Stone	DeadNotReclaimedKobjs	DeadNotReclaimedObjs
Stone	EpochNewKobjs	EpochNewObjs
Stone	EpochScannedKobjs	EpochScannedObjs
Stone	EpochPossibleDeadKobjs	EpochPossibleDeadObjs
Stone	FreeOopsK	FreeOops

Process type	Pre-3.0 name	Statistics Name in 3.0
Stone	OopNumberKHighWaterMark	OopNumberHighWaterMark
Gem	ProgressKobjs	ProgressCount This cache statistic is now also used to track the progress of a auditIndexes.

In addition to the above renamed statistics, the numbers cache statistics have been renamed to include an extra 0, so they will sort correctly.

SessionCacheStat0...**SessionCacheStat9** have been renamed to **SessionCacheStat00**...**SessionCacheStat09**. These are also now capitalized for all cases.

GlobalStat0...**GlobalStat9** have been renamed to **GlobalStat00**...**GlobalStat09**.

PercentCpuUsed

This Gem OS statistic is now also available on Linux; previously it was Solaris only.

This is reported as the percent of the total number of CPUs that this gem is using. While this is information originating from the OS, PercentCpuUsed is reported in a way that is consistent with Solaris' reporting of this statistic. On Linux, top reports the percentage of one CPU that the Gem is using, rather than the percentage of all CPUs. The statistic value will not match what top reports on Linux.

Improved internal statistic typing information

To better support the WebTools example code, a number of changes and corrections have been made in the counter information for statistics.

Removed cache statistics

The following cache statistics have been removed or replaced by new statistics:

CodeGenClearSendCachesCount
CommitRecordsReadCommitWithoutTokenCount — replaced by
    CommitRecordsReadCommitBeforeCommitQueueCount and
    CommitRecordsReadCommitInCommitQueueCount
LocalPageCacheWrites
LoginQueueSize
PagesNeedRemovingThreshold
SpinLockSmcQSleepCount
StnAioCompleted1Suspend
StnAioCompletedNoSleep
StnAioCompletedNoSuspend
StnAioLastSuspendCount
StnAioSuspendPrematureReturn
StnAioSuspendTimeoutCount
StnAioSyncWritesAfterCancel
StnAioTotalSleepCount

StnAioWaitLastTime
StnAioWriteEAGAIN

New cache statistics

Version 3.0 introduces the following new cache statistics:

AbortInProgress (Gem)
A boolean indicating if the session is currently performing an abort operation.

CacheStartQueueSize (Stone)
Shows the number of sessions waiting for remote cache startup.

ClientAbortInProgress (Page Server)
A boolean indicating if the page server's client gem is currently performing an abort operation.

CommitRecordsReadCommitBeforeCommitQueueCount
The number of commit records read by the session before it requested the commit token during a commit operation.

CommitRecordsReadCommitInCommitQueueCount
Number of commit records read by the session after it has requested but not yet received the commit token.

Env1sendCacheSerialNum
Counter on env 1 send site caches.

ExportedSetSizePinnedInMemory (Gem)
The number of objects in the ExportSet which cannot drop out of temporary object memory nor be garbage collected by in-memory collection of temporary objects. Will include any objects represented in ExportedSetSize that are not committed. Can be less than ExportedSetSize if ((System gemConfigurationAt:GemDropCommittedExportedObjs) == true).

GciRpcKeepAlivePacketCount (Gem)
Number of keep-alive packets received from the GCI client on a remote host.

GcLockKind (Stone)
Indicates the state of the garbage collection lock and why it is being held:

0 - Free
1 - MarkForCollection
2 - FindDisconnectedObjects
3 - Pre-MarkGcCandidates (loading candidate objects)
4 - MarkGcCandidates
5 - Epoch GC
6 - Reclaim All
7 - (not used)
8 - Repository Scan (listInstances, listReferences, etc)

IndexProgressCount (Gem)
Used to monitor the status of certain index operations:

0 - Inactive
1 - Identity index audit in progress.
2 - Equality index audit - auditing root terms.
3 - Equality index audit - auditing NSC counts.
4 - Equality index audit - auditing btree counts.
5 - IndexManager>>removeAllIndexes in progress.

LastCommandFromClient (Pgsvr)
The numeric index of the last message sent to the page server but its client.

LastErrorNumber (Gem)
The number of the last error processed by the session. Fatal errors are not reported in this statistic.

LookupCacheSerialNum
Counter on per-class method lookup caches.

LostOtDeferCount (Stone)
The total number of times stone has deferred sending a LostOtRoot signal to a session.

LosOtDeferLastReason (Stone)
An integer value indicating the most recent reason deferred sending a LostOtRoot signal to a session. The codes are defined as follows:

0 - none - no LostOtRoot signals have been deferred.
1 - the session was executing a garbage collection primitive (markForCollection, etc).
2 - the session held the commit token or was waiting in the commit queue.
3 - the session was blocked on an internal stone queue.
4 - the session was waiting in the run queue or the SMC queue.
5 - the session was performing an abort.
6 - the session was a system gem.

LostOtDeferLastSession (Stone)
The session ID of the last session for which the stone deferred the sending of a LostOtRoot signal.

MlClearAllCount
Number of times all send-site caches were cleared.

MlFullLookupCount
Number of times that fullMethodLookup was called.

MlLuCacheGrowCount
Number of times a per-class lookup cache was grown to larger table size.

MlLuCacheLargeCount
Number of times a per-class lookup cache became a large object.

MlLuCacheResetCount
Number of times a per-class lookup cache was reset.

MlPolyCacheCreateCount
Number of times that a polymorphic send-site cache was created.

MlPolyCacheFullCount
Number of times that a polymorphic send-site cache overflowed.

MlPolyCacheGrowCount
Number of times that a polymorphic send-site cache was grown.

MtActiveThreads (Gem)
The number of threads actively working on a multi-threaded task in a Gem executable.

MtMaxThreads (Gem)
The maximum number of threads currently configured for a multi-threaded task in a Gem executable. Set by the method that initiated the operation.

MtPercentCpuActiveLimit (Gem)
A statistic that can be set by a session to control the maximum number of active threads working on the task. Value must be between 0 and 100. When a non-zero value is set, the controller thread attempts to keep the percentCpuActive stat below this value by limiting the number of threads working on the task.

MtThreadsLimit (Gem)
A statistic that can be set by a session to control the maximum number of active threads working on the task. Must be less than or equal to MtMaxThreads.

NewObjsCommittedNotLogged (Gem)
Number of newly created objects that were committed but not added to the tranlog because they are reachable from the NotTranloggedGlobals root object.

NumEphemerons (Gem)
The number of live ephemerons remaining at the end of the last mark/sweep garbage collection.

NumProcsSleepingForLock (shrpcmon)
Number of processes on this shared page cache that are currently sleeping while waiting to acquire a spin lock.

ObjsCommittedNotLogged (Gem)
The total number of objects (new and modified objects) reachable from the NotTranloggedGlobals root object which the session has committed.

PageMgrCompressionEnabled (Stone)
A boolean value: true if the page manager session is compressing the list of pages that it sends to remote cache page servers, false otherwise.

PageMgrPageRemovalRetryCount (Stone)
Number of times the page manager session has retried removing one or more pages from the shared page caches because the first attempt to remove the pages failed.

PageMgrPrintTimeoutThreshold (Stone)
The time threshold in seconds used by the page manager to decide if a message should be printed to its log file indicating a remote cache page server was slow to respond.

PageMgrRemoteCachePgsvrTimeout (Stone)
Number of seconds the page manager session will wait to receive a response from a remote cache page server.

PageMgrRemoveMaxPages (Stone)
The maximum number of pages the Stone will return to the page manager session in a single batch.

PageMgrRemoveMinPages (Stone)
The minimum batch size of pages that the page manager will process. The page manager will request pages to process from the Stone only if the Stone cache statistic PagesWaitingForRemovalInStone exceeds this value.

ProcessesWaitingForQueueLocks (Shrpcmon)
Number of processes attached to the shared cache that are spinning while attempting to acquire a queue lock.

SessionPerformingBackup (Stone)
The session ID which is performing a full backup, or 0 if a full backup is not in progress.

StnAioNumWriteThreads (Stone)
Value of the STN_NUM_AIO_WRITE_THREADS Stone configuration parameter, which determines how many threads are dedicating to writing to the tranlog.

StnAioWriteQueueHighWaterSize (Stone)
High water mark of the StnAioWriteQueueSize statistic.

StnAioWriteQueueSize (Stone)
Size of the tranlog write request queue.

StnAioWritesQueuedCount (Stone)
Total number of tranlog write requests that have been queued.

StnAioWriteThreadsIdle (Stone)
Number of tranlog write threads in the stone process which are currently idle.

TimeInWaitsForOtherReaders
The real number of milliseconds the process spent waiting for the read of another process to complete.

TotEphemeronsFired (Gem)
Total number of ephemerons whose key has been garbage collected. Only updated at the end of a mark/sweep GC.

28. Configuration Parameter Changes

Configuration parameter change reporting

Changes to runtime configuration parameters are reported in the stone log. The format of these reports has been improved to more consistently use the exact name of the parameter that is changed.

Changed configuration parameters

The following configuration parameters have been changed:

STN_MAX_REMOTE_CACHES

The default has changed from 16 to 255.

The specified maximum has changed from 65535 to 10000; however, this is not a relevant change, since the number of remote caches is limited by the number of sessions, and the maximum for STN_MAX_SESSIONS is and was 10000.

STN_HALT_ON_FATAL_ERR

The default has changed from TRUE to FALSE.

STN_PAGE_REMOVAL_THRESHOLD

The configuration parameter STN_PAGE_REMOVAL_THRESHOLD has been renamed to STN_PAGE_MGR_REMOVE_MIN_PAGES.

A related parameter, STN_PAGE_MGR_REMOVE_MAX_PAGES, has been added.

STN_PAGE_MGR_REMOVE_MIN_PAGES must be equal to or less than STN_PAGE_MGR_REMOVE_MAX_PAGES.

The runtime equivalent has changed from #StnPageRemovalThreshold to #StnPageMgrRemoveMinPages.

The cache statistic has changed from PagesNeedRemovingThreshold to PageMgrRemoveMinPages.

The maximum has changed from 1792 to 16384.

Removed configuration parameters

STN_AIO_WAIT_TIME

STN_MAX_SLEEP_TIME

STN_NUM_LOOPS_PER_NET_POLL

STN_OOB_SOCKET_POLL_INTERVAL

GEM_IO_LIMIT

The following configuration parameter permitted caches with an initial smaller footprint on HP-UX and AIX platforms, and was not used on Solaris or AIX. Growable caches are not available in this release for HP-UX and AIX.

GEM_TEMPOBJ_INITIAL_SIZE

Added configuration parameters

The following configuration parameters have been added:

GEM_ABORT_MAX_CRS

When a Gem is not in a transaction and aborts, this parameter specifies the maximum number of commit records to analyze to compute the writeSetUnion since the last time this session aborted. If the number of commit records would exceed this limit, the abort is treated similar to a LostOt and all in-memory copies of committed objects are marked invalid and will be re-read as needed during subsequent execution.

A value of 0 (zero) means no limit on number of commit records to analyze.

Min: 0
Max: 2147483647
Default: 0

GEM_FREE_PAGEIDS_CACHE

Specifies the maximum number of free pageIds to be cached in Gem. Larger values reduce number of calls to Stone, at a cost of needing more free space within the extents.

Runtime equivalent: #GemFreePageIdsCache
Min: 40
Max: 3500
Default: 200

GEM_NATIVE_CODE_ENABLED

If true, enables the generation and use of native code.

Runtime equivalent: #GemNativeCodeEnabled can be used to disable native code, but cannot be used to reenable native code once it has been disabled during a session.

Runtime equivalent: #GemNativeCodeEnabled
Default: TRUE.

GEM_PGSVR_COMPRESS_PAGE_TRANSFERS

If TRUE, use compress2() from zlib library with default compression level to compress page transfers between pgsvr on Stone's machine and Gem or mid-cache pgsvr.

For the first Gem to login on a remote machine, that Gem's configuration file value of this parameter is propagated to the page manager and is used to configure the page manager's communication to the page manager's pgsvr on the new remote cache.

When a Gem triggers creation of a mid-level cache, via the method **`midLevelCacheConnect:cacheSizeKB:maxSessions:`**, that Gem's current runtime value of this parameter is propagated to the page manager and is used to configure the page manager's communication to the page manager's pgsvr on the new mid-level cache.

Runtime equivalent: #GemPgsvrCompressPageTransfers
Default: FALSE

GEM_RPC_KEEPALIVE_INTERVAL

Interval in seconds for the RPC GCI client keep-alive packet to be sent on the seldom used out-of-band (OOB) socket between the Gem and the GCI client. Has no effect for linked sessions or RPC sessions running the same host as the Gem process.

If enabled, keep-alive packets are sent during the GciPollForSignal() call.

Min: 0
Max: 7200
Default: 0 (disabled)

GEM_TEMPOBJ_SCOPES_SIZE

Size of the scopes stack in Gem temporary object garbage collection.

This value is set at Gem initialization (linkable GciInit or rpc GciLogin) and cannot be changed without restarting the Gem process. The scopes stack consumes (8 bytes * GEM_TEMPOBJ_SCOPES_SIZE) of C heap memory.

Primary user-visible effect of this setting is maximum depth of nested expressions that can be compiled by the method compiler. Default setting is sufficient for expression nesting of about 200, such as in depth of nested parenthesized expressions.

Min: 1000
Max: 10000000
Default: 2000

SHR_WELL_KNOWN_PORT_NUMBER

Port number that the shared page cache monitor will use as its well-known port. The well-known port is used by all Gems and page servers on this host to connect to the cache monitor.

If the specified port is in use by another process, the monitor process will not start and exits with an error.

A value of zero indicates that the port number will be selected by the system.

Min: 1
Max: 65535
Default: 0

STN_CACHE_WARMER

Specifies whether to run the cache warmer when the Stone is started, and whether to load just the object table pages or both the object table and the data pages.

STN_CACHE_WARMER has the following possible values:

0 - Disabled, the Stone does not start the cache warmer. This is the default.
1 - Start the cache warmer and load only the object table pages.
2 - Start the cache warmer and load the object table and data pages.

Default: 0

STN_CACHE_WARMER_SESSIONS

Specifies the number of worker sessions to use in the cache warmer Gem. These are slave threads as described in the section "New Multi-Threaded Operations" on page 56. In addition, one additional "master" session is allocated. The cache warmer will exit with an error if not enough sessions are available.

Units: sessions
Cache statistic: NumCacheWarmers (Stone)
Min: 0
Max: 256
Default: 0; run the cache warmer with (numberOfCpus + numberOfExtents) sessions

STN_NUM_AIO_WRITE_THREADS

Number of native threads the Stone will start to perform writes to the tranlog. In commit-intensive system, this should be increased to 8 or even 16.

Cache Statistic: StnAioNumWriteThreads (Stone)
Min: 4
Max: 256
Default: 4

STN_PAGE_MGR_COMPRESSION_ENABLED

Determines if the page manager will compress the list of pages that it sends to remote shared page caches for removal. If set to TRUE, all lists of pages larger than 50 will be compressed before transmission. The same compressed list is used to send to all remote shared page caches; i.e., the compression operation is performed no more than once for each list of pages to be sent. Has no effect on systems that do not use remote shared page caches.

Runtime equivalent: #StnPageMgrCompressionEnabled
Cache Statistic: PageMgrCompressionEnabled (Stone)
Default: FALSE

STN_PAGE_MGR_PRINT_TIMEOUT_THRESHOLD

A threshold in real seconds used by the page manager to determine if a slow response from a remote shared page cache should be printed to the page manager log file. If a remote cache takes longer than this number of seconds to respond to the page manager, the page manager will print a message to the log file. If a remote cache takes less than this number of seconds to respond, no message is printed.

Note that this value controls the writing of log messages only. The connection to the remote cache will not be terminated by page manager unless STN_REMOTE_CACHE_PGSVR_TIMEOUT is exceeded.

Runtime equivalent: #StnPageMgrPrintTimeoutThreshold
Cache Statistic: PageMgrPrintTimeoutThreshold (Stone)
Min: 0
Max: 3600
Default: 5

STN_REMOTE_CACHE_PGSVR_TIMEOUT

Maximum time in seconds that the page manager session will wait for a response from a page server on a remote shared page cache. If no response is received within the timeout period, all Gems attached to that cache are logged off and a message is written to the Stone and page manager logs. Negative timeouts are not allowed. A timeout value of zero causes the page manager to wait forever.

Runtime equivalent: #StnRemoteCachePgsvrTimeout
Cache Statistic: PageMgrRemoteCachePgsvrTimeout (Stone)
Min: 0
Max: 3600
Default: 15

STN_PAGE_MGR_REMOVE_MAX_PAGES

Maximum batch size for the page manager system Gem. This value is the maximum number of pages the page manager gets from the Stone in a single request. Must be equal to or greater than STN_PAGE_MGR_REMOVE_MIN_PAGES.

Runtime equivalent: #StnPageMgrRemoveMaxPages
Cache Statistic: PageMgrRemoveMaxPages (Stone)
Min: 1
Max: 16384
Default: 16384

STN_WELL_KNOWN_PORT_NUMBER

Port number that the Stone will use as its well-known port. The well-known port is used by all Gems while establishing their initial connection to the Stone during the login sequence.

If the specified port is in use by another process, the Stone will not start and exits with an error.

A value of zero indicates the port number will be selected by the system.

Min: 1
Max: 65535
Default: 0

Added Runtime Configuration Parameters

#GemConvertArrayBuilder

Allows old style Array Builder syntax #[a, b] to be parsed correctly; the compiler converts this syntax to the new form { a . b }, and updates method source as well as compiled code.

#GemDropCommittedExportedObjs

If this configuration parameter is turned on, clean, committed objects may be dropped from RAM. This reduces demand on memory in the gem, but there is the small cost of an additional bitmap lookup when the object is faulted, to detect if this object is in the Pure Export Set. By default, this configuration is false; it is set to true by sessions logging in from GBS, in which large number of objects may be replicated but not modified.

#GemExceptionSignalCapturesStack

When true, primitive 2022 (AbstractException>>_signalWith:) fills in the gsStack instance variable of the receiver, so that you can later send stackReport to the instance of AbstractException. See also AbstractException>>stackReport.Default for Smalltalk is false; default for Ruby is true.

Chapter 2

GCI Changes

GemStone/S 64 Bit version 3.0 introduces a small set of changes to the GCI interface. The most significant change is a set of new functions that include the environmentId argument. For more information, see "Environment Id" on page 17.

Added GCI functions

GciAllocTravBuf

```
(GciTravBufType *) GciAllocTravBuf(
   size_t allocationSize
   );
```

Allocate and initialize a new GciTravBufType structure.

GciCompileMethod

```
(OopType) GciCompileMethod(
   OopType source,
   OopType oclass,
   OopType category,
   OopType symbolList,
   OopType overrideSelector,
   int compileFlags,
   ushort environmentId
   );
```

GciCompileMethod is a replacement for GciInstMethodForClass and GciClassMethodForClass, with additional arguments.

GciDeclareAction

```
(void) GciDeclareAction(
   const char* name,
   void* func,
   int nargs,
   uint flags,
   BoolType errorIfDuplicate
   );
```

GciDeclareAction makes it easier to code calls to GciInstallUserAction.

GciFetchDynamicIv

```
(OopType) GciFetchDynamicIv(
   OopType obj,
   OopType aSymbol
   );
```

Return the value of the dynamic instance variable specified by `aSymbol`, or OOP_NIL if no such dynamic instance variable exists in the object `obj`.

GciFetchDynamicIvs

```
(int) GciFetchDynamicIvs(
   OopType obj,
   OopType *buf,
   int bufSizeOops
   );
```

The function result is the number of OOPs returned in `buf`. The number of dynamic instance variable pairs returned is (function result / 2). To obtain all dynamic instance variables in one call, use a buffer.

GciInstallUserAction_

```
(void) GciInstallUserAction_(
   GciUserActionSType *userAction,
   BoolType errorIfDuplicate
   );
```

Added variant of GciInstallUserAction with new argument errorIfDuplicate.

GciOopToChar32

```
(unsigned int) GciOopToChar32(
   OopType theObject
   );
```

Convert a Character object to a 32 bit C character value.

GciRealloc

```
(void*) GciRealloc(
   void *p,
   size_t length,
   int lineNumber,
   const char* fileName
   );
```

Return NULL if the underlying realloc() fails.

GciSetDynLib

```
(void) GciSetDynLib(
   void *handle
   );
```

Used by the topaz.c main program to save the result of dlopen() which loaded the GCI shared library.

GciStoreDynamicIv

```
(void) GciStoreDynamicIv(
   OopType obj,
   OopType aSymbol,
   OopType value
   );
```

Create or change the value of the dynamic instance variable specified by `aSymbol` within the object `obj`. Dynamic instance variables are not allowed in instances of ExecBlock, Behavior, GsNMethod, or special objects.

To delete a dynamic instance variable, pass OOP_REMOTE_NIL as the value.

GciSwapBytesUint

```
(void) GciSwapBytesUint(
   uint *buf,
   intptr_t numChars
   );
```

Swap the byte order of the specified array of uint.

GciSwapBytesUshort

```
(void) GciSwapBytesUshort(
   ushort *buf,
   intptr_t numChars
   );
```

Swap the byte order of the specified array of ushort.

Change to GciErrSType

The GemBuilder C structured type GciErrSType now includes an additional field: *exceptionObj*, which is either an instance of Exception or nil (if the error was not signaled from Smalltalk execution). For more details about GciErrSType, see "The Error Report Structure" in the *GemBuilder for C* manual.

```
...
OopType category
OopType context
OopType exceptionObj
...
```

Change to GciObjInfoSType

The GemBuilder C structured type GciObjInfoSType field segmentId has been renamed to objectSecurityPolicyId.

Renamed functions

The following functions have been renamed in version 3.0:

Old function name	Renamed to:
GciNbStoreTravDoTravRefs	GciNbStoreTravDoTravRefs_
GciNbStoreTravDoTrav	GciNbStoreTravDoTrav_
GciNbStoreTravDo	GciNbStoreTravDo_
GciStoreTravDoTravRefs	GciStoreTravDoTravRefs_
GciStoreTravDoTrav	GciStoreTravDoTrav_
GciStoreTravDo	GciStoreTravDo_

Removed functions

The following functions were present in previous releases, but are not present in version 3.0:

Old function name
GciGetCmdLine
GciGetCmdReader
GciGetRtlTable
GciSetCmdLine
GciSetCmdReader
GciUtlStrError

Functions created by new environmentId argument

Version 3.0 introduces a set of functions that are identical to existing GCI functions, but with two differences:

- Each of the new functions includes the environmentId argument.
- The name of the new function includes a trailing underscore.

The original functions (without the trailing underscore) remain available in version 3.0 and are unchanged.

Functions created by environmentId argument

New functions in version 3.0 (with environmentId argument)	Existing functions (without environmentId argument)
GciExecuteDbg_	GciExecuteDbg
GciExecuteFromContextDbg_	GciExecuteFromContextDbg
GciExecuteFromContext_	GciExecuteFromContext
GciExecuteStrDbg_	GciExecuteStrDbg
GciExecuteStrFromContextDbg_	GciExecuteStrFromContextDbg
GciExecuteStrFromContext_	GciExecuteStrFromContext
GciExecuteStrTrav_	GciExecuteStrTrav
GciExecuteStr_	GciExecuteStr
GciExecute_	GciExecute
GciNbExecuteDbg_	GciNbExecuteDbg
GciNbExecuteStrDbg_	GciNbExecuteStrDbg
GciNbExecuteStrFromContextDbg_	GciNbExecuteStrFromContextDbg
GciNbExecuteStrFromContext_	GciNbExecuteStrFromContext
GciNbExecuteStrTrav_	GciNbExecuteStrTrav
GciNbExecuteStr_	GciNbExecuteStr
GciNbExecute_	GciNbExecute
GciNbPerformNoDebug_	GciNbPerformNoDebug
GciNbPerformTraverse_	GciNbPerformTraverse
GciNbPerformTrav_	GciNbPerformTrav
GciNbPerform_	GciNbPerform
GciPerformNoDebug_	GciPerformNoDebug
GciPerformSymDbg_	GciPerformSymDbg
GciPerformTraverse_	GciPerformTraverse
GciPerformTrav_	GciPerformTrav
GciPerform_	GciPerform

Compile and Link Information

Compile and link has changed on all platforms. Below is the updated information for each platform. This material is also provided in the platform-specific Installation Guides and the GemBuilder for C manual.

Linux

Compiler version

g++ (GCC) 4.1.2 20070115 (prerelease) (SUSE Linux)

Debugger version

GNU gdb 6.6

Compiling a user action or GCI application

```
/usr/bin/g++ -fmessage-length=0 -fcheck-new -Wformat -Wtrigraphs
-Wcomment -Wsystem-headers -Wtrigraphs -Wno-aggregate-return
-Wswitch -Wshadow -Wunused-value -Wunused-variable -Wunused-label
-Wno-unused-function -Wchar-subscripts -Wmissing-braces
-Wmultichar -Wparentheses -Wsign-compare -Wsign-promo
-Wwrite-strings -Wreturn-type -Wuninitialized -Werror -O3 -ggdb
-m64 -pipe -D_REENTRANT -D_GNU_SOURCE -pthread -fPIC -m64
-fno-strict-aliasing -fno-exceptions -I$GEMSTONE/include -x c++
-c userCode.c -o userCode.o
```

Linking a user action

```
/usr/bin/g++ -shared -Wl,-Bdynamic,-hliboserAct.so userCode.o
$GEMSTONE/lib/gciualib.o -o liboserAct.so -m64 -lpthread -lcrypt
-ldl -lc -lm -Wl,-z,muldefs -Wl,--warn-unresolved-symbols
```

Linking a GCI application

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

Solaris on Sparc

Compiler version

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

Debugger version

Sun Dbx Debugger 7.5 Patch 121023-02 2006/05/26

Compiling a user action or GCI application

```
CC -xO4 -xcode=pic32 -m64 -mt -xchip=ultra2 -D_REENTRANT
-D_POSIX_PTHREAD_SEMANTICS  -I$GEMSTONE/include
-c usractOrApp.c -o usractOrApp.o
```

Linking a user action

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

Linking a GCI application

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

Solaris on x86

Compiler version

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

Debugger version

Sun DBX Debugger 7.7 SunOS_i386 2009/06/03

Compiling a user action or GCI application

```
CC -xO4 -m64 -Kpic -mt -D_REENTRANT -D_POSIX_PTHREAD_SEMANTICS
-I$GEMSTONE/include -c userCode.c -o userCode.o
```

Linking a user action

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

Linking a GCI application

```
CC -xildoff -m64 -i userCode.o $GEMSTONE/lib/gcirtlobj.o
-z nodefs -Bdynamic -lc -lpthread -ldl -lrt -lsocket -lnsl -lm
-lCrun -o userAppl
```

AIX

Compiler version

IBM XL C/C++ for AIX, V11.1 (5724-X13)
Version: 11.01.0000.0004

Debugger version

dbx

Compiling a user action or GCI application

```
/usr/vacpp/bin/xlC_r -O3 -qstrict -qalias=noansi -q64 -+
-D_REENTRANT -D_THREAD_SAFE -qpic -qthreaded -qarch=pwr5
-qtune=balanced -qminimaltoc -qmaxmem=-1
-qsuppress=1500-010:1500-029:1540-1103:1540-2907:1540-0804:1540-
1281:1540-1090 -qnoeh -I$GEMSTONE/include -c userCode.c
-o userCode.o
```

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

Linking a user action

```
/usr/vacpp/bin/xlC_r -G -q64 userCode.o $GEMSTONE/lib/gciualib.o
-o libuserAct.so -e GciUserActionLibraryMain -L/usr/vacpp/lib
-lpthreads -lc_r -lC_r -lm -ldl -lbsd -Wl,-berok
```

Linking a GCI application

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

HPUX on Itanium

Compiler version

aCC: HP C/aC++ B3910B A.06.25.02 [Nov 25 2010]

Debugger version

HP gdb 6.2 for HP Itanium (32 or 64 bit) and target HP-UX 11iv2 and 11iv3.

Compiling a user action or GCI application

```
/opt/aCC/bin/aCC +O2 +Onolimit +Z +DD64 +DSitanium2 -Aa -D_PSTAT64
-D_LARGEFILE64_SOURCE -mt -Wl,+vnocompatwarnings
+W212,749,740,863,2225,2175,2177,4232,4189,4070,20011,20009,2368
-D_HPUX -D_POSIX_C_SOURCE=199506L -D_HPUX_SOURCE
-D_INCLUDE_LONGLONG -D_XOPEN_SOURCE=600 -D_XOPEN_SOURCE_EXTENDED=1
+We -I$GEMSTONE/include -c -S userCode.c -o userCode.o
```

Linking a user action

```
/opt/aCC/bin/aCC -v +DD64 +DSitanium2
-Wl,+allowdups,+k,+n,+pd4M,+pi64K -v -b -Wl,-B,symbolic -z user-
Code.o $GEMSTONE/lib/gciualib.o -o libuserAct.sl
-Wl,+e,GciUserActionLibraryMain -lxnet -lrt -lsec -ldld -lm
-l:libcres.a -lc -lCsup -lunwind -Wl,+allowunsats,+vnoshlibunsats
```

Linking a GCI application

```
/opt/aCC/bin/aCC -v +DD64 +DSitanium2
-Wl,+allowdups,+k,+n,+pd4M,+pi64K -z userCode.o
$GEMSTONE/lib/gcirtlobj.o -Wl,+allowunsats,+vnoshlibunsats
-lxnet -lrt -lsec -ldld -lm -l:libcres.a -lcl -lpthread -o userAppl
```

Darwin

Compiler version

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

Debugger version

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

Compiling a user action or GCI application

```
/usr/bin/g++ -fmessage-length=0 -fcheck-new -Wformat -Wtrigraphs
-Wcomment -Wsystem-headers -Wtrigraphs -Wno-aggregate-return
-Wswitch -Wshadow -Wunused-value -Wunused-variable -Wunused-label
-Wno-unused-function -Wchar-subscripts -Wconversion
-Wmissing-braces -Wmultichar -Wparentheses -Wsign-compare
-Wsign-promo -Wwrite-strings -Wreturn-type -O3 -ggdb -m64 -pipe
-D_XOPEN_SOURCE -D_REENTRANT -D_GNU_SOURCE -fPIC -m64
-fno-strict-aliasing -I$GEMSTONE/include -x c++ -c userCode.c
-o userCode.o
```

Linking a user action

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

Linking a GCI application

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

Windows

Compiler/Debugger version

Microsoft Visual Studio 2008 Version 9.0.21022.8 RTM

Microsoft Visual C++ 2008 91605-270-6514982-60495

Compiling a GCI application

```
cl /W3 /MD /Zi /TP /nologo /DWIN32 /D_CONSOLE /D_DLL /DNATIVE
/I 'VisualStudioInstallPath\VC\atlmfc\include'
/I 'VisualStudioInstallPath\VC\include'
/I 'C:\Program Files\Microsoft SDKs\Windows\v6.0A\Include'
/I%GEMSTONE%\include -c userCode.c -FouserCode.obj
```

Linking a GCI application

```
link /LIBPATH:'VisualStudioInstallPath\VC\lib'
/LIBPATH:'VisualStudioInstallPath\VC\atlmfc\lib'
/LIBPATH:'C:\Program Files\Microsoft SDKs\Windows\v6.0A\Lib'
-INCREMENTAL:NO -nologo 'userCode.obj' '%GEMSTONE%\lib\gcirpc.lib'
wsock32.lib netapi32.lib advapi32.lib comdlg32.lib user32.lib
gdi32.lib kernel32.lib winspool.lib -out:userAppl.exe
```

Chapter 3

Changes in Topaz

In GemStone/S 64 Bit version 3.0, Topaz has had many updates and new commands. Many of these new commands are for debugging, including debugging of multi-threaded sessions, and for enhanced support for development and debugging in Ruby.

Ruby and environmentId

Many of the changes in Topaz are specific to working in Ruby images, and are not applicable in a Smalltalk-only environment.

In addition, some Topaz commands now also accept an optional environmentId argument. This argument is not needed in the Smalltalk environment, although it may be used if multiple method environments are implemented. For more details, see "Environment Id" on page 17.

Use of .topazini

The headers printed when invoking topaz now include a message regarding the use of login initialization file, `.topazini`. If a login initialization file is found in the current directory, the user's home directory, or passed in using the -I option, the name and location of the file is displayed; otherwise the message "`neither .topazini nor $HOME/.topazini were found`" is displayed.

Line editor

Topaz now supports line editing, using the open source linenoise library.

The line editor is enabled by default. You can disable it using OMIT LINEEDITOR and reenable using DISPLAY LINEEDITOR. When the line editor is enabled, you can use HISTORY to view the list of previously executed commands.

The line editor functions are not available on Windows.

Display level changes

Topaz display level controls the level of instance variables to display when displaying an object (see the LEVEL command).

Previously, the default level when starting up topaz was 1; executing statements using PRINTIT or RUN displayed the resulting object and its instance variables and their values. Now, the default is 0, which will show only the object itself.

PRINTIT now always displays with a level of 1, regardless of the current display level; statements evaluated using PRINTIT with default level have no changes.

RUN, which was previously deprecated, has been undeprecated. It retains its previous behavior, to display the results using the current display level.

DOIT, which previously returned a statement such as "The OOP of the object is NNN", now displays results with a level of 0, regardless of the current display level.

Report on output truncated by LIMIT

Now, when output is truncated due to LIMIT commands such as LIMIT BYTES, the output reports the number of bytes truncated.

Removed Topaz command

The long-obsolete OPAL command has been removed.

New Topaz commands

This section introduces several new Topaz commands that are useful in exploring and debugging your Smalltalk code. For details about additional Topaz commands, use the Topaz HELP command, or refer to the *GemStone/S 64 Bit Topaz Programming Environment* manual.

EXITIFNOERROR

If ERRORCOUNT would return 0, same as EXIT 0. Otherwise has no effect.

FR_CLS [*anInt*]

Similar to FRAME, but turn on DISPLAY CLASSOOPS (page 125) for display of the specified stack frames.

HIERARCHY [*className*]

Print the class hierarchy up to Object for the specified class. If you don't specify a class, Topaz prints the hierarchy for the current class.

HISTORY [*anInteger*]

Print the last *anInteger* entries in the Topaz line editor history. Has no effect if the line editor is not enabled. (Not available on Windows.)

IMPLEMENTORS *selectorSpec*

Expects one argument, a *selectorSpec*, the value of which is either a String or Symbol. Equivalent to a DOIT of

```
ClassOrganizer new implementorsOfReport: aString
```

which reports implementors of (aString asSymbol). May require larger than default GEM_TEMPOBJ_CACHE_SIZE configuration.

INSPECT [*anObjectSpec*]

Expects one argument, an object specification. Equivalent to SEND *anObjectSpec* `describe`.

INTERP

Same as RUN, but do the execution in interpreted mode.

LISTW

For the method implied by the current stack frame, list a maximum of *windowSize* lines (as defined by SET LISTWINDOW), centered around the current insertion point for the frame.

If Topaz has a valid session, then LISTW can be abbreviated as L.

LOGOUTIFLOGGEDIN

If logged in, logs out the current GemStone session. If there is no current session, does not increment the Topaz error count.

OBJ1 *anObjectSpec*
OBJ2 *anObjectSpec*

Equivalent to the following series of Topaz commands:

`LEVEL 1` or `LEVEL 2` (depending on command)
`OMIT ZEROBASED`
`OBJECT` *anObjectSpec*
`LEVEL` *previousLevel*
`DISPLAY ZEROBASED` (if previously set)

For details about these Topaz commands, use the Topaz HELP command, or refer to the *GemStone/S 64 Bit Topaz Programming Environment* manual.

OBJ1Z *anObjectSpec*
OBJ2Z *anObjectSpec*

Equivalent to the following series of Topaz commands:

`LEVEL 1` or `LEVEL 2` (depending on command)
`DISPLAY ZEROBASED`
`OBJECT` *anObjectSpec*
`LEVEL` *previousLevel*
`OMIT ZEROBASED` (if not previously set)

For details about these Topaz commands, use the Topaz HELP command, or refer to the *GemStone/S 64 Bit Topaz Programming Environment* manual.

PKGLOOKUP (**METH** | **METHOD** | **CMETH** | **CMETHOD**) *selectorSpec*
PKGLOOKUP *classname* **[CLASS]** *selectorSpec*

Similar to LOOKUP (page 126), but with one key exception: PKGLOOKUP looks first in GsPackagePolicy state, then in the persistent method dictionaries for each class up the hierarchy. PKGLOOKUP does not look at transient (session method) dictionaries.

SENDERS *selectorSpec*

Expects one argument, a *selectorSpec*, the value of which is either a String or Symbol. Equivalent to a DOIT of

```
ClassOrganizer new sendersOfReport: aString
```

which reports senders of (aString asSymbol). May require larger than default GEM_TEMPOBJ_CACHE_SIZE configuration.

STRINGS *selectorSpec*

Expects one argument, a *selectorSpec*, the value of which is either a String or Symbol. Equivalent to a DOIT of

```
ClassOrganizer new strings: aString
```

which reports methods containing *aString* in their source string. May require larger than default GEM_TEMPOBJ_CACHE_SIZE configuration.

SUBCLASSES [*className*]

Prints subclasses of the specified class. If you don't specify a className, prints subclasses of the current class.

THREAD [*anInt*] [**CLEAR**]

Displays the currently selected GsProcess from among the stack saved from the last error, or those retrieved by last THREADS command.

THREAD *anInt* changes the currently selected GsProcess. You can specify an integer value from among those shown in the most recent THREADS command.

THREAD *anInt* CLEAR clears the selected GsProcess from the Topaz stack cache.

THREADS [**CLEAR**]

With no parameters, forces any dirty instances of GsProcess cached in VM stack memory to be flushed to object memory. Then executes a message send of

```
ProcessorScheduler>>topazAllProcesses
```

and retrieves and displays the list of processes.

THREADS CLEAR clears the Topaz cache of all instances of GsProcess.

Changes to existing Topaz functions

BREAK

New arguments to BREAK allow you to establish breakpoints at method step points.

BREAK *@aNumber*
establishes a breakpoint at the specified step point of the method selected by the previous DOWN, FRAME, LOOKUP, LIST, or UP command. For example: `BREAK @6`

BREAK METHOD *@anObjectSpec methodName*
establishes a breakpoint at step point 1 of the specified instance method for the class with the specified objectId.

BREAK METHOD *@anObjectSpec*
establishes a breakpoint at step point 1 of the GsNMethod with the specified objectId.

DISPLAY
OMIT

If DISPLAY OOPS is set, then DISPLAY CLASSOOPS or OMIT CLASSOOPS respectively enable and disable the display of OOPs of classes along with class names in object display. DISPLAY CLASSOOPS also causes the OOPs of classes to be printed by stack display and method lookup commands, and enables the printing of evaluation temporary objects in stack frame printouts from the FRAME command.

DISPLAY LINEEDITOR and OMIT LINEEDITOR enable and disable the use of the new Topaz line editor, respectively. Not available on Windows.

DISPLAY PUSHONLY and OMIT PUSHONLY enable and disable the effect of the ONLY keyword in an OUTPUT PUSH command, respectively.

DISPLAY ZEROBASED and OMIT ZEROBASED shows offsets of instance variables as zero-based or one-based, respectively, when displaying objects. (By default, offsets are one-based.)

DOIT

Previously, DOIT performed the operation and returned a string of the form "result oop is NNNN". Now, DOIT displays results as if the LEVEL was 0.

EXIT
QUIT

QUIT and EXIT now take an additional argument that is used to provide an explicit exitStatus for the Topaz process. Either a SmallInteger or an object specification that resolves to a SmallInteger is accepted. If no argument is specified, the exitStatus will be 1 if there was a GCI error or if the Topaz errorCount was nonzero, or 0 if no errors occurred during Topaz execution.

LIST

New and enhanced arguments to LIST provide additional functionality.

LIST*@anObjectSpec*
lists the source code of the GsNMethod or ExecBlock with the specified objectId. That method, or the block's home method, becomes the default method for subsequent LIST or DISASSEM commands.

LIST CCATEGORIES: [*className*]
lists all of the class method selectors for the named class, by category. If you do specify a class name, that class becomes the current class for subsequent Topaz commands. If you do not specify a class name, list the categories of the current class.

LIST CLASSMETHOD: [*@anObjectSpec* | *selectorSpec*]
With an argument of the form *@objectSpec*, lists the category and source of the method with that objectSpec.

With an argument that is a *selectorSpec*, lists the category and source of the given class method selector for the current class.

LIST CMETHOD:
is equivalent to LIST CLASSMETHOD:.

LIST CSELECTORS
lists selectors of all class methods. May be abbreviated as LIST SEL.

LIST ICATEGORIES: [*className*]
lists all of the instance method selectors for the named class, by category. If you do specify a class name, that class becomes the current class for subsequent Topaz commands. If you do not specify a class name, list the categories of the current class.

LIST IMETHOD:
is equivalent to LIST METHOD:.

LIST METHOD: [*@anObjectSpec* | *selectorSpec*]
With an argument of the form *@objectSpec*, lists the category and source of the method with that objectSpec.

With an argument that is a *selectorSpec*, lists the category and source of the given instance method selector for the current class.

LIST SELECTORS
lists selectors of all instance methods. May be abbreviated as LIST CSEL.

LOOKUP (**METH** | **METHOD** | **CMETH** | **CMETHOD**) *selectorSpec*

In previous releases, LOOKUP allowed you to search for a specified instance or class method, going up the hierarchy of the current class. Two keyword variations were available: `method` and `classmethod`. In GemStone/S 64 Bit version 3.0, you can also use the abbreviated keywords `meth` and `cmeth`, respectively. (The value of *selectorSpec* is either a String or Symbol.)

LOOKUP *classname* **[CLASS]** *selectorSpec*

You can now search the hierarchy of any class (not just the current one). Using this syntax, make the specified class the current class, and do the specified lookup for a instance method (if the keyword CLASS is specified, do the lookup for a class method).

PRINTIT
RUN

PRINTIT does not use the current display level setting, and always displays the result as if LEVEL 1 were the most recent LEVEL command. It does not alter the current level setting.

RUN previously was described as deprecated; it is no longer deprecated. The behavior is unchanged; RUN uses the current setting for level.

SET

SET HISTORY *anInteger* sets the history size of the line editor. Not available on Windows.

SET LISTWINDOW *anInteger* defines the maximum number of source lines to be listed by the LISTW command (page 123).

SET STACKPAD *anInteger* defines the minimum size used when formatting lines in a stack display. The argument *anInteger* may be between 0 and 256, inclusive. (Default: 45.)

SET TAB *anInteger* defines the number of spaces to insert when translating a tab (CTRL-I) character when printing method source strings. The argument *anInteger* may be between 1 and 16, inclusive. (Default: 8.)

WHERE

WHERE now accepts a string argument. The string must not begin with a decimal digit, whitespace, or any of the three characters (' + -), and must not contain whitespace. For example:

```
WHERE aString
```

If you wish to specify a string that contains digits or whitespace characters, enclose it in single-quotes. For example:

```
WHERE 'bString'
```

WHERE searches all frames in the current stack, and displays only those for which output of WHERE for that frame matches a case-sensitive search for *aString* anywhere in that frame's output (not including the frame number or ==> marker at the start of the frame's line). The current frame is set to the first frame matched by the search.

Chapter 4

Bugs Fixed

The following bugs in GemStone/S 64 Bit 2.4.4.6 have been fixed in GemStone/S 64 Bit 3.0.

Shared cache monitor lock file regeneration

If GemStone process lock files (*filename*`..LCK`) are deleted, normally they are recreated within a minute. The shared page cache monitor lock previously did not regenerate itself; now it will do so. (#41521)

Chance of duplicate keys for shared memory segments

GemStone uses ftok() to acquire a key for a shared memory segment, which is used by all processes that need to attach to that segment. Since ftok() uses only 16 bits of the inode, as well as the device and constant, there was a risk that the shared memory key could be created for two repositories running on the same host. This requires multiple repositories on the same host, and chances were exacerbated by running with multiple lock directories (GEMSTONE_GLOBAL_DIR settings).

To reduce the risk, instead of the constant, ftok() is passed a random value from /dev/random. The resulting key is stored in the SPC monitor lock file. (#41501)

SEGV due to duplicate free memory call after network problem encountered

When the page manager removes pages from remote shared page caches, there is a code path in which, if a call fails (for example, due to a network error), the buffer used to communicate to the remote caches is freed a second time. This cause the page manager to SEGV, and therefore shutdown the system. (#41184)

The Page Manager code has also been modified so that all pages are retried if any network error is detected while talking to any remote cache. Previously code paths existed where pages may not have been retried. Not retrying risks a race condition with the Stone where a page could be reported ready for disposal before the stone has terminated all gems on the dead remote cache.

The number of times that all pages had to be retried due to a network error or timeout with a remote SPC is now reported in the Page Manger cache statistic SessionStat4.

Uninitialized block temporaries retained values over iterations

For example:

```
| a |
a := Array new .
1 to: 3 do:[:j | | b |
   a add: b.
   b := j.
   a add: b.
].
^ a printString
```

In previous releases, this code returned `anArray( nil, 1, 1, 2, 2, 3).` In version 3.0, it returns `anArray( nil, 1, nil, 2, nil, 3).` (#38112)

Incorrect nested block variable scoping

Previously, the scope of variables within nested blocks was not correct. For example, the following code:

```
| a b |
a := Array new: 4.
b := Array new: 4.
1 to: a size do: [:i |
   b at: i put: [10 timesRepeat: [a at: i put: i]]
   ].
1 to: b size do: [:j | (b at: j) value].
^a
```

This code incorrectly returned `anArray( nil, nil, nil, 4)` in previous versions.

In v3.0, the correct answer, `anArray( 1, 2, 3, 4),` is now returned. (#38359)

Statmonitor could corrupt output or crash when stone started with -r option

If the Stone is killed and restarted while statmonitor is running with the -r option, statmonitor may write final output from the previous cache to the beginning of the next statmonitor data file, before writing the header data. This resulted in an statmonitor file that cannot be read by VSD. It was also possible for statmonitor to crash with a SEGV in this circumstance. (#40880)

LoadAverageStat not handled by VSD

The cache statistics LoadAverageStat was not handled correctly by VSD, resulting in a very large, useless number. (#41420)

Signals re-enabled timeout disabled via disableStoneGemTimeout

The method disableStoneGemTimeout allows a gem to perform a long running operation and not be subject to STN_GEM_TIMEOUT. However, receiving signals such as #rtErrSignalAlmostOutOfMemory, #rtErrSignalAbort, #rtErrSignalGemStoneSession,

and other signals re-enabled the timeout, leaving the gem subject to to termination due to the STN_GEM_TIMEOUT. (#41530)

Tranlog full issues

Extensive testing on transaction log full conditions has been done, and a number of error conditions and inconsistencies in behavior have been fixed. (#41129 and others)

Now, administrative uses can log in; their login, logout and abort records are queued until they can be written to tranlog. Also, on tranlog full, sigAborts are not sent to non-administrative uses, since these users cannot complete an abort. When tranlogs are full, stopstone results in an unclean shutdown.

No information provided for errors on extent pregrow on startup

On startup, extent files may be pre-grown if configured to do so. Failures in this pre-grow, for example due to insufficient disk space, did not report useful errors. (#41525)

Slow performance with heavy use of permGen memory space

The permGen memory space holds classes, and when there are repeated iterations through a large number of classes, the permGen space may experience excessive GC rather than growing to accommodate the load. (#40921)

Inconsistent stone name quoting

Informational messages in logs sometimes put the Stone name in single quotes, other times in double quotes. Single quotes are now used throughout (#40307)

Uninformative error on repository startup failure

Some causes of Stone startup failure may report "Repository not attached", which did not indicate any cause of the problem. This has been improved and most startup conditions should now return a useful error. (#40801)

System forceEpochGc does not increment EpochGcCount

Invoking System class >> forceEpochGc forces an Epoch to start, but did not increment the statistics EpochGcCount. (#41534)

Encoded size ByteArray comparison method primitive failures

Using the primitives called by ByteArray>>compareStringAt:to:startingAt:sizeBytes:useCase: or ByteArray>>shortStringAt:compareWith:startingAt:opCode: with an index other than 1 may have encountered primitive failures (#41341)

removing indexes did not disableStoneGemTimeout

To avoid gems performing long-running indexing operations from termination due to STN_GEM_TIMEOUT, index creation invokes disableStoneGemTimeout. Now, index removal operations also invoke this code, to allow long-running index removal tasks to complete. (#41484)

Risk of garbage when UserProfile removed

When a new user is created, some of the creation methods create and commit a new Segment (now ObjectSecurityPolicy) for that user. When the user is deleted by removal from AllUsers, this Segment remains owned by the user. This reference keeps the UserProfile and all data specific to that UserProfile alive. (#40819)

Dynamically created extents may not get page reclaim

If Reclaim GcGems are specified for existing extents, and a new extent is added programmatically, the newly created extent will not have an assigned Reclaim GcGem and pages in this extent will not get reclaimed (#40859)

Now, a reclaim GcGem is started by default when an extent is added. A new method has been added, `Repository>>createExtent:withMaxSize:startNewReclaimGem:` that can be used for more specific control.

Number of caches limited to 1024

Internal code limits on the number of network connections resulted in the maximum number of shared page caches being limited to 1024, if the actual configured limit was higher. (#40823)

Race condition in page manager handling of dead remote caches

The sequence of recycling pages and the detection of a dead remote cache was not correct, allowing the pages returned by Gems on a remote cache that had timed out to potentially be reused before the Gems were terminated. This resulted in cache coherency errors in Gems on the dead remote cache. (#40746)

Inefficient code building hidden sets

Some operations to retrieve the size of hidden sets constructed a copy of the hidden set in the process. This has been optimized where possible to avoid constructing a copy. (#41463)