GemStone®

GemStone/S 64 Bit Porting Guide

for GemStone/S 6.x Applications

Version 2.2.1

June 2007

INTELLECTUAL PROPERTY OWNERSHIP

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

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

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

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

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

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

COPYRIGHTS

PATENTS

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

TRADEMARKS

GemStone, **GemBuilder**, **GemConnect**, and the GemStone logos are trademarks or registered trademarks of GemStone Systems, Inc. in the United States and other countries.

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

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

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

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

Microsoft, **MS**, **Windows**, **Windows 2000** and **Windows XP** are registered trademarks of Microsoft Corporation in the United States and other countries.

Linux is a registered trademark of Linus Torvalds and others.

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

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

Other company or product names mentioned herein may be trademarks or registered trademarks of their respective owners. Trademark specifications are subject to change without notice. All terms mentioned in this documentation that are known to be trademarks or service marks have been appropriately capitalized to the best of our knowledge; however, GemStone cannot attest to the accuracy of all trademark information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

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

Preface

About This Manual

This manual addresses issues in porting applications from GemStone/S version 6.1.5, the 32-bit GemStone object server, to GemStone/S 64 Bit version 2.2.1. While this manual addresses the major changes to consider when porting, it does not attempt to describe every change or difference in behavior, and does not document every new feature in detail.

This manual is intended for users who are familiar with GemStone/S 6.1.5.

GemStone/S 64 Bit is similar in function to GemStone/S, while scaling to handle much larger and more demanding applications. In addition to the architectural changes allowed by the 64-bit environment, many subsystems have been redesigned for performance and scalability.

Significant internal changes include the port to 64-bit and rearchitecting of memory. These changes are largely transparent to the user, and are described at a high level in this document. These changes significantly affect configuration and tuning; new features and functions provide greater ability to tune for optimal performance.

Some critical subsystems, such as garbage collection, have been redesigned to be more robust and scalable. GemStone/S 64 Bit also introduces new features, such as the Page Manager Gem, to address specific bottlenecks in the system. This document describes the changes; for more detail, refer to the GemStone/S 64 Bit documentation.

Terminology Conventions

In this document, the term “GemStone” is used to refer both to the server products GemStone/S 64 Bit or GemStone/S, and to the company, GemStone Systems, Inc.

Other Useful Documents

This document provides a summary of changes between GemStone/S and GemStone/S 64 Bit. For more detail on the current GemStone/S 64 Bit behavior, you will find it useful to look at the following documents:

- The *System Administration Guide for GemStone/S 64 Bit* describes how to administer the GemStone server.
- The *GemStone/S 64 Bit Programming Guide* provides useful information for developing applications in GemStone/S 64 Bit.
- The GemStone image contains comments for Classes and Methods that describe the behavior in greater detail.
- The *GemStone/S 64 Bit Topaz Programming Environment Manual* describes Topaz, a scriptable command-line interface to GemStone Smalltalk. Topaz is most commonly used for performing repository maintenance operations.
- The *GemBuilder for Smalltalk* manual describes GemBuilder for Smalltalk, a programming interface that provides a rich set of features for building and running client Smalltalk applications that interact transparently with GemStone Smalltalk.
- The *GemBuilder for C* manual describes GemBuilder for C — a set of C functions that provide a bridge between your application's C code and the application's database controlled by GemStone.

Technical Support

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

GemStone Web Site: http://support.gemstone.com

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

The following types of information are provided at the GemStone Technical Support website:

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

Documentation for GemStone products are provided in PDF format.

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

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

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

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

Community provides customer forums for discussion of GemStone product issues.

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

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

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

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

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

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

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

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

Email: support@gemstone.com

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

Requests for technical assistance may also be submitted by email or by telephone. We recommend you use telephone contact only for more serious requests that require immediate evaluation, such as a production system that is non-operational. In these cases,

please also submit your request via the web or email, including pertinent details such error messages and relevant log files.

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

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

24x7 Emergency Technical Support

GemStone offers, at an additional charge, 24x7 emergency technical support. This support entitles customers to contact us 24 hours a day, 7 days a week, 365 days a year, if they encounter problems that cause their production application to go down, or that have the potential to bring their production application down. For more details, contact your GemStone account manager.

Training and Consulting

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

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

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

Contents

Chapter 1. Architecture

Chapter 2. Garbage Collection

Chapter 3. Backup and Restore

Chapter 4. System Administration Changes

Chapter 5. Changes That Affect Application Code

Chapter 6. GemBuilder for C

Chapter 7. Topaz Changes

Chapter 8. Changes in Configuration Parameters

Chapter 9. Changes in Errors

Chapter 10. Changes in Cache Statistics

Chapter

1 Architecture

This chapter summarizes the architectural differences between GemStone/S 6.1.5 and GemStone/S 64 Bit version 2.2.1.

While GemStone/S is similar in many ways to GemStone/S 64 Bit, there are significant architectural changes. In addition to the physical limits such as 32-bit address space, field experience in how GemStone/S applications scale has indicated other areas in which GemStone/S failed to scale well. Many of these issues have been addressed in GemStone/S 64 Bit.

Architectural Overview

GemStone/S 64 Bit internal server code has been redesigned and recompiled to run on 64-bit processors using 64-bit virtual addresses. This removes the maximum upper limit of 4 GB on shared page cache (SPC) size. While the new theoretical limit of SPC size is 16 terabytes (16384 GB), version 2.2.1 has been tested and is fully certified with caches up to 16 GB in size.

GemStone/S 64 Bit has also been modified to use 64-bit OOPs, avoiding the 1 billion object limit of 32-bit OOPs. The new upper limit on the number of objects is 2^{40} - 1 (1,099,511,627,775); sizing of internal structures currently limits the number of OOPs to 2^{37} - 1 (137,438,953,471). The maximum size of an object is also 2^{40} - 1. Since the OOPs are larger, GemStone/S repositories will typically grow between 25% and 40% during conversion.

As part of this change, OOP formats and reserved OOP numbers have all changed. For more about this, see “OOP Format” on page 12.

Page size is increased from 8K to 16K (16384) bytes.

The maximum number of extents remains 255. However, the maximum number of pages per extent has increased from 2^{23}-1 to 2^{31}-1.

The Virtual Machine (VM) has been redesigned in GemStone/S 64 Bit to use a copy-on-read architecture. In GemStone/S, the VM accessed objects in the shared page cache (SPC) directly, by attaching pages. In a copy-on-read architecture, the first access to a persistent (POM) object by a VM copies that object from the SPC into the VM's private memory. Subsequent accesses to that object do not involve the SPC, so the page does not need to be attached.

Note that the maximum size of a compiled method in memory is now 65535 bytes. Very large methods must be split prior to upgrading. For more information on detecting very large methods in GemStone/S, see the *GemStone/S 64 Bit Installation Guide*.

GemStone/S 64 Bit also introduces the following performance improvements:

- The Stone is multi-threaded.
- The shared page cache monitor is multi-threaded. Slot recovery is done in a separate thread for improved response.
- The out of band socket is multi-threaded.
- Polymorphic method lookup caches have been added. Method lookups to `super` and `send` that produce do-not-understand have been optimized.
- With the multithreaded Stone, all Gems close their in-band socket to Stone after establishing either SMC communication or Pgsvr-SMC communication to Stone.

OOP Format

OOP formats have changed. The new tag bits for OOPs are:

`2r000`	RAM OOP (memory pointer)
`2r001`	PomObjId - disk object IDs have the form `0x0000nnnnnnnnnn01`, with the oopNumber shifted left by 8 before adding the pom tag bit
`2r010`	SmallInteger
`2r110`	SmallDouble
`2r100`	Other specials: true, false, nil, Char, JISChar

All reserved OOPs and tag bits have been renumbered.

To convert a GemStone/S 6.1.5 POM OOP to the equivalent OOP in GemStone/S 64 Bit 2.2.1, the formula is

gs64OOP `= (64 *` *gssOOP*`) - 63`

Special Objects

SmallInteger range change

In GemStone/S 6.1.5, SmallIntegers were 30 bits, with a range of:

$-(2^{29})$ to $(2^{29} - 1)$

or

`-536,870,912` to `536,870,911`

In GemStone/S 64 Bit, SmallIntegers are now 61 bits, with a range of:

$-(2^{60})$ to $(2^{60} - 1)$

or

-1,152,921,504,606,846,976 to 1,152,921,504,606,846,975

Note that during conversion, instances of LargePositiveInteger and LargeNegativeInteger in GemStone/S that are within the new GemStone/S 64 Bit range of SmallIntegers are **not** converted into SmallIntegers. This avoids any potential problems with hashes and references. You can use special flags during the conversion process to collect all references to LargeIntegers, which allows you to manually convert them. For more information, see the *GemStone/S 64 Bit Installation Guide.* Adding zero to a Large Integer that is within the SmallInteger range will perform the conversion.

Due to changes in the range of SmallIntegers and Floats, it is now possible to lose precision when converting from SmallInteger to Float and back.

SmallDouble replaces SmallFloat

The class SmallFloat (representing 4-byte floating point numbers) is deprecated in GemStone/S 64 Bit. The new class SmallDouble should be used instead. SmallDoubles are special objects; that is, they are canonical and do not require separate OOPs. Most non-Integer numeric operations now return instances of Float or SmallDouble. This includes implementations of `asSmallFloat`.

SmallDouble represents 8-byte binary floating point numbers, as defined in IEEE standard 754, but with a reduced exponent. SmallDouble has 8 bits of exponent, compared to 11 bits of exponent in an IEEE-754 8-byte float. For numbers that require more than 8 bits of exponent, the VM automatically converts the number to an object of class Float.

Each SmallDouble contains a 61-bit value. The floats are stored on disk and in object memory in big-endian IEEE format. GemStone for Smalltalk primitives and GemBuilder for C (GCI) float conversion functions automatically convert the format of a float to or from the machines' native format, as required.

SmallDoubles can represent C doubles that have exponent bits in range 16r380 to 16r3FE, which corresponds to about 5.0e-39 to 3.0e+38, approximately the range of C float.

Memory Use

Process memory has been significantly reorganized in GemStone/S 64 Bit. The memory footprint of all processes is necessarily larger. In addition, bitmaps and shadowed objects now are kept in memory structures, rather than in the private page cache. This reduces the demand on the private page cache, so the private page cache can be sized considerably smaller in GemStone/S 64 Bit.

In GemStone/S 64 Bit, local objects do not overflow into the shared page cache via the NotConnectedSet. The NotConnectedSet no longer exists in GemStone/S 64 Bit. This avoids the problem of objects inadvertently becoming committed. However, it presents a significant risk if there is not enough temporary object cache; if the temporary object cache is exhausted, the Gem will encounter an out-of-memory condition and terminate. This is

particularly likely to be a problem if there are long transactions that modify or create a large number of objects.

Configuring the correct amount of temporary object cache is an important step in moving from GemStone/S 6.1.5 to GemStone/S 64 Bit. To avoid out-of-memory errors, you may need to modify your application. The default temporary object cache size is 10 MB; up to 1 GB of temporary object cache may be configured.

One option to avoid out-of-memory issues is to configure each gem to use a very large temporary object cache. On Solaris and Linux, you can specify larger temporary caches than needed without wasting memory; on these platforms, the memory is reserved but is only allocated as needed. On AIX and HP-UX, however, memory is allocated immediately, and becomes unavailable to other processes. To avoid this problem, you can use the new configuration option GEM_TEMPOBJ_INITIAL_SIZE (page 60) to specify an initial size, in addition to the maximum specified by GEM_TEMPOBJ_CACHE_SIZE, and the cache will be enlarged as needed.

The tuning and special processing that were required to avoid NotConnectedSet problems are no longer beneficial. The method `System class >> _markNotConnectedForCollection` is no longer relevant, and has been removed from the image, as have NotConnectedSet related cache statistics.

To further reduce the memory footprint of Gems, the maximum number of network connections has been reduced to 1000. This does not apply to the Stone or shared page cache monitor. This affects the maximum number of connections active from a particular process, and the maximum number of active GsFile or GsSocket instances in a Gem.

In linked Topaz, the combined total of GsFile, GsSocket, and RPC sessions may be limited to 1000. In RPC Topaz, the total number of client side GsFile plus RPC sessions may be limited to 1000.

To reduce memory footprint for certain types of processes, processes are loaded differently in GemStone/S 64 Bit. The gem is now a small executable that loads the `gcilnk.so` shared library, and pgsvrmain is now a small executable that loads `libpgsvr.so`. Statically linked versions of these executables (geml and pgsvrmainl) are still available for debugging.

In Virtual Machine garbage collection, stubbing has been added to the scavenge operation. Except for certain special objects used by the VM, references from temporary to clean committed objects will be stubbed by scavenge. Exceptions include references from kinds of VariableContext, BlockClosure, ClassHistory, SessionState, ExportSet, and TrackedSet.

For more about in-memory garbage collection, and managing the size of temporary object memory, including a description of the methods used to track the load on temporary object memory, see the “Managing Growth” chapter of the *System Administration Guide for GemStone/S 64 Bit*.

Signal on almost out of memory

In GemStone/S 64 Bit, the exception `#rtErrSignalAlmostOutOfMemory` (error 6013) can be signalled when a session’s temporary object memory is almost full. This signal is asynchronous, and is similar to the sigAbort mechanism.

For a discussion of the methods that make use of this exception, see “Signal on low memory condition” in the “Tuning Performance” chapter of the *GemStone/S 64 Bit Programming Guide*.

Debugging and the GS_DEBUG Environment Variables

In addition to gemnetobject, GemStone/S 64 Bit provides the script

```
$GEMSTONE/sys/gemnetdebug
```

which includes settings for the various environment variables beginning with GS_DEBUG, and other debugging aids. The intent is that you can use gemnetobject for production, and gemnetdebug for debugging, so that you can easily switch login parameters. You can edit gemnetdebug as needed for debugging.

Note that GemStone/S 64 Bit environment variables beginning with GS_DEBUG are subject to change. The information in the *System Administration Guide for GemStone/S 64 Bit* may not be correct between product releases. For accurate information for each release, you should always refer to `$GEMSTONE/sys/gemnetdebug`.

New Processes and Process Changes

GemStone/S 64 Bit introduces two new processes: Page Manager Gem and SymbolGem. These processes run along with the Stone, shared page cache monitor, and other GemStone processes. In addition, some GemStone/S processes behave somewhat differently in GemStone/S 64 Bit.

Page Manager Gem

When pages need to be disposed, it is necessary to communicate with all remote caches in order to ensure that the pages are not in use anywhere, before the pages are actually made available for reuse. In GemStone/S, analysis of production data showed that coordination with the remote caches was a bottleneck in the system, causing significant reduction in commit performance.

The Page Manager Gem addresses this problem by taking the list of potential pages to be disposed, and checking if any of them are in use in the local cache. It then communicates with each of the remote shared cache page servers (if any), which provide the same information for the remote caches. The set of pages that are not in use and can be safely disposed is returned to the Stone, which performs the actual page disposal. The Page Manager Gem remembers any pages that are in use, and will retry the removal at a later time.

The Page Manager Gem is also responsible for starting up remote page caches, and handles remote page cache shutdown or unavailability.

The Page Manager Gem is always running. It is started by the Stone during Stone startup, and shut down by the Stone during Stone shutdown. You cannot shut down the Page Manager Gem while the Stone is running; killing the process also brings down the Stone. It does not run during crash recovery or restore from tranlog situations, but is automatically started when recovery or restore is complete.

The Page Manager Gem does not reference a commit record, and has no transactional view of the database. It therefore does not affect the commit record backlog.

SymbolGem and symbol creation

In GemStone/S, Symbols were canonicalized after creation, using a scheme that modified the OOPs in place and handled concurrent updates to the AllSymbols dictionary. This architecture had several failings, both in scalability and robustness. In GemStone/S 64 Bit, symbol creation has been redesigned to be both more reliable and to avoid scaling issues.

GemStone/S 64 Bit introduces a new GemStone user named SymbolUser, and a new Gem, the SymbolGem, which should normally always be running. The SymbolGem is responsible for creating all new Symbols, based on requests from sessions, which are managed by the Stone. Starting and stopping the SymbolGem is handled by the Stone. The SymbolGem must be running in order to create new symbols, which includes any use of any previously undefined Symbol. If the SymbolGem is shut down while user sessions have created new symbols but not yet committed, these user sessions are unable to commit, and will lose all work done in that session.

In GemStone/S 64 Bit, the AllSymbols collection has been redesigned to be more efficient and scalable, using sorted collision buckets to permit fast binary searches, and eliminating the extra space taken by using the symbol as both key and value. AllSymbols is no longer in Globals; it is now in the UserGlobals of the SymbolUser, to reduce the chance of application code inadvertently modifying AllSymbols.

There is a new size limitation: Symbols may not be larger than 1024 bytes.

Symbol creation on the GBS client now requires an additional network roundtrip for each symbol created. Avoid creating many new symbols on remote clients, which can become a performance issue. This is a concern only when creating symbols that did not previously exist.

Using GemBuilder for C, you cannot create symbols using GciCreateByteObj and providing a free OOP returned from GciGetFreeOop. Using any other calls that create Symbols and return the OOP of the Symbol will work.

Improved page server page management

GemStone/S 64 Bit implements a number of improvements to the aioPageServer algorithm, improving the AIO performance for dead reclaim.

When Free Frame and AIO page servers need to preempt pages from the cache, they now favor preempting data pages over other page kinds. However, if the cache is sufficiently low in free frames, they will still preempt any page that is clean.

Free frame caches

Free frame caches have been implemented for all processes that use the shared page cache. This enables processes to add or remove multiple free frames from the free frame list in a single operation, thereby reducing contention on the free frame spin lock. Cached free frames are visible in shared memory so the shared cache monitor can recover any outstanding free frames should a process crash.

To control the size and usage of the Gem and remote page server free frame caches, you can set the configuration options GEM_FREE_FRAME_CACHE_SIZE and

GEM_PGSVR_FREE_FRAME_CACHE_SIZE, respectively. For details, see the *GemStone/S 64 Bit System Administrator's Guide.*

To avoid the maximum size limitation of the free frame cache, the Page Manager bypasses the free frame cache when returning pages to the free frame list.

Segments and Security

Segment protocol in GemStone/S 64 Bit is similar to that in GemStone/S, with the following differences:

- Segments must be committed before they can be used.
- Changes to segments and authorizations, and other changes that may affect read or write authorization checks, take effect only for sessions that log in after the changes are committed.
- Committed Segments cannot be deleted.
- The total number of committed Segments is now limited to 65535.
- Objects can have a nil segment. This is equivalent to world write, but no authorization checks are performed for objects with a nil segment. Nil segments provide the fastest performance. Nil can now be used or returned by any segment protocol.
- New user creation has changed. Most new user creation protocols do not include a Segment argument and create a user with a nil default segment. However, the method `addNewUserWithId:password:` will create and commit a new Segment instance for the new user's default segment.
- Login requires that the user logging in have read authorization for both SystemSegment and DataCuratorSegment.

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

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

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

In GemStone/S 64 Bit, the new DataCuratorGroup is provided to simplify granting privileges for Segment creation and other restricted activities.

GemStone/S 64 Bit provides these methods to help you determine which objects are in a Segment:

```
Repository >> listObjectsInSegments: anArray
Repository >> listObjectsInSegmentToHiddenSet: aSegmentId
Repository >> listObjectsInSegments: anArray toDirectory: aString
```

For details, see the "Object Security and Authorization" chapter of the *GemStone/S 64 Bit Programming Guide.*

Code security

Since object-level security using Segments is optional, GemStone/S 64 Bit defines a new mechanism to ensure that only authorized users are able to modify application code. There is a new privilege #CodeModification, and the existing privilege #OtherPassword is used in new ways. The following table summarizes operations that require one or both of these privileges. Attempting to perform any of these operations without the appropriate privilege will generate a #rtErrNoPriv error. The privileges are required in addition to any necessary Segment authorizations.

Operation	Privilege(s) Required
Modify the classes GsMethod, GsMethodDictionary, Behavior, and Class	#CodeModification
Add a class to a SymbolDictionary	#CodeModification
Add a non-Class object to a SymbolDictionary	none
Remove a class from a SymbolDictionary	#CodeModification
Remove a non-Class object from a SymbolDictionary	none
Add a SymbolDictionary to your own SymbolList[a]	#CodeModification
Add a SymbolDictionary to a SymbolList that is not your own[a]	#CodeModification, #OtherPassword
Remove a SymbolDictionary from your own SymbolList[a]	#CodeModification
Remove a SymbolDictionary from a SymbolList that is not your own[a]	#CodeModification, #OtherPassword
Add or remove a non-SymbolDictionary to/from a SymbolList	none
Modify a UserProfile	#OtherPassword
Modify the AllUsers UserProfileSet	#OtherPassword

a. A SymbolList is your own when (aSymbolList == System myUserProfile symbolList).

The `become:` method does not work with a Class or GsMethod target in the absence of the appropriate privilege.

The following methods (and their senders) also have new restrictions:

```
Behavior >> _primitiveCompileMethod:symbolList:category:
obsoleteClassNames:oldLitVars:

Class >> _subclass:instVarNames:format:constraints:classVars:
classInstVars:poolDictionaries:inDictionary:
inClassHistory:description:isModifiable:

Object >> changeClassTo:

GsMethod >> _at:put:
```

```
Class >> _insertCivAt:
```

You cannot use GCI calls to modify the classes GsMethod, GsMethodDictionary, Class, SymbolDictionary, SymbolList, and UserProfile. Attempting to do so generates an #rtErrObjectProtected error.

Log Files

System log file locations

As in GemStone/S, by default, system gem log files (log files for the GcGems, SymbolGem, and Page Manager Gem) go to the same directory as the Stone log, `$GEMSTONE/data`. In GemStone/S 64 Bit, you can use environment variables to specify alternate directories in which these log files are created. The environment variable must be set in the Stone's UNIX environment before the Stone is started. If the directory specified is invalid or not writable, the default directory is used.

The following environment variables define log directories for the Gem types:

$GEMSTONE_RECLAIM_GC_LOG_DIR
Directory for all Reclaim GcGem logs

$GEMSTONE_ADMIN_GC_LOG_DIR
Directory for Admin GcGem logs

$GEMSTONE_SYMBOL_GEM_LOG_DIR
Directory for SymbolGem logs

$GEMSTONE_PAGE_MGR_LOG_DIR
Directory for Page Manager Gem logs

Runtime control of gem log deletion

The new method `System class >> removeGemLogOnExit:` *aBoolean* overrides the state set in the $GEMSTONE_CHILD_LOG environment variable. If *aBoolean* is true, the gem log file will be deleted if the gem process exits normally. If *aBoolean* is false, the gem log file will not be deleted.

Client Library Changes

Naming

The library naming scheme is different in GemStone/S 64 Bit. The new library names are:

	HP-UX	Solaris, AIX and Linux	Windows
Linked library	libgcilnk64-221.sl	libgcilnk64-221.so	*linked logins from Windows not supported*
Symbolic link name	libgcilnk.sl	libgcilnk.so	*linked logins from Windows not supported*
RPC library	libgcirpc64-221.sl	libgcirpc64-221.so	libgcirpc64-221.dll

	HP-UX	Solaris, AIX and Linux	Windows
Symbolic link name	libgcirpc.sl	libgcirpc.so	libgcirpc.dll (copy; symbolic link unavailable on Windows)

The symbolic links have been created so that the version-independent library name can be used, and application code need not be updated when new versions are installed. On Windows, which does not support symbolic links, the version-independent libraries are copies.

Note that since error translation is no longer supported, `englis*.err` is no longer provided.

Distribution

The distribution of client libraries has also changed. The product distribution no longer includes `clientFiles.zip`.

Windows client libraries are provided in a separate product installation, which also includes Topaz. You may copy the client libraries into more appropriate locations if desired.

VisualWorks on UNIX

Because the 32-bit VisualWorks virtual machine cannot load 64-bit libraries, 32-bit client libraries are provided with the distribution tree, in $GEMSTONE/lib32.

Other GemStone/S Products

GemBuilder for Smalltalk (GBS)

You must use GemBuilder for Smalltalk version 7.1.1 or later, on VisualWorks Smalltalk, with GemStone/S 64 Bit 2.2.1. GBS versions that support VisualAge with GemStone/S 64 Bit 2.2.1 are under development.

From clients running on Windows, only RPC logins are supported.

For the most current information, see the *GemStone/S 64 Bit Installation Guide* chapter on GBS.

Comparison operators

There are some objects that are immediate objects on the server but not on the client. SmallDoubles are replicated as client Doubles; SmallIntegers are replicated as client SmallIntegers if they are within the client SmallInteger range, and as client large integers if they are outside that range. Immediate objects are inherently canonical (a = b implies a== b). However, the client replicates, if not immediate, are not canonical. Therefore, client-side code that compares numbers that may not be immediate should always compare using #=, not #==, although the same code on the server may use #==.

GemBuilder for Java (GBJ)

GemBuilder for Java version 2.3 supports GemStone/S 64 Bit 2.2.1. For details, see the *GemBuilder for Java Installation Guide.*

GemConnect

GemConnect for Oracle version 2.0.1 supports GemStone/S 64 Bit 2.2.1. For details, see the *GemConnect Installation Guide.*

GemEnterprise

GemEnterprise is not available for GemStone/S 64 Bit.

Chapter

2 Garbage Collection

GcGems

In GemStone/S, you had the option of running a single GcGem, or one of several combinations of multiple specialized GcGems. This has been simplified and streamlined in GemStone/S 64 Bit. The GcGem tasks are now performed by two new GcGem types: the Reclaim GcGem and the Admin GcGem. In GemStone/S 64 Bit, the interface to start and stop GcGems has changed, new configuration parameters control the behavior, and many internal parameters are different.

Reclaim GcGems

Reclaim GcGems perform all page reclaim operations, on both shadow objects and dead objects. They perform no other function besides page reclaim. Each Reclaim GcGem performs reclaim on one or more extents. You can have as many Reclaim GcGems running as the number of extents in the repository.

Unlike ParallelDeadReclaim GcGems in GemStone/S, Reclaim GcGems can be run while other sessions are committing changes.

Admin GcGem

The Admin GcGem performs administrative GC functions. It finalizes the vote on possibly dead objects, and performs epoch garbage collection. A repository can have a maximum of one Admin GcGem process running.

Starting and stopping GcGems

The default Admin and Reclaim GcGems are started based on the configuration options STN_ADMIN_GC_SESSION_ENABLED and STN_NUM_GC_RECLAIM_SESSIONS. (For details, see Appendix A of the *System Administration Guide for GemStone/S 64 Bit*.) The default configuration is started by the Stone at startup time.

You may also enable and disable (start and stop) Admin and Reclaim GcGems manually. Unlike in GemStone/S, the Stone will not automatically restart a GcGem that has been disabled.

It is important to understand the difference between *configuring* and *enabling* a GC session. GC session configuration is determined by the settings in the configuration file, or by run-time changes to the configuration. Once you have configured a GC session, it may not necessarily be enabled and running. Disabling a GC session is normally a temporary occurrence, while configuring a GcGem session is intended to be a more permanent change.

To start and stop GcGems, use the following methods:

```
System class >> startAllReclaimGcSessions
System class >> startAdminGcSession
System class >> startAllGcSessions
System class >> startReclaimGemForExtentRange:to:

System class >> stopAllGcSessions
System class >> stopAllReclaimGcSessions
System class >> stopAdminGcSession
```

In GemStone/S 64 Bit, you can now run Reclaim GcGems on a separate host than the Stone, by using the following methods:

```
System class >> startReclaimGemForExtentRange:to:onHost:
System class >> startReclaimGemForExtentRange:to:onHost:
   stoneHost:
```

For details about these methods, see “Running the Admin GcGem” and “Configuring and Starting the Reclaim GcGems” in the “Managing Growth” chapter of the *System Administration Guide for GemStone/S 64 Bit*, and method comments.

The following added GcGem-related methods may also be useful:

```
System class >> adminGcGemSessionId
System class >> reclaimGcSessionCount
System class >> currentGcReclaimSessionsByExtent
System class >> numberOfExtentsWithoutGC
System class >> numberOfExtentRangesWithoutGC
System class >> hasMissingGcGems
```

Epoch garbage collection

In GemStone/S 64 Bit, epoch garbage collection is disabled by default. To enable epoch, set the configuration option STN_EPOCH_GC_ENABLED prior to starting the Stone, or use the methods provided to control the state of epoch GC.

To vary the mark/sweep buffer size of the epoch GC, you can adjust the new Admin GcGem configuration parameter `#epochGcPageBufferSize`. The default value is 150 pages.

You can enable or disable epoch GC at runtime, using the runtime configuration parameter methods. In addition, the following new methods may be used to control epoch garbage collection:

```
System class >> disableEpochGc
System class >> enableEpochGc
System class >> forceEpochGc
System class >> clearEpochGcState
```

GcGem Runtime Parameters

The GcUser's UserGlobals defines a set of GcGem runtime parameters that you can use to tune garbage collection.

GemStone/S 64 Bit provides the following new parameters:

```
#autoRefreshGcGemConfig
#dataPageBufferSize
#enableDebugging
#epochGcPageBufferSize
#maxTransactionDuration
#objsMovedPerCommitThreshold
#reclaimDeadShadowPageThreshold
#verboseLogging
```

For details about these parameters, see the *System Administration Guide for GemStone/S 64 Bit*.

The following GemStone/S 6.1.5 parameters are no longer used in GemStone/S 64 Bit:

```
#commitHeartbeatInterval
#deferEpochReclaimThreshold
#epochGcByteLimit
#epochGcEnabled
#epochGcStats
#epochGcStatsEnabled
#GemIOLimit
#reclaimDeadCommitLimit
#reclaimMaxPages
#reclaimStats
#reclaimStatsEnabled
```

Reclaim Changes

ReclaimAll

In GemStone/S 64 Bit, the method `Repository >> reclaimAll` now acquires the GC lock for the duration of the reclaim all operation, in order to prevent Epoch GC from running. The `reclaimAll` method automatically retries the GC lock for up to 1 minute before raising an #abortErrGarbageCollection error.

The `reclaimAll` method returns without releasing the GC lock. The `postReclaimAll:` now releases the GC lock. This method must be called after a `reclaimAll` in order to release the GC lock.

Garbage Collection using FDC/MGC

Repository-wide garbage collection requires two steps: a full sweep of all objects in the repository, and the marking of each possibleDead object. In GemStone/S 6.1.5, the markForCollection operation combined these steps in a single operation.

In GemStone/S 64 Bit, you have the additional option of performing these operations separately: findDisconnectedObjects (FDC) and markGcCandidates (MGC). FDC generates a collection of all possibleDead objects. This collection is normally written to a binary file, to avoid the need to either commit or keep in memory a potentially very large collection. MGC is then executed, specifying the name of this file.

In GemStone/S 6.1.5, user actions that performed the FDC/MGC steps were provided as an unsupported add-on. In GemStone/S 64 Bit, FDC/MGC are provided as system primitives, and are fully supported.

The primary advantage is that the FDC can be run in a copy of the production repository, to avoid impact on production. It is important that epoch garbage collection be disabled while running offline FDC, to avoid duplicate garbage collection.

NOTE

The requirement to disable epoch while running offline FDC was inadvertently omitted from the System Administration Guide.

In addition, GemStone/S 64 Bit provides a "fast" FDC, which maximizes use of system resources to achieve optimal performance, for use offline when impact on other sessions is not an issue.

For details, see "The FDC/MGC Process" in the "Managing Growth" chapter of the *System Administration Guide for GemStone/S 64 Bit*.

Effect of Other Methods on GcGems

The following methods may have an effect on GcGems running on the system:

```
System class >> suspendLogins
```

No additional GcGems can be started until logins are resumed.

```
System class >> resumeLogins
```

All configured GcGems will be restarted if not running; similar to executing `System startAllGcSessions`.

```
System class >> stopUserSessions
```

Suspends logins; no additional GcGems can be started until logins are resumed.

Chapter

3 Backup and Restore

In addition to backup files created programmatically, GemStone/S 64 Bit allows you to take online extent file backups. An online extent backup is, essentially, a snapshot copy of the repository extents with the system running. If the online backup completes successfully, the copies will be usable to restore the database.

In GemStone/S 64 Bit, online backup is the primary means of performing database backups and can be run during production hours.

The old "full backup" programmatic backup is also available. Full backup functionality is required if you want to reduce the number of extents in the repository, and is intended to be used during non-production hours.

Creating an Online Backup

Checkpoints are not permitted while the online backup is in progress. There must not be a checkpoint in progress when the online backup begins, and no checkpoints are allowed to begin until the online backup has finished. All other database operations (including commits and aborts) are permitted during the online backup.

GemStone/S 64 Bit provides Smalltalk methods that you can use to request the Stone to suspend checkpoints for a user-specified time duration. (To do so, you must have the required privilege.) If the Smalltalk method `System class >> suspendCheckpointsForMinutes:` is called when a checkpoint is in progress, it will block until the current checkpoint completes. If one session attempts to suspend checkpoints and is blocked while the current checkpoint completes, and then a second session attempts to suspend checkpoints, the second session fails and the method returns false.

You cannot suspend checkpoints while in partial transaction log mode or while the repository is in restore mode. However, you can start a new transaction log while checkpoints are suspended. To start a new transaction log while checkpoints are suspended or unsuspended, call `Repository >> startNewLog`.

To query the current status of checkpoints, call `System class >> checkpointStatus`. This method returns an Array object containing a Boolean that indicates whether checkpoints are suspended as well as an Integer that indicates the number of seconds remaining in the suspension.

Once checkpoints are suspended, the session requesting the suspension can log out from GemStone and start the extent copy. Once the extent copy has completed, a session should log in to GemStone and request the Stone to resume checkpoints. The result of the Smalltalk `resumeCheckpoints` method indicates if the online backup was completed while checkpoints were still suspended. If the backup was completed in time, no further action is required and the backup is complete. If the backup did not complete before checkpoints were resumed, then the backup must be discarded and another online backup must be taken. If the system is shut down while checkpoints are suspended, checkpoints are reenabled and a final checkpoint is written during the clean shutdown process. Any online backups in progress during system shutdown must be discarded.

Restoring an Online Backup

To restore the repository from an online extent backup to the last committed transaction, the online backup files must be available, along with all transaction logs written since the checkpoint before the backup was started. The restore procedure includes these steps:

1. Copy the extents from the backup to the location where the repository extents reside.
2. Use **startstone –R –N** to restart GemStone. These options start the stone in restore mode, but do not attempt to automatically recover by replaying all transaction logs.
3. Restore all transaction logs written since the online extent backup was performed. You can query the stone to determine the sequence number of the first transaction log required for the restore.
4. When all transaction logs have been restored, commit the restore. The repository is now ready for use.

Methods that manage checkpoints

The GemStone/S 6.1.5 method `System class >> checkpoint` has been removed from the image. This method had inconsistencies in behavior in the cases where checkpoints were suspended.

GemStone/S 64 Bit provides two new methods in class System (category Transaction Control) to perform checkpoints:

```
System class >> startCheckpointAsync

System class >> startCheckpointSync
```

GemStone/S 64 Bit also provides the following new methods in class System (category Online Backup Support):

```
System class >> suspendCheckpointsForMinutes:

System class >> resumeCheckpoints

System class >> checkpointStatus
```

For details about these methods, see “Methods to Perform Checkpoints” in the “Making and Restoring Backups” chapter of the *System Administration Guide for GemStone/S 64 Bit*.

Example

As part of a complete backup strategy, you can create an online extent backup script for your system. GemStone/S 64 Bit provides an example script `$GEMSTONE/examples/admin/onlinebackup.sh`.

You can customize this example script for your system. You must add the necessary code to perform the file system copies of your extents.

For more information, see “An Example Script” in the “Making and Restoring Backups” chapter of the *System Administration Guide for GemStone/S 64 Bit*.

Session Management During Restore

The process for restoring Smalltalk fullBackups, or restoring transaction logs to any backup, has also changed. The restore methods `restoreFromBackup:`, `restoreFromCurrentLogs`, and so on, now terminate the session when they complete. You will need to log in again to perform the next restore operation.

The GemStone/S method `Repository >> restoreFromLog:` has been removed. Instead, you can use the GemStone/S 64 Bit method `Repository >> restoreToEndOfLog:`, which restores one or more log files by file id, rather than by file name.

To support the restore process, GemStone/S 64 Bit provides the following methods in class Repository (category Backup and Restore):

```
Repository class >> restoreStatusOldestFileId

Repository class >> restoreToEndOfLog: aFileId

Repository class >> setArchiveLogDirectories: arrayOfDirectorySpec

Repository class >> setArchiveLogDirectory: aDirectorySpec

Repository class >> setArchiveLogDirectory: aDirectorySpec
   tranlogPrefix: aPrefix
```

For details about these methods, see “Methods for the restore process” in the “Making and Restoring Backups” chapter of the *System Administration Guide for GemStone/S 64 Bit*.

Reclaim During Transaction Log Restore

Page reclaim is not done when replaying transaction logs during recovery (such as after a crash). However, page reclaim is done during restore of transaction logs — specifically, when restoring an online backup. The page reclaim is done in such a way as to minimize impact on both time and repository space. You cannot configure this reclaim activity.

Chapter

4 System Administration Changes

This chapter summarizes changes in GemStone/S 64 Bit that affect system administration: session control, hidden sets, performance tuning, and more. While this chapter addresses the major changes to consider when porting, it does not attempt to describe every change or difference in behavior, and does not document new features in detail.

For more details about the topics in this chapter, see the *System Administration Guide for GemStone/S 64 Bit*.

Session Control Changes

In GemStone/S 64 Bit, the method `System class >> stopOtherSessions` has been replaced by the method `System class >> stopUserSessions`, which stops other user sessions but leaves the GcGems, Symbol Gem, and Page Manager Gem running.

Other methods that previously required stopping other sessions have been redesigned to stop the sessions automatically as needed. There is no longer any need to stop sessions as part of regular maintenance. The new Symbol Gem and Page Manager Gem sessions should always be running and can cause potentially serious problems if they are shut down inappropriately.

The following methods have been added:

`System class >> currentUserSessionCount`

Returns the number of user sessions present in the system; does not include GC sessions, the Symbol Gem, or the Page Manager Gem.

`System class >> currentSessionCount`

Returns the number of sessions present in the system, including the Symbol Gem and GC sessions, but not the Page Manager Gem.

To bypass the normal `stopSession:` timeout, particular in cases where you need to stop a session immediately, a method has been added that allows you to specify the timeout:

```
System class >> terminateSession: aSessionId timeout: aSeconds
```

This method can be used to stop all sessions, including GcGems, but not the Symbol Gem. To stop any session, including the Symbol Gem, the following new method may be used. This method may only be executed by SystemUser.

```
System class >> terminateSymbolCreationSession: aSessionId
   timeout: aSeconds
```

The new method `System class >> logout` allows sessions to log themselves out even if they do not have session control privilege.

Signal in-transaction sessions on CR backlog

In GemStone / S 64 Bit, sessions that are in transaction can be notified of a commit record (CR) backlog, allowing them to update their commit record via a continueTransaction to avoid the problems associated with commit record backlogs. The mechanism is similar to sigAbort, but without a subsequent timeout such as LostOTRoot.

The new signal #rtErrSignalFinishTransaction (6012) has been added.

The following new methods allow you to turn on and off signaling in transaction, and to determine the current status:

```
System class >> disableSignaledFinishTransactionError
System class >> enableSignaledFinishTransactionError
System class >> signaledFinishTransactionErrorStatus
```

To explicitly send a signal to another session, the following method has been added:

```
System class >> sendSignalFinishTransactionToSession:
```

Sessions that have enabled receipt of this signal should set up a signal handler for the #rtErrSignalFinishTransaction signal.

Object Audits

Object audits, including quick audits, now perform a reclaimAll by default. If there are dead or shadowed objects in the system and the Admin and Reclaim GcGems are not running on the system, the object audit signals the new error #rtErrReclaimAllMissingGcGem. If the Admin and Reclaim GcGems are running, but other sessions are running and do not abort to allow voting to complete, reclaimAll will not complete and the audit will appear to hang.

If you do not have the appropriate GcGems running, or if there are other user sessions running, use one of the following operations:

```
SystemRepository objectAuditNoReclaim
```

```
SystemRepository auditWithLimit: sizeLimit reclaimAll: false
```

In order to safely do an object audit with reclaim, you may use the following sequence of operations:

```
System stopUserSessions.
System startAdminGcSession.
System startAllReclaimGcSessions.
SystemRepository objectAudit.
```

Core Dump Behavior

In GemStone/S, a process that encountered a fatal error wrote a core file to preserve the stack trace information. This file included the dump of the full shared memory region, and resulted in unreasonably large files on systems with a large shared page cache.

In GemStone/S 64 Bit, the C level stack trace is written to the process log file prior to the process shutdown, on all platforms (Solaris, Linux, AIX and HP-UX). With this information, the actual core file is usually unnecessary. In GemStone/S 64 Bit, the core files by default are no longer written when a fatal error occurs. If a core file is needed for some reason, you can set the environment variable GS_WRITE_CORE_FILE, and core files will be written as in GemStone/S.

The amount of time a process sleeps before exiting after a core dump has also changed, and is now 1 minute by default. You can configure longer sleep times by setting the environment variable GS_CORE_TIME_OUT to the number of seconds the process should sleep after a core dump before exiting.

Translating OOP Value to OOP Number

Because several of the bits in an OOP are used for special objects, integers, etc., the number of OOPs used in a GemStone system is smaller than the OOP itself.

The number returned by:

```
System _oopHighWaterMark
```

is the OOP (or OOP value). The count of the number of OOPs is a percentage of this value, referred to as the OOP number. The actual relationship varies in different products and releases. The following new method allows you to avoid the need to calculate this manually:

```
System class >> _oopNumberHighWaterMark
```

Note that the methods `_oopHighWaterMark` and `_oopNumberHighWaterMark` return the exact numbers, requiring a call to the Stone to complete. Two other new methods allow you to obtain approximate values, with a faster return and less impact on the system:

```
System class >> _approxOopHighWaterMark
System class >> _approxOopNumberHighWaterMark
```

Hidden Sets

Hidden sets are C level internal bitmap structures, which behave somewhat like Smalltalk IdentitySet. They do not consume object memory and do not affect temporary object cache settings; this makes them useful in working with very large collections of objects without risking out-of-memory issues.

Hidden sets are used in GemStone/S for internal object tracking tasks. In GemStone/S 64 Bit, hidden sets are also available for customer use. In addition, there are changes in the details of how GemStone/S 64 Bit uses hidden sets. The current use of internal hidden sets is subject to change with each release. For details, see the method `System class >> HiddenSetSpecifiers`.

Hidden sets 41 through 45 are available for customer use. You may use the following methods to access and modify the contents of hidden sets 41 to 45:

```
System class >> writeHiddenSet: hiddenSetSpecifier toFile: aString
System class >> readHiddenSet: hiddenSetSpecifier
   fromFile: aString
System class >> addHiddenSet: first to: second
System class >> removeContentsOfHiddenSet: first from: second
System class >> computeUnionOfHiddenSet: first and: second
   into: third
System class >> computeDifferenceOfHiddenSet: first and: second
   into: third
System class >> removeFirst: count objectsFromHiddenSet:
   hiddenSetSpecifier
System class >> truncateHiddenSet: hiddenSetSpecifier
   toSize: newSize
```

These methods may only be used with customer-available hidden sets (41–45). However, this check is bypassed if the session is logged in as SystemUser. **Use extreme caution in this case**.

To enumerate hidden sets, you may use the following methods:

```
System class >> _hiddenSetEnumerate: hiddenSetSpecifier
   limit: maxResultSize
System class >> _hiddenSetEnumerateAsInts: hiddenSetSpecifier
   limit: maxResultSize
```

These methods remove the first *maxResultSize* objects from the hidden set, and return them in an Array. `_hiddenSetEnumerate:` returns the objects corresponding to the OOPs, while `_hiddenSetEnumerateAsInts:` returns Integer OOPs

```
System class >> _hiddenSetAsArray
```

This method removes and returns the entire hidden set contents. Note that this can create a very large collection object and cause out-of-memory errors.

Alternatives to listInstances: for Large Result Sets

`Repository >> listInstances:` can return a very large result set, with the associated risk of out-of-memory errors. To avoid the need to have the entire result in memory, GemStone / S 64 Bit provides the following new methods:

`Repository >> listInstances:` *anArray* `limit:` *aSmallInteger*

This method is similar to `listInstances:`, but returns just the first *aSmallInteger* instances of each of the classes in *anArray*.

`Repository >> listInstancesToHiddenSet:` *aClass*

This method puts the set of all instances of for *aClass* in hidden set 1. You can then use the method `System class >> _hiddenSetEnumerate:limit:` to enumerate the results of the listInstances, in hidden set 1, in chunks. Note that this hidden set number may be subject to change in new releases; for a list of hidden sets in a particu-

lar release, see the GemStone Smalltalk method `System class >> HiddenSetSpecifiers`.

`Repository >> listInstances:` *anArray* `toDirectory:` *aString*

This method is similar to `listInstances`, but writes the results to a binary bitmap (.bm) file in the specified directory. This file can later be read into a hidden set. Bitmap files are named *className-classOop*-instances.bm, where *className* is the name of the class and *classOop* is the object ID of the class.

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

For more information, see the image method comments.

Signal When Transaction Logs Are Full

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

In GemStone/S 6.1.5, this signal was sent to administrative sessions only and did not require specific enabling. In GemStone/S 64 Bit, any session can be set up to receive the signal, and each session must enable receipt.

The following methods have been added:

`System class >> enableSignalTranlogsFull`

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

`System class >> disableSignalTranlogsFull`

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

`System class >> signalTranlogsFullStatus`

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

Cache Warmup

In GemStone/S 64 Bit, you can load object and data pages into the shared page cache on startup. This allows the overhead of initial page loading to occur in a controlled way on system startup, rather than more gradually as the repository is in use.

To start the cache warming sessions, you use the new utility `$GEMSTONE/bin/startcachewarmer`. For more information, see the "GemStone Utility Commands" appendix of the *System Administration Guide for GemStone/S 64 Bit*.

GemStone/S 64 Bit also provides the following methods for cache warming:

```
Repository >> readObjectTableForGem:of:useSharedCache:
   cacheFullLimit:

Repository >> readObjectTableAndDataPagesForGem:of:
   useSharedCache:cacheFullLimit:

Repository >> readObjectTableForGem:of:useSharedCache:
   loadDataPagesOpCode:cacheFullLimit:
```

Error Message Translation Removed

GemStone/S 64 Bit does not provide translation of system error messages. The related files `englis*.err`, `msgcom`, and `saymessage` are no longer part of the product.

Support for Replicates Removed

In GemStone/S 64 Bit, replicate extents and replicate transaction logs are not supported. The following methods are no longer in the image:

```
Repository >> createReplicateOf:named:
Repository >> disposeReplicate:
Repository >> currentLogReplicate
```

The value returned from `System stoneConfigurationReport at: #StnCurrentTranLogNames` is now a single tranlog identifier, rather than a collection.

The following 6.1.5 configuration options are not present in GemStone/S 64 Bit:

DBF_REPLICATE_NAMES
STN_REPL_TRAN_LOG_DIRECTORIES
STN_REPL_TRAN_LOG_PREFIX

The following table lists methods that are obsolete in GemStone/S 64 Bit, and the replacement method or sequence of methods to use.

Deprecated GemStone/S 6.1.5 Method	Replacement Method(s) in GemStone/S 64 Bit
`Repository >> addTransaction-Log:replicate:size:`	`Repository >> addTransaction-Log:size:`
`Repository >> setArchiveLogDi-rectories:tranlogPrefix:repli-cateDirectories:replicatePrefix:`	`Repository >> setArchiveLogDi-rectories:tranlogPrefix:`
`Repository >> restoreFromAr-chiveLogDirectories:tranlogPre-fix:replicateDirectories:replica tePrefix:`	`Repository >> setArchiveLogDi-rectories:tranlogPrefix:.` `Repository >> restoreFromAr-chiveLogs`

Csh Versions of Scripts No Longer Provided

Csh versions of GemStone scripts, other than gemsetup.csh, are no longer maintained or distributed. Scripts are provided as posix-compliant.sh scripts. Users should use a posix-compliant shell, such as Bourne or Korn shell.

Performance Tuning

Configurable maximum for asynchronous I/O requests

In GemStone/S 64 Bit, you can configure the maximum number of pending AIO transaction log write requests by setting the configuration option **STN_MAX_AIO_REQUESTS**. If more than this number of asynchronous writes are requested, the Stone will wait (sleep) until one or more of the pending requests have completed.

Allow Stone to postpone sleep

If there is no work for the Stone, the Stone will sleep, waking up when there is work to do. In GemStone/S 64 Bit, you can avoid this sleep by setting the configuration option **STN_LOOP_NO_WORK_THRESHOLD**. Avoiding this sleep incurs more CPU, but may allow the Stone to respond more quickly.

For more information about STN_LOOP_NO_WORK_THRESHOLD, see the “GemStone Configuration Options” appendix of the *System Administration Guide for GemStone/S 64 Bit*. See also the descriptions of the cache statistics StnLoopNoWorkThreshold, StnLoopsNoWork, and StnLoopsSinceSleep in the “Monitoring Performance” chapter of the *System Administration Guide*.

Control of sessions on commit queue

If there are a number of sessions waiting to commit, the Stone allows the later ones to perform work processing unions, which may make them unavailable when their turn comes up to commit. You can now set the configuration option **STN_TRAN_Q_TO_RUN_Q_THRESHOLD** to specify the number of sessions in the commit queue that will be required to keep waiting.

Chapter

5 Changes That Affect Application Code

This chapter summarizes changes in GemStone/S 64 Bit that may affect your application code: indexing, exception handling, object constraints, locking, transaction handling and collections, and more. While this chapter addresses the major changes to consider when porting, it does not attempt to describe every change or difference in behavior, and does not document every new feature in detail.

For more details about the topics in this chapter, see the *GemStone/S 64 Bit Programming Guide*.

Indexing

Indexing in GemStone/S 64 Bit is similar to GemStone/S, with the following exceptions:

- Set-valued indexes (indexes with a '*' in the path) are not supported in GemStone/S 64 Bit.
- Constraints are no longer used, although protocol remains to set constraints when creating classes. The method `UnorderedCollection >> createEqualityIndexOn:` should no longer be used. Instead, you should always use `UnorderedCollection >> createEqualityIndexOn:withLastElementClass:`
- Index creation methods with the `commitInterval:` keyword are no longer available in GemStone/S 64 Bit. The IndexManager class now controls transactional behavior of index creation and removal.
- Nils may now be valid objects in indexed collections.
- Indexes now correctly handle various NaNs (not a number).

- When sorting heterogeneous collections in an index, the ordering is as follows (from low to high):

  `UndefinedObject`
  `Symbol` and `String`
  `DoubleByteSymbol` and `DoubleByteString`
  `Boolean`
  `Character`
  `Number` (NaN sorts before other Numbers)

- As in GemStone/S, Btree nodes cache some or all of an indexed object, avoiding the need to look up the actual object on disk for some indexed lookups. The encoding of objects into the caches has been modified, allowing a reduction in the space consumed by nodes.

Automatic index maintenance is done using a new mechanism (not using tag 0, as in GemStone/S). The object dependencies are kept in an internal table parallel to the shared object table, the DependencyMap. Using the DependencyMap, rather than marking the object itself, avoids the problem of modifying the object when the object is added or removed from the index, and the risk of commit conflicts when the object values are unchanged. It also avoids shadowing objects that are added to or removed from indexes, making index creation noticeably faster.

As in GemStone/S, the DependencyLists mechanism is used to maintain the specific index relationships.

It is possible for sessions to encounter DependencyMap conflicts, if one session modifies an object while another session has DependencyMap updates for that same object. DependencyMap updates include the object being adding or removed from an index, or a change in the number of indexes that the object participates in. The second session to commit will fail with a new type of conflict, the Write-Dependency commit conflict.

Two new methods have been added for Write-Dependency conflicts:

```
System class >> currentTransactionWDConflicts
System class >> currentTransactionHasWDConflicts
```

These methods behave as similar methods that detect and return other types of conflicts. For more information, see image method comments.

IndexManager

GemStone/S 64 Bit provides a new class, **IndexManager**, to control index maintenance operations. There is a singleton instance of this class, lazy initialized and available via the class method `current`.

The class IndexManager controls the transactional behavior of index creation and removal. IndexManager provides methods that allow you to commit your work to the repository incrementally during index creation (or removal). This approach enables you to avoid out-of-memory conditions, while reducing the overall time required to build the index. When **autoCommit** is set to true, the current transaction is committed during indexing whenever either the **dirtyObjectCommitThreshold** or the **percentTempObjSpaceCommitThreshold** is reached.

The IndexManager can also provide general information about indexes on your system — for example, returning a list of all the collections that have indexes.

For more information, see the *GemStone/S 64 Bit Programming Guide.*

Reduced-conflict equality indexes

GemStone/S 64 Bit provides a new type of index, a reduced-conflict equality index. This allows you to avoid some index maintenance-related commit conflicts. To create a reduced-conflict equality index, use the method:

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

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

Indexing audit now public

Indexing audit has been modified extensively to improve performance greatly, as well as to catch more forms of index corruption. Indexing audit is now fast enough, and is recommended, to be run as part of regular repository maintenance including production systems. To reflect this, it has been renamed without the initial underscore:

```
UnorderedCollection >> auditIndexes
```

ANSI Exception Handling

The ANSI Exception handling framework was available as a goodie in GemStone/S. In GemStone/S 64 Bit, it is provided with the image. Provision is made for signaling that an exception has occurred and for defining handlers for signaled exceptions.

The legacy exception handing mechanisms are still available; upgraded applications do not need to be modified. The ANSI framework is built completely out of the legacy framework, and is intended to be backward-compatible with it. In order to accommodate the legacy framework, the top-level exception in the ANSI framework is named `ExceptionA` rather than `Exception`.

ANSI Exceptions are class-based, meaning that you use a class in the ExceptionA hierarchy to describe errors and other exceptions in your GemStone Smalltalk programs. ANSI errors, for example, include the new GemStone Smalltalk classes `MessageNotUnderstood` and `ZeroDivide`.

SUnit, an open source testing framework, has also been added.

Object Constraints

In GemStone/S 64 Bit, object constraints are no longer enforced by the virtual machine, although some Smalltalk methods may still enforce constraints.

In place of the subclass creation methods that included the non-functional `constraints:` keyword, GemStone/S 64 Bit provides the following new methods without the keyword:

```
Class >> indexableSubclass:instVarNames:classVars:
   classInstVars:poolDictionaries:inDictionary:
   instancesInvariant:isModifiable:
```

```
Class >> indexableSubclass:instVarNames:classVars:
   classInstVars:poolDictionaries:inDictionary:
   instancesInvariant:newVersionOf:isModifiable:

Class >> subclass:instVarNames:classVars:classInstVars:
   poolDictionaries:inDictionary:instancesInvariant:isModifiable:

Class >> subclass:instVarNames:classVars:classInstVars:
   poolDictionaries:inDictionary:instancesInvariant:
   newVersionOf:isModifiable:
```

The old methods are still available but may be removed in future releases.

Locking

ExclusiveLocks no longer available

ExclusiveLocks are no longer available in GemStone / S 64 Bit. ReadLocks and WriteLocks remain and are unchanged.

The following GemStone / S 6.1.5 methods have been removed:

```
System class >> exclusiveLock:
System class >> exclusiveLock:ifDenied:ifChanged:
System class >> exclusiveLockAll:
System class >> exclusiveLockAll:ifIncomplete:
```

The methods `systemLocks` and `sessionLocks` continue to return a three-element array. However, the final element, which previously contained exclusive locks, is always empty.

Application write lock

GemStone / S 64 Bit introduces a new type of lock, the *application write lock*. An application write lock differs from a regular GemStone write lock in these ways:

- When you request an application write lock on an object, the request will not return until the lock is granted, or until the wait times out. This frees you from having to repeatedly request a lock if it is not immediately available. Timeout is controlled by the configuration option STN_OBJ_LOCK_TIMEOUT.
- When you request an application write lock, you must specify a *lock queue*. There are ten lock queues available in the Stone. Once you use a lock queue to lock an object, that queue can only be used to lock that object, until the Stone is stopped and restarted. Thus, you must choose no more than ten objects that can be application write-locked, and you must take down your repository if you want to change that choice.
- Application write locks can detect whether a request would cause deadlock, and will deny such a request.

For more information, see the *GemStone/S 64 Bit Programming Guide*.

Transaction Handling

continueTransaction

After a failed commit, you must now abort before you can use `continueTransaction`. In GemStone/S 64 Bit, if a call is made to `continueTransaction` after a failed commit (due to conflicts) but without subsequently aborting, the call signals the new error 2409, `#rtErrContinueTransFail`.

Transaction conflicts

The transaction conflict type Read-ExclusiveLock has been removed.

A new transaction conflict type, Write-Dependency, has been added. For more information, see page 40.

Reduced conflict retries now retry multiple times

Reduced conflict logic provides the capability of retrying operations that failed to commit. In GemStone/S 6.1.5, there was one retry before reporting the failure to the session. (Internal logic provided for three retries, but only one retry was being attempted.)

In GemStone/S 64 Bit, the system attempts to retry 15 times before reporting failure. A new commit result, #retryLimitExceeded, is returned in this case.

To avoid race conditions, the retries are serialized using a new type of lock, the RcWriteLock. This lock uses an instance of Object, which is in Globals at #GemStoneRCLock. To disable use of the lock object, set (`Globals at: #GemStoneRCLock`) to nil. (To do so, you must have SystemUser privileges.)

To support the lock, GemStone/S 64 Bit provides the new method `System class >> waitForRcWriteLock:` *rcLockObject*. This is not intended for general use. If any other sessions have locked *rcLockObject*, this method will wait for it to be released before returning; it will time out according to the configuration parameter STN_OBJ_LOCK_TIMEOUT.

Methods that now perform aborts

In GemStone/S 64 Bit, the following methods now perform an abort. If there are modifications to persistent objects that would be lost due to the abort, the method fails to abort and returns the new error 2412, `#rtErrAbortWouldLoseData`.

The following methods are affected by this change. Methods that call any of these methods are also affected. This is not an exhaustive list.

```
Repository >> fullBackupTo:MBytes:compressed:
Repository >> continueFullBackupTo:MBytes:
Repository >> continueFullBackupCompressedTo:MBytes:
Repository >> findObjsConnectedTo:
Repository >> findDisconnectedObjectsAndWriteToFile:
   pageBufferSize:saveToRepositor:
Repository >> findReferencePathToObjs:limitObjArray:
   findAllRefs:printToLog:
Repository >> listInstancesToHiddenSet:
Repository >> listInstances:limit:
```

```
Repository >> listReferences:
Repository >> listReferences:withLimit:
Repository >> markForCollection
Repository >> markForCollectionWait:
Repository >> markGcCandidatesFromFile:forceOnError:
Repository >> reclaimAll
Repository >> auditWithLimit:reclaimAll:
Repository >> repairWithLimit:
System class >> findObjectsLargerThan:limit:
UnorderedCollection >> createIdentityIndexOn:commitInterval:
UnorderedCollection >> createEqualityIndexOn:commitInterval:
```

If the session is in manual transaction mode and is in a transaction, a new transaction is begun before returning, leaving the session in the same state as when the method is invoked.

SoftReferences

GemStone/S 64 Bit adds a new feature, SoftReferences. Two new Dictionary subclasses, KeySoftValueDictionary and IdentityKeySoftValueDictionary, allow the virtual machine to remove entries as needed to free up memory. For more information, see the "Collection and Stream Classes" chapter of the *GemStone/S 64 Bit Programming Guide*.

In order to ensure that these SoftReferences are themselves not persisted in the repository, a new attribute has been added to instances of Class, allowing them to be flagged as non-persistent. For more information on non-persistent Classes, see the "Transactions and Concurrency Control" chapter of the *GemStone/S 64 Bit Programming Guide*.

Collections

RcQueue

RcQueueEntry

GemStone/S 64 Bit includes the new class RcQueueEntry, a subclass of RcQueueElement that is used interchangeably. While converted repositories may include RcQueues containing RcQueueElement, any objects added to an RcQueue will be an RcQueueEntry, rather than RcQueueElement. There is no need to perform any modifications to existing RcQueues.

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

Performance

For performance, conflicts on the RcQueue itself are no longer handled by reduced-conflict (RC) retry logic. This means that attempts to grow the RcQueue while in use are likely to fail. Therefore, an RcQueue is no longer lazy initialized; instead, session components are added at RcQueue creation time when the `new:` method is used. If you

know the maximum number of sessions that will be using the RcQueue, it is advisable to specify the size on creation.

Some RcQueue methods have been reimplemented as primitives for performance.

Fast collection sorting

SortedCollections are sorted with a simple binary search. With large SortedCollections, performance is less than desirable. New methods have been added:

```
Collection >> sortWithBlock:
Collection >> sortWithBlock:persistentRoot:
```

These methods sort the elements of Collections using an n-way merge sort, and return an Array containing the sorted elements. Unlike SortedCollections, the Arrays will not retain the sort if more elements are added.

The method `Collection>>sortWithBlock:persistentRoot:` allows you to pass in an empty Array, which is used to persist the intermediate results of the sort (provided the IndexManager's autoCommit is true). This allows you to sort Collections that are too large for temporary object space, avoiding out-of-memory errors.

Manipulating large collections without faulting contents

GemStone/S 64 Bit provides new methods to copy the contents of a Collection without faulting the contents into memory:

```
IdentityBag >> copyFrom:count:into:startingAt:
IdentityBag >> copyFrom:to:into:startingAt:
OrderedCollection >> addAll:
OrderedCollection >> _addAllFromNsc:
SequencableCollection >> copyFrom:count:into:startingAt:
```

If the argument to `IdentityBag >> addAll:` is an Array or OrderedCollection, the elements in the collection are not faulted into memory.

Other Changes

Added methods

The following methods have been added:

`Time class >> secondsElapsedTime:` *aBlock*.

This method returns a Float whose value represents the number of seconds elapsed during execution of *aBlock*, with microsecond resolution.

`DateTime >> asMillisecondsGmt`

Returns the number of milliseconds since midnight, January 1, 1901, GMT.

`Object class >> _objectForOop:` *anOop*

Returns the object corresponding to the given oop. This method is provided for convenience and should be used with caution.

NOTE

In GemStone/S, the method performing this function was named `_objectWithOop:`.

The behavior for the methods `CharacterCollection >> asArrayOfPathTerms` and `EUCString >> asArrayOfPathTerms` has changed. The backslash character is no longer recognized as an escape character, and each of the terms is expected to be a valid path term (as defined in `CharacterCollection >> _isValidPathTermName`).

The deprecated method `System class >> deleteServerFile:` no longer accepts wildcard arguments. Do not use this method; use `GsFile >> removeServerFile:` instead.

Some milliseconds no longer roll over at 524287999

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

The GemStone/S 6.1.5 “goodies” classes Random or FastRandom depended on `_timeMs` returning a value less than 524287999. If you have filed these classes into your application, they will not be automatically upgraded during upgrade/conversion. You must manually file in the updated classes.

Configurable number of shared counters

Shared counters provide a means for multiple sessions on the same host to share a common counter value (for creation of unique keys, etc.). Previously, you were limited to 1900 shared counters. This is now configurable via the new configuration parameter SHR_PAGE_CACHE_NUM_SHARED_COUNTERS (page 61).

For shared counter related protocol, see the image methods in System class category Shared Counter.

Compiler interpretation of the ‘_’ character

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

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

Symbol printing

All Symbols now print with single quotes, in the form #'....'. This includes Symbols that would previously have been printed without quotes, in the form #....

Chapter

6 GemBuilder for C

UserAction Compile and Link Changes

GemStone has adjusted the code base to use new compilers; a C++ compiler is required, although code may be written in C. We strongly recommend that you use the specific compilers listed in the documentation to compile user actions, although other compilers may work.

With the new compilers, the compile and link flags for user actions have also changed. For a list of compiler versions, along with compile and link information, refer to the *GemBuilder for C* manual for GemStone / S 64 Bit.

GemStone / S 32-bit user actions do not work with GemStone / S 64 Bit. You must convert and rebuild all user action code in order for it to run in 64-bit mode. For example, you must replace all references to `long` in the 32-bit libraries with `int` in the 64-bit version.

GemStone / S 64 Bit no longer supports build time binding of user action code.

Static user actions use "`.a`" rather than "`.o`" files

Changes in Functions

There have been changes to the GCI entry points. For a more detailed description of the GemStone C Interface, see the GemStone / S 64 Bit *GemBuilder for C* manual, and the include files.

Removed GCI functions

The following GCI functions have been removed from GemStone / S 64 Bit:

GciAddSaveObjsToReadSet
GciEncodedLongToOop
GciErrMsgSymToFile (undocumented API)
GciErrMsgSymToText (undocumented API)
GciFetchIdxOop (undocumented API)
GciFetchIdxOops (undocumented API)
GciFetchIdxSize (undocumented API)
GciHandleError
GCI_IS_REPORT_CLAMPED
GCI_LONG_IS_SMALL_INT (replaced by GCI_I64_IS_SMALL_INT)
GCI_LONGJMP (replaced by new function GciLongJmp)
GciLongToOop (use GciI64ToOop instead)
GCI_LONG_TO_OOP
GciNetOobHandler (undocumented API)
GCI_OOP_IS_CHAR16
GciOopToEncodedLong
GciOopToLong (use GciOopToI64 or GciOopToI32 instead)
GCI_OOP_TO_LONG
GciOopToUnsignedLong
GciPushErrHandler
GciRefreshClassCache (undocumented API)
GciReplaceOopsInNsc (use GciReplaceVaryingOops)
GCI_SETJMP (replaced by new macro Gci_SETJMP)
GciUnsignedLongToOop

GciSendMsg — This function no longer exists. For convenience, it is provided as an inline function (see gcisend.hf), without variable arguments. Use GciPerform instead.

Renamed GCI functions

The following GCI functions have been renamed in GemStone / S 64 Bit. You must manually inspect any application code that uses these functions. Most of these functions now return an int64, and thus a C variable of type int will not hold the complete result, risking data loss.

Table 1 Renamed GCI Functions

GemStone/S 6.1.5 name	GemStone/S 64 Bit v2.x name
GciFetchBytes	GciFetchBytes_
GciFetchChars	GciFetchChars_
GciFetchSize	GciFetchSize_
GciFetchVaryingSize	GciFetchVaryingSize_
GciObjRepSize	GciObjRepSize_
GciProcessDeferredUpdates	GciProcessDeferredUpdates_
GciSetCacheName	GciSetCacheName_

Changes in GCI functions

long

All arguments and return types that were of type `long` in GemStone/S 6.1.5 are now `int` or `int64`.

ArraySizeType

In GemStone/S 6.1.5, the type `ArraySizeType` was used to represent the size of an array. Because SmallIntegers are now 61-bit signed integers, all `ArraySizeType` arguments have been replaced with `int` or `int64`, to allow for larger arrays and strings in GemStone/S 64 Bit.

Internal sets

Internal/hidden sets have changed. For example, in GemStone/S 64 Bit, the PureExportSet no longer exists. Therefore, many functions that affected internal/hidden sets (for example, `GciDirtySaveObjs`, `GciReleaseOops`, etc.) operate on the internal sets somewhat differently. To ensure the desired behavior, examine any code that operates on internal sets.

Traversal buffers

In GemStone/S 64 Bit, some functions have traversal buffer arguments with the new data type `GciTravBufType`, which encapsulates information that was formerly kept in a ByteArray. Most of these functions also have fewer arguments than in 6.1.5, as the size of the traversal buffer is no longer an argument.

The following functions are affected:

`GciFindObjRep` (no change in the number of arguments)
`GciMoreTraversal`
`GciNbClampedTraverseObjs`
`GciNbMoreTraversal`
`GciNbPerformTraverse`
`GciNbStoreTrav`(no change in the number of arguments)
`GciNbTraverseObjs`
`GciPerformTraverse`
`GciStoreTrav` (no change in the number of arguments)
`GciTraverseObjs`

Other changes in functions

The following functions have changed in GemStone / S 64 Bit. For details about any of these functions, see the GemStone / S 64 Bit *GemBuilder for C* manual.

Table 2 Other changed GCI functions

Function Name	Change in GemStone/S 64 Bit
`GciPopErrJump` `GciPushErrJump`	Argument type changed from `jmp_buf` to `GciJmpBufSType *`
`GciStorePaths`	Changes in error detection behavior.
`GciAlteredObjs`	Fewer arguments; since symbol canonicalization is no longer required, the related arguments have been removed.
`GciCreateByteObj` `GciCreateOopObj`	The argument `makePermanent` has no effect in GemStone / S 64 Bit. Also note that to create a symbol, you cannot pass in an OOP; Symbol OOPs are assigned by the SymbolGem.
`GciStoreTrav` `GciNbStoreTrav`	The store traversal flag `GCI_STORE_TRAV_CREATE_PERMANENT` has no effect in GemStone / S 64 Bit.
`GciFetchObjInfo` `GciFetchObjectInfo`	May return a different number of elements in the results for OOP objects, since `sizeof(OopType)` has changed from 4 to 8.
`GciSetCacheName_`	Now returns a `BoolType` value rather than `void`. (Renamed from `GciSetCacheName`.)
`GciObjRepSize_`	Now returns a `size_t` value rather than `long`. (Renamed from `GciObjRepSize`.)

Added GCI functions

The following GCI functions and macros have been added in GemStone / S 64 Bit:

GciByteArrayToPointer
GciClampedTravRefs
GciDecSharedCounter
GciFetchNumSharedCounters
GciFetchSharedCounterValuesNoLock
GCI_I64_IS_SMALL_INT
GciI64ToOop
GciIncSharedCounter
GciInitAppName_ (variant of GciInitAppName)
GciLongJmp replaces GCI_LONGJMP
GciNbClampedTravRefs
GciNbStoreTravDoTravRefs
GciObjIsCommitted
GciOldOopToNewOop
GCI_OOP_SPECIAL_VALUE_SHIFT
GciOopToI32
GciOopToI32_
GciOopToI64
GciOopToI64_
GciPointerToByteArray
GciReadSharedCounter
GciReadSharedCounterNoLock
GciReleaseAllGlobalOops
GciReleaseGlobalOops
GciSaveGlobalObjs
GciServerIsBigEndian
GciSetHaltOnError
Gci_SETJMP
GciSetSharedCounter
GciSetTraversalBufSwizzling
GciStoreTravDoTravRefs

The following functions were present in 6.1.5, but not in the GCI manual, and remain available and are documented and supported in GemStone / S 64 Bit:

```
GciDbgEstablishToFile
GciDbgLogString
GciDecodeOopArray
GciDirtyExportedObjs
GciDirtyTrackedObjs
GciEnableFreeOopEncoding
GciEnableFullCompression
GciEncodeOopArray
GciFetchNumEncodedOops
GciFloatKind
GciGetFreeOopsEncoded
GciOopToChar16
GciProduct
GciReleaseAllTrackedOops
GciReleaseTrackedOops
GciSaveAndTrackObjs
GciStep
GciStringToInteger
GciTrackedObjsFetchAllDirty
GciTrackedObjsInit
```

Changes to GCI Structured Types

GemStone / S 64 Bit introduces the following GCI structured type:

```
GciTravBufType
```

This type encapsulates information that was formerly kept in a ByteArray. For more information, see the GemStone / S 64 Bit *GemBuilder for C* manual.

The new class `GciTravBufHolder` simplifies use of a `GciTravBufType` within a single C function.

With the type changes listed elsewhere in this chapter, most of the structured data types have been modified. This includes replacing longs with `int` or `int64`, including the new `GciTravBufType`, and replacing `ArraySizeType` with `int` or `int64`. Some C structs have been replaced by C++ classes. In addition, fields have been reordered. If you use any of GemStone's structured data types, please review the changes in the `gci*` include files.

In particular, `GciStoreTravDoArgsSType` has changes to accommodate new ExecuteBlock functionality, and fields that formerly included `OopType` Segment now use `unsigned short` segmentId.

Chapter

7

Topaz Changes

New Command Line Argument

In GemStone/S 64 Bit, you can use the new Topaz command line argument `-T` to specify the temporary object cache size. This setting takes precedence over the configuration file setting.

`-T` *tocSizeKB*

This argument is only valid for linked Topaz.

New Command STK

GemStone/S 64 Bit provides the new command STK, which is similar to STACK but does not display parameters and temporaries for each context.

Expanded EXPECTVALUE Functionality

In GemStone/S 64 Bit, EXPECTVALUE now accepts additional arguments. For more information, see the *GemStone/S 64 Bit Topaz Programming Environment Manual*.

OUTPUT Keywords

The Topaz `OUTPUT` command has been modified for more clarity in file handling, and to allow scripts to work identically in both GemStone/S and GemStone/S 64 Bit. In GemStone/S 64 Bit, the `OUTPUT` command has the following behavior:

`output` *aFileName* `output push` *aFileName*	Overwrites the specified file, or creates it (if the file does not already exist).
`output` &*aFileName* `output push` &*aFileName*	Appends to the specified file, or creates it (if the file does not already exist).
`output append` *aFileName* `output append` &*aFileName*	Appends to the specified file, or creates it (if the file does not already exist). An initial & character is ignored.
`output pushnew` *aFileName.ext* `output pushnew` &*aFileName.ext*	Creates *aFileName.ext* if it does not already exist. If you name an existing file, creates a new file with the name *aFileName*_N.*ext*, where N is an integer between 1 and 999 (inclusive) and *aFileName*_N.*ext* does not already exist. An initial & character is ignored.

Concurrent Error Handling

In GemStone/S 64 Bit, the new IFERR command enables the use of 10 post-error command buffers. You can specify a set of Topaz command lines to be executed whenever subsequent commands return an unexpected error or unexpected return value. You can use up to five EXPECTERROR and EXPECTVALUE commands to specify expected errors and return values.

For details, see the discussion of IFERR in the *GemStone/S 64 Bit Topaz Programming Environment Manual*.

GemStone/S 64 Bit also introduces or modifies these related Topaz commands:

- IFERR_LIST — Prints all of the non-empty post-error command buffers.
- IFERR_CLEAR — Clears all the post-error command buffers.
- DISPLAY ERRORCHECK
  OMIT ERRORCHECK — Similar to DISPLAY or OMIT RESULTCHECK, but without the implied `expectvalue true`.

Chapter

8 Changes in Configuration Parameters

This chapter presents the following information:

- A list of configuration options that have new maximum, minimum, or default values in GemStone/S 64 Bit
- A list of 6.1.5 configuration options that are not present in GemStone/S 64 Bit (page 57)
- A list of new configuration options in GemStone/S 64 Bit (page 57)

Changes to Existing Stone Configuration Options

The following Stone configuration options have changed between GemStone/S 6.1.5 and GemStone/S 64 Bit:

DBF_EXTENT_SIZES
Maximum increased from 16384 to 33554432.

GEM_PRIVATE_PAGE_CACHE_KB
Default size increased from 200 to 1000.
Minimum increased from 64 to 128.
Due to changes in how memory is used in GemStone/S 64 Bit, you can now configure smaller sizes.

GEM_TEMPOBJ_CACHE_SIZE
Default changed from 600 to 10000 in 64-bit executables, 500 on 32-bit executables.
Minimum increased from 200 to 2000.
Maximum increased from 20000 to 1000000.
For a discussion of GemStone/S 64 Bit memory architecture, see “Memory Use” on page 13.

SHR_PAGE_CACHE_LOCKED
Default changed from false to true.

SHR_PAGE_CACHE_SIZE_KB
Default increased from 10000 to 750000.
In GemStone/S 6.1.5, the maximum was limited by 32-bit address space. These limitations no longer apply. See the discussion of memory architecture in Chapter 1.

SHR_SPIN_LOCK_COUNT
Default increased from 1200 to 4000 for multi-CPU systems.

STN_DISKFULL_TERMINATION_INTERVAL
Added runtime equivalent #StnDiskfullTerminationInterval.

STN_FREE_FRAME_CACHE_SIZE
In GemStone/S 6.1.5, the default value was 1, which disabled the free frame cache (frames acquired one at a time). The maximum was 1% of the number of frames in the shared page cache.
In GemStone/S 64 Bit, the default is -1, which means disable the free frame cache for shared page caches less than 100MB, or use 10 for shared page caches larger than 100MB.
The maximum is now 63.
To disable the cache, use 0.

STN_FREE_SPACE_THRESHOLD
Added runtime equivalent #StnFreeSpaceThreshold.

STN_MAX_SESSIONS
Maximum increased from 8192 to 10000.

STN_NUM_LOCAL_AIO_SERVERS
Maximum increased from 30 to 256.
The number specified is now used, without the calculated adjustment that was applied in 6.1.5.

STN_PRIVATE_PAGE_CACHE_KB
Default increased from 1000 to 2000.
Minimum increased from 64 to 128.
Maximum unchanged at 524288.

STN_SHR_TARGET_PERCENT_DIRTY
Default increased from 20 to 33.
Minimum increased from 1 to 5.
The runtime equivalent has changed from #StnMntShrPcTargetPercentDirty to #ShrPcTargetPercentDirty.

STN_TRAN_LOG_SIZES
Maximum increased from 2147 to 16384.
With GemStone/S 64 Bit, we recommend tranlog sizes of 100 or larger. The default provided in `system.conf` has changed from 10, 10 to 100, 100.

Default free frame sizes

The calculation of the default free frame sizes has changed to avoid excessively high limits for very large caches. The calculations are the same for cache sizes up to 400MB (50,000 frames), inclusive.

GEM_FREE_FRAME_LIMIT
In GemStone/S 6.1.5, this was always 10% of the number of frames in the main SPC. In

GemStone / S 64 Bit, for main caches larger than 400MB (50,000 frames), the default is 5000 frames.

GEM_PGSVR_FREE_FRAME_LIMIT

In GemStone / S 6.1.5, this was previously always 10% of the number of frames in the remote cache. In GemStone / S 64 Bit, for remote caches larger than 400MB (50,000 frames), the default is 5000 frames.

SHR_TARGET_FREE_FRAME_COUNT

For local caches in GemStone / S 6.1.5, this was always 12.5% of the number of frames in the main cache. In GemStone / S 64 Bit, for local caches larger than 400MB (50,000 frames), the default is 7000 frames.

For remote caches in GemStone / S 6.1.5, this was always 1% of the number of frames in the remote cache. In GemStone / S 64 Bit, for remote caches larger than 800MB (100,000 frames), 1000 frames is used.

Removed Configuration Options

The following 6.1.5 configuration options are not present in GemStone / S 64 Bit:

CONCURRENCY_MODE (NO_RW_CHECKS is always used)
DBF_REPLICATE_NAMES
GEM_ATTACHED_PAGE_LIMIT
GEM_DETACH_PAGES_ON_ABORT
GEM_DETACH_PAGES_ON_COMMIT
GEM_NATIVE_CODE_MAX
GEM_NATIVE_CODE_THRESHOLD
GEM_NOT_CONNECTED_DELTA
GEM_NOT_CONNECTED_THRESHOLD
STN_DEAD_X_LOCKING_ENABLED
STN_GC_SESSION_ENABLED
STN_GC_SESSION_CONFIGURATION
STN_RECOVERY_PAGE_RECLAIM_LIMIT
STN_REMOTE_CACHE_PGSVR_TIMEOUT
STN_REPL_TRAN_LOG_PREFIX
STN_REPL_TRAN_LOG_DIRECTORIES

Added Configuration Options

GemStone / S 64 Bit introduces the following new configuration options:

GEM_FREE_FRAME_CACHE_SIZE

Determines the size of the Gem's free frame cache. When using the free frame cache, the Gem removes enough frames from the free frame list to refill the cache in a single operation. When adding frames to the free list, the Gem does not add them until the cache is full.

A value of 0 disables the free frame cache (the Gem acquires frames one at a time). A value of -1 means use the default value: 0 for caches less than 100 MB and 10 for caches of 100 MB or greater.

Units: Frames
Minimum: -1
Maximum: 63
Default: -1 (cache size=0 for caches less than 100 MB and 10 for caches of 100 MB or greater)

GEM_KEEP_MIN_SOFTREFS

The minimum number of most recently used SoftReferences that will not be cleared by VM markSweep if *startingMemUsed* — the percentage of temporary object memory in-use at the beginning of a VM mark/sweep — is greater than GEM_SOFTREF_CLEANUP_PERCENT_MEM but less than 80%.

In most cases, the default (0) is appropriate and should not be changed.

Runtime equivalent: GemKeepMinSoftRefs
Default: 0
Minimum: 0
Maximum: 10000000

GEM_PGSVR_FREE_FRAME_CACHE_SIZE

Determines the size of the free frame cache used by the Gem's remote page server. Has no effect for Gems that are local to the repository extents (which do not have a page server).

When using the free frame cache, the page server removes enough frames from the free frame list to refill the cache in a single operation. When adding frames to the free list, the page server does not add them until the cache is full.

A value of 0 disables the free frame cache (the page server acquires frames one at a time). A value of -1 means use the default value: 0 for shared page caches less than 100 MB and 10 for caches of 100 MB or greater.

Units: Frames
Minimum: -1
Maximum: 63
Default: -1 (cache size=0 for caches less than 100 MB and 10 for caches of 100 MB or greater)

GEM_PGSVR_UPDATE_CACHE_ON_READ

Determines the read behavior of the Gem's remote page server when pages are read from disk. If this option is set to True, pages read from disk are also added to the shared page cache on the page server's host. If this option is False, pages read are not added to the page server's shared cache.

Has no effect for Gems local to the repository extents (such Gems do not have a page server).

Runtime equivalent: GemPgsvrUpdateCacheOnRead
Default: False

GEM_SEND_STN_MSGS_VIA_PGSVR

Specifies whether a remote Gem should send messages to the Stone via the Gem's page server process. Normally, each Gem sends messages to the Stone via a private socket

connection. For heavily loaded systems, it is less expensive for the Stone if messages are sent via the Gem's page server.

Has no effect for Gems running on the same host as the Stone. In that case, communication is performed using shared memory.

Default: True

GEM_SOFTREF_CLEANUP_PERCENT_MEM

Controls the cleanup of SoftReferences.

If *startingMemUsed* — the percentage of temporary object memory in-use at the beginning of a VM mark/sweep — is less than the value of this option, no SoftReferences will be cleared.

If *startingMemUsed* is greater than the value of this option and less than 80%, the VM mark/sweep will attempt to clear an internally determined number of least recently used SoftReferences. Under rare circumstances, you might choose to specify a minimum number (GEM_KEEP_MIN_SOFTREFS) that will not be cleared.

If *startingMemUsed* is greater than 80%, VM mark/sweep will attempt to clear all SoftReferences.

Runtime equivalent: GemSoftRefCleanupPercentMem
Default: 50
Minimum: 10
Maximum: 80

GEM_TEMPOBJ_AGGRESSIVE_STUBBING

Controls stubbing in in-memory garbage collection.

If instance variable X in object A references object B, and X contains a memory pointer to B, then the reference is *stubbed* by storing the objectId of object B into instance variable X.

When this option is TRUE (the default), references from temporary objects to in-memory copies of committed objects are stubbed whenever possible, during both scavenge and mark/sweep. Also, references from not-dirty in-memory copies of committed objects to other committed objects are stubbed whenever possible. This reduces the number of committed objects forced to stay in-memory, but can slow down garbage collection and subsequent execution.

When this option is FALSE, references from temporary objects to in-memory copies of committed objects are never stubbed. References from not-dirty in-memory copies of committed objects to other committed objects are stubbed after the number of objects flushed during commits reaches a threshold, or if almost OutOfMemory. Many applications will run faster in this configuration, but there is a greater risk of OutOfMemory errors.

Stubbing is always disabled when a commit attempt is in progress, regardless of the setting of this option. Certain objects private to the object manager are always immune from stubbing, and so are references stored into Session State by using `System class >> _sessionStateAt:put:`.

If GEM_TEMPOBJ_POMGEN_SIZE is configured to be more than twice the value of GEM_TEMPOBJ_CACHE_SIZE, then this option is ignored and is always False, to ensure efficient operation of the garbage collector with a very large POM generation area.

Default: TRUE

GEM_TEMPOBJ_INITIAL_SIZE

Has no effect on Solaris and Linux, since those platforms support MAP_NORESERVE on mmap().

On HP-UX and AIX, if less than GEM_TEMPOBJ_CACHE_SIZE, this option specifies the initial amount of memory to allocate with mmap(). When temporary object memory is at least 80% full at the end of a mark sweep, a new memory region is allocated with mmap(), contents of previous memory are copied to the new memory, and the previous memory is released with munmap().

The new memory will be twice the size of previous memory if new memory would be less than 50% of GEM_TEMPOBJ_CACHE_SIZE. Otherwise, new memory is 1.2 times previous memory, up to a limit of GEM_TEMPOBJ_CACHE_SIZE.

Minimum: 2000
Maximum: GEM_TEMPOBJ_CACHE_SIZE
Default: 50% of value of GEM_TEMPOBJ_CACHE_SIZE if GEM_TEMPOBJ_CACHE_SIZE< 100000, otherwise 30% of value of GEM_TEMPOBJ_CACHE_SIZE. If the computed default would be less than 2000, 2000 is used. A value of -1 means use the computed default.

GEM_TEMPOBJ_MESPACE_SIZE

Sets the maximum size (in KB) of the Map Entries space within the Gem's temporary object memory. This value is set when the Gem is initialized (linkable GciInit or rpc GciLogin) and cannot be changed without restarting the Gem process.

If this value is not specified, or the specified value is out of range, the default (0) is used. The default means that the system calculates the size of the Map Entries space based on other memory sizes.

One Map Entry is required for each faulted-in committed object, or for any temporary object that might become committed, referenced from an IdentityBag, or exported to the GCI. One Map Entry occupies 32 bytes.

If a Map Entry is needed and Map Entries Space is full, an OutOfMemory occurs, terminating the session.

Default: 0
Minimum: 1000
Maximum: 1000000

GEM_TEMPOBJ_OOPMAP_SIZE

Sets the size of the hash table (that is, the number of 8-byte entries) in the objId-to-object map within the Gem's temporary object memory. The hash table is completely allocated at GciLogin and consumes 8 bytes per entry. Entries in the hash table are nil, or are pointers to "Map Entries" which are allocated out of the Map Entries space.

This value is set when the Gem is initialized (linkable GciInit or rpc GciLogin) and cannot be changed without restarting the Gem process.

The specified value is rounded up to the next higher power of 2.

If this value is not specified, or if the specified value is out of range, the default is used. The default value (0) means the map is calculated based on other memory sizes.

Default: 0
Minimum: 16384
Maximum: 524288000

GEM_TEMPOBJ_POMGEN_SIZE

Sets the maximum size (in KB) of the POM generation area in the Gem's temporary object memory. This value is set when the Gem is initialized (linkable GciInit or rpc GciLogin) and cannot be changed without restarting the Gem process.

The POM generation area holds unmodified copies of committed objects that have been faulted into a Gem, and is divided into ten subspaces.

If this value is not specified, or if the specified value is out of range, the default is used. The default value (0) means that the allocated POM generation area will be approximately 0.8 times GEM_TEMPOBJ_CACHE_SIZE.

Default: 0
Minimum: 1000
Maximum: 1000000

SHR_PAGE_CACHE_NUM_SHARED_COUNTERS

Number of shared counters available in the shared page cache. On most platforms, each counter consumes 128 bytes of shared memory. On AIX, each counter consumes 256 bytes of shared memory. Shared memory used for shared counters is in addition to the shared memory size specified in SHR_PAGE_CACHE_SIZE_KB.

Default: 1900
Minimum: 0
Maximum: 500000

STN_ADMIN_GC_SESSION_ENABLED

Determines whether the Admin GcGem is started when the Stone is started. (The Admin GcGem performs administrative garbage collection functions such as write set union sweeps; the Reclaim GcGems perform dead object and page reclamation.)

Runtime equivalent: StnAdminGcSessionEnabled
Default: True

STN_COMMIT_QUEUE_THRESHOLD

Determines whether the Stone defers the disposal of commit records, based on the number of sessions in the commit queue. If the size of the commit queue exceeds this threshold, the Stone defers commit record disposal until the commit queue is less than or equal to the value.

This setting is ignored if the commit record backlog exceeds the value of STN_CR_BACKLOG_THRESHOLD.

Runtime equivalent: StnCommitQueueThreshold
Default: -1 (never defer commit record disposal)
Minimum: -1
Maximum: 1024

STN_COMMIT_TOKEN_TIMEOUT

Sets the maximum interval (in seconds) that a session may possess the commit token. If the session possesses the token for longer than this period, the session will be logged off the system and an error message will be written to the Stone log. GcGems of all types are exempted from this timeout.

Default: 0 (stone waits forever)
Minimum: 0
Maximum: 86400
Units: seconds

STN_COMMITS_ASYNC

If STN_COMMITS_ASYNC is set to TRUE, it causes the stone to acknowledge each commit to the requesting session without waiting for the tranlog writes for that commit to complete.

Default: FALSE

STN_CR_BACKLOG_THRESHOLD

Determines the size of the commit record backlog above which the Stone aggressively disposes of commit records. This setting overrides the deferral of commit record disposal provided by STN_COMMIT_QUEUE_THRESHOLD.

The default setting (-1) causes the Stone to use a setting equal to (2 * STN_MAX_SESSIONS). A setting of 0 disables this threshold.

Runtime equivalent: StnCrBacklogThreshold
Default: -1
Minimum: -1
Maximum: 500000

STN_EPOCH_GC_ENABLED

Determines if epoch garbage collection can be run on the system.

Leave this value set to the default (False) unless you plan to run epoch garbage collection on the system. Setting this to True adds a small amount of overhead to commit processing.

Runtime equivalent: StnEpochGcEnabled
Default: False

STN_LOOP_NO_WORK_THRESHOLD

Indicates the maximum number of times the Stone will continue executing its main service loop when there is no work to do. If the Stone loops more than this number of

times and finds no work, the Stone will sleep for up to 1 second. The Stone will immediately wake up when there is any work to be done.

In addition, when this threshold is non-zero, the Stone will not sleep whenever any of the following conditions is true and the no work threshold has not been exceeded:

- A session holds the commit token.
- One or more sessions are waiting in the commit queue.
- One or more sessions are waiting in the run queue.

The default value (0) disables this feature. Setting this option to a non-zero value causes the Stone to consume more CPU.

The Stone cache statistic StnLoopNoWorkThreshold shows the current value of the parameter.

Default: 0
Minimum: 0
Maximum: 536870911
Run time equivalent: StnLoopNoWorkThreshold

STN_MAX_AIO_REQUESTS

Specifies the maximum number of asynchronous write requests the Stone can have pending. If more than this number of asynchronous writes are requested, the Stone will wait (sleep) until one or more of the pending requests have completed.

Asynchronous write requests are only used to write to the current transaction log.

The maximum value allowed depends on the maximum allowed by the UNIX kernel. The maximum value for this parameter allowed by GemStone is the value of _SC_AIO_MAX or 4096, which ever is lower. On some systems (such as Solaris), it is not possible to determine the value of _SC_AIO_MAX. In that case, GemStone imposes a maximum value of 128. Otherwise, the maximum is 4096 or _SC_AIO_MAX, whichever is lower.

For further information on the _SC_AIO_MAX kernel parameter, refer to the UNIX documentation for your system or to the UNIX man page for the sysconf() call.

Default: 128
Minimum: 100
Maximum: MIN(4096, _SC_AIO_MAX) or 128 (see above)

STN_MAX_VOTING_SESSIONS

Specifies the maximum number of sessions that can simultaneously vote on possible dead objects, at the end of a markForCollection or epoch garbage collection. To help prevent the voting on possible dead objects from causing large increases in response time of the system, set this to a value substantially lower than STN_MAX_SESSIONS.

Runtime equivalent: StnMaxVotingSessions
Default: 100
Minimum: 1
Maximum: 1000000

STN_NUM_GC_RECLAIM_SESSIONS

Sets the number of Reclaim GcGems that will be started when the Stone starts. The maximum number of Reclaim GcGems is equal to the number of extents as specified in DBF_EXTENT_NAMES. If the specified value exceeds the number of extents, one Reclaim GcGem will be started for each extent.

Runtime equivalent: StnNumGcReclaimSessions
Default: 1
Minimum: 0
Maximum: 256

STN_OBJ_LOCK_TIMEOUT

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

Default: 0 (stone waits forever)
Minimum: 0
Maximum: 86400

STN_PAGE_REMOVAL_THRESHOLD

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

Default: 40
Minimum: 0
Maximum: 1792

STN_TRAN_Q_TO_RUN_Q_THRESHOLD

Specifies the number of sessions in the commit queue (waiting for the commit token) above which the Stone will allow the remaining sessions in the queue to process unions (read old commit records) while waiting for the commit token.

Example: if this parameter is set to 6 and there are nine sessions in the commit queue, the last three sessions will be allowed to process unions while waiting for the token. If there are six or fewer sessions in the queue, no sessions will process unions.

The first session in the commit queue never processes unions, since it will receive the token when the current commit completes.

Runtime equivalent: StnTranQToRunQThreshold
Default: 6
Minimum: 1
Maximum: 1024

Chapter

9 Changes in Errors

While most error numbers have remained the same between products, many errors have been added, while others have been changed or removed.

This chapter presents the following information:

- A list of errors that are new in GemStone/S 64 Bit
- A list of 6.1.5 errors that are no longer used in GemStone/S 64 Bit (page 69)
- A list of 6.1.5 errors that have been renumbered, renamed, or had changes in arguments (page 69)

New Errors

The following error numbers/mnemonics were not defined in GemStone/S 6.1.5, but are in GemStone/S 64 Bit.

Table 1 New Errors in GemStone/S 64 Bit

Error	Error Name	Definition
1037	#StDBErrStmtNoEffect	Statement has no effect.
2169	#commitPromoteFailed	Unable to promote the commit to a checkpoint.
2388	#rtErrCantReadFile	The system was unable to read the given file. Args: (1) the filename (2) errno
2389	#rtErrFileCorrupt	The given file is corrupted. Args: (1) the filename (2) message
2390	#rtErrGcCommitFailure	A commit during a garbage collection operation failed.

Table 1 New Errors in GemStone/S 64 Bit (Continued)

Error	Error Name	Definition
2391	#rtErrSymbolCreateErr	Creation of new Symbols not allowed at this time. Args: (1) reason (2) symbol value
2393	#rtErrDecodedObjDoesNotExist	The decoded object at the given offset does not exist. Args: (1) Integer value of objectId (2) offset in the byte array
2394	#rtErrMaxSuperclassDepth	An attempt was made to create a subclass too far below Object in the superclass chain. Args (1) receiver (2) max depth
2395	#rtErrReclaimAllMissingGcGem	A reclaimAll operation was attempted but at least one GC session is not running. Ensure that all reclaim sessions and the Admin GC session are running and try the operation again.
2400	#rtErrCantSuspendLogins	Cannot suspend logins. Args: (1) reason
2401	#rtErrWeakDictNeedsInit	GciWeakDictInit() must be called before other weak dictionary functions.
2402	#rtErrSymbolTooLarge	Attempt to create a symbol that is more than 1024 bytes.
2403	#rtErrOmFlushFailed	Cannot commit. Args (1) reason
2404	#rtErrNoExistingSymbol	GciSendMsg failed because there is no existing Symbol that matches the selector String. Args: (1) selector
2405	#rtErrObjMustBeCommitted	The given object is a temporary object but is required to be a committed object. Args: (1) the object (2) details
2407	#rtErrClassIsNp	Object may not be committed, the object's class is NP. Args: (1) an object
2408	#rtErrSuperclassIsNP	Class may not have instances persistent because superclass is NP. Args: (1) a Class
2409	#rtErrContinueTransFail	continueTransaction is not allowed. Args (1) reason (a String)
2410	#rtErrMethodSrcTooBig	Source string for a method is too big. Args (1) source string (2) source string size (3) maxSize
2411	#lgcErrTravBuffSize	A Traversal buffer received over the network exceeded the RPC client's allocated buffer size.
2412	#rtErrAbortWouldLoseData	A method is being run that requires an abort to function; however, an abort would result in lost data as there are modified objects.
2413	#bkupErrNotInProgress	An attempt was made to continue a full backup when no backup is in progress.

Table 1 New Errors in GemStone/S 64 Bit (Continued)

Error	Error Name	Definition
2414	#gciErrCallNotSupported	The GCI call is not supported when invoked from a client user action.
2415	#rtErrRemoveAllIndexesFailed	An attempt to remove all indexes has failed.
2416	#rtErrCollectionWithIncompleteIndex	An attempt was made to create an index on a collection that has incomplete indexes. The incomplete indexes must be removed before creating new indexes.
2417	#rtErrNoMoreSegments	No more segments can be created. SystemRepository has reached maximum size.
2418	#lockErrDeadlock	RcRetry or Application write lock denied due to possible deadlock. Args (1) object upon which lock was requested.
2419	#lockErrTimeout	RcRetry or Application write lock denied due to timeout. Args: (1) object upon which lock was requested.
2420	#lockErrInvalidObject	RcRetry or Application write lock denied; a different object is already registered with the lock queue. Args: (1) details
2421	#authErrProcessSwitch	processor scheduler cannot switch processes while a primitive is within a bypass-authorization block.
2422	#rtErrNotInExportSet	Illegal argument to GciStoreTravDoTravRefs; some objects were not found in the referenced and / or exported set. Args: (1) details
2423	#rtErrGciTravNotLicensed	License does not allow use of Gci Traversal operations.
3010	#otErrRebuildSuccessful	Rebuild of the object table was successful, not trappable in Gemstone Smalltalk.
3013	#otErrCompactSuccessful	Compact of the object table was successful, not trappable in Gemstone Smalltalk.
3015	#bkupErrDisallowed	Full backups are not allowed with reclaim GcGems running.
3022	#abortErrObjAuditFail	Object audit failed. Args: (1) number of objects containing errors. This error is not trappable from GemStone Smalltalk.
3030	#abortErrRecordDeadFail	Record possibleDead failed at end of MFC. Args: (1) reason

Table 1 New Errors in GemStone/S 64 Bit (Continued)

Error	Error Name	Definition
4014	GS_FATAL_ERR_TRANLOG_DIR_FULL	Login denied to other than SystemUser or DataCurator because all tranlog directories or partitions are full. The system is waiting for an operator to make more space available either by cleaning up the existing files (copying them to archive media and deleting them) or by adding a new tranlog directory.
4033	ERR_FINISHED_OBJ_AUDIT_REPAIR	Object audit completed after repair. Args: (1) number of objects containing errors.
4043	GS_ERR_SHRPC_LOST	Network connection to the shrpcmonitor was lost.
4047	#gsErrShrpcLostOtTimeout	LostOt timeout detected when accessing the shared cache.
4054	#gsErrCacheTooBig	GemStone could not start a remote shared page cache because the requested cache size exceeds the license limit.
4055	GS_ERR_GET_TRAN_HINTS_FAILED	StnCallTranSerialization: PageGetTranHints() call failed. Arg (1) the errorCode (int) PAGE_CACHE_HINT_NO_ERROR = 0 PAGE_CACHE_HINT_NO_EXTENT = 1 PAGE_CACHE_HINT_NO_FILE = 2 PAGE_CACHE_HINT_NO_REMOTE_SERVER = 3 PAGE_CACHE_HINT_NO_GEM = 4
4056	GS_ERR_NO_FREE_FRAMES	Gem ran out of free frames - cache must be larger.
4058	#errLostOtHandlingFailed	An error occurred during LostOT handling; view of SharedOt may not be correct. This error indicates that a lostOt was not handled properly on the transition from transactionless or outside of a transaction to inside a transaction.
4067	GS_ERR_VM_OUT_OF_MEM	VM local object memory is full. Args: (1) reason
4068	GS_ERR_CLASS_LOAD_ERROR	Fatal error during class loading, session terminated. Args: (1) reason
4069	GS_ERR_VM_OUT_OF_GSSCOPES	VM ran out of GsScopes memory. Args: (1) reason.

Table 1 New Errors in GemStone/S 64 Bit (Continued)

Error	Error Name	Definition
4070	GS_ERR_VM_REFRESH_FAILED	Another error happened during object memory refresh at a transaction boundary. Object memory is in an inconsistent state and execution cannot continue. Args: (1) other error number.
4148	ERR_LGC_NET_SHUTDOWN	An end of file was received over the GCI network, indicating that the network is being shut down.
6012	#rtErrSignalFinishTransaction	The stone has requested the gem to commit, abort or continue (with continueTransaction) the gem's current transaction. This error is only generated if the session has executed the enableSignaledFinishTransactionError method and is in-transaction at the time stone sends the error.
6013	#rtErrSignalAlmostOutOfMemory	The session's temporary object memory is almost full. The error is deferred if in user action or index maintenance. See method signalAlmostOutOfMemoryThreshold: in class System for more detail. Trappable only by an Exception specifying exactly this error.

Removed Errors

The following errors in GemStone/S 6.1.5 are no longer used.

Table 2 Removed Errors

Error	Error Name
2050	#repErrReplicateOnline
2134	#objErrBadFetchOffset
2156	#repErrReplicateNotMounted
2182	#repErrCantDispose
2380	#rtErrLostOtHandlingFailed
2382	#rtErrBadGcType
2383	#rtErrBadExtentId
2384	#rtErrGcSessionInvalid
2385	#rtErrGcSessionFailed
2386	#bkupErrNoSpc
3016	#bkupErrRestoreCommitFailed
3021	#abortErrFinishedObjAudit

Table 2 Removed Errors

Error	Error Name
3032	#abortErrFinishedObjAuditWithErrors
3033	#abortErrReclaimAllFailure

Changed Errors

Error numbers with new meanings

In the following cases, GemStone/S 64 Bit and GemStone/S 6.1.5 errors with the same number are not identical.

Error #	GemStone/S 6.1.5	GemStone/S 64 Bit
2173	AUTH_ERR_SEG_WRITE_SEG An attempt was made to write to a segment with insufficient authorization.	AUTH_ERR_SEG_LOAD Error loading a Segment into authorization cache. Args: (1) details (2) object whose segment was being loaded.
2174	AUTH_ERR_SEG_READ_SEG An attempt was made to read from a segment with insufficient authorization.	AUTH_ERR_SEG_READ_RECURSION Infinite recursion detected trying to load a Segment. You do not have read authorization to the segment of the Segment. All Segments should be in DataCuratorSegment. Args: (1) segmentId of the Segment.
4147	GS_ERR_REMOTE_GEMS_DISALLOWED GemStone/S Limited Edition does not allow remote sessions.	AUTH_ERR_IN_LOGIN Fatal read authorization error during login. Args: (1) detail.

Renumbered errors

The following errors have new numbers in GemStone/S 64 Bit.

Error Name	GemStone/S 6.1.5	GemStone/S 64 Bit
BKUP_ERR_NO_START	2387	2397
BKUP_ERR_RESTORE_LOG_FAIL	3012	4049
BKUP_ERR_RESTORE_LOG_SUCCESS	3011	4048
BKUP_ERR_RESTORE_SUCCESSFUL	3008	4046

In GemStone/S 64 Bit, the three BKUP_ERR_RESTORE_* errors are fatal errors, since restore now terminates the session.

Also, the error arguments for BKUP_ERR_RESTORE_SUCCESSFUL have been reordered. In GemStone/S 6.1.5, the order was (1) number of objects restored; (2) number of corrupt objects not restored; (3) status.

In GemStone/S 64 Bit 2.2.1, the order is: (1) status; (2) number of objects restored; (3) number of corrupt objects not restored.

Changes in arguments

The following errors have changes in arguments:

OBJ_ERR_NOT_SEGMENT 2011
In GemStone/S 6.1.5, this error returned the (non segment) object. In GemStone/S 64 Bit 2.2.1, it returns two arguments: the segmentId and a reason.

REP_ERR_MAX_EXTENTS 2016
This error now also can be returned if keyfile limits would be exceeded. In GemStone/S 6.1.5, this error returned the maximum number of extents. In GemStone/S 64 Bit 2.2.1, it returns the reason.

RT_ERR_ARG_OUT_OF_RANGE 2061
This error returns an additional argument in GemStone/S 64 Bit 2.2.1, the maximum or minimum value that was exceeded.

AUTH_ERR_SEG_READ 2115
This error returns a third argument in GemStone/S 64 Bit 2.2.1: a detail string.
The first argument used to be the object itself, now the oop is returned.

AUTH_ERR_SEG_WRITE 2116
In GemStone/S 6.1.5, the second argument was the segment. In GemStone/S 64 Bit 2.2.1, it is the id of the segment.

REP_ERR_FILE_ALREADY_EXISTS 2126
In GemStone/S 64 Bit 2.2.1, this error returns an argument, the filename.

OBJ_ERR_MAX_SIZE 2141
This error's arguments have been reordered. In GemStone/S 6.1.5, it was:
(1) the object; (2) the maximum size (3) the specified size
In GemStone/S 64 Bit 2.2.1:
(1) the object; (2) the specified size (3) the maximum size

GCI_ERR_TRAV_BUFF_TOO_SMALL 2217
In GemStone/S 64 Bit 2.2.1, the buffer must be >= GCI_MIN_TRAV_BUFF_SIZE and a multiple of 8 bytes.

RT_ERR_COMMIT_DISALLOWED 2249
In GemStone/S 6.1.5, abort was required. In GemStone/S 64 Bit 2.2.1, the session must log out.

BKUP_ERR_READ_FAILED 2307
In GemStone/S 6.1.5, this error had three arguments: reason, filename and record id. In GemStone/S 64 Bit 2.2.1, it has only one argument, a string description.

RT_ERR_SCHEDULER_DEADLOCKED 2366
In GemStone/S 6.1.5, this error returned just one argument, the process scheduler.
In GemStone/S 64 Bit 2.2.1, it returns an additional argument, an Array of GsProcesses that may be contributing to the deadlock.

GS_ERR_NO_CAPABILITY 4037
In GemStone/S 6.1.5, this error did not have an argument. In GemStone/S 64 Bit 2.2.1, it has one argument, a string.

AUTH_ERR_SEG_LOGIN_SEG 4140
In GemStone/S 6.1.5, this error returned only one argument, the OOP of the bad segment.
In GemStone/S 64 Bit 2.2.1, it returns an additional argument, the segment's id.

Chapter

10 Changes in Cache Statistics

Cache statistics are visible using VSD or via the Smalltalk programmatic interface. In GemStone/S 64 Bit, statistics have been added and removed, and the indexes of all statistics have changed.

Statistics Indexes Subject to Change

The indexes of cache statistics can no longer be assumed to be consistent between releases of GemStone/S 64 Bit. Any code or scripts that reference cache statistics by index must be updated, and should no longer hard-code the cache statistics index number.

You can use the method `System class >> cacheStatisticsDescription` to obtain the correct indexes of cache statistics. In GemStone/S 64 Bit, this method calls a primitive that returns an array of strings; the list of descriptions no longer resides in the text of the Smalltalk method. We recommend that applications using cache statistics call this method on startup and store the resulting names and indexes in a dictionary.

Global Cache Statistics

In GemStone/S 64 Bit, you can use global session cache statistics — user-defined statistics that can be written and read by any Gem on any Gem server. Global session cache statistics are stored in the shared page cache of the Stone, rather than of the machine on which the Gem is running. There are 48 global cache statistic slots available. They are listed in VSD under the Stone's list of statistics.

The following new methods allow you to write and read the global cache statistics:

```
System class >> globalSessionStatAt:
System class >> incrementGlobalSessionStatAt:by:
System class >> globalSessionStatAt:put:
```

System Statistics

In GemStone/S 64 Bit, system statistics are available in VSD for SolarisSystem, HP_System, and AIX_System. These statistics are not available programmatically in Smalltalk.

Changed Cache Statistics

The following statistics have been renamed:

GcSweepCount has been renamed to **GcWsUnionSweepCount**

DeadObjsCount has been renamed to **DeadObjsReclaimedCount**

The following statistics have been renamed, and the units changed to be multiples of 1K (1024):

DeadNotReclaimedSize is now **DeadNotReclaimedKobjs**.

EpochNewObjsSize is now **EpochNewKobjs**.

EpochPossibleDeadSize is now **EpochPossibleDeadKobjs**.

EpochScannedObjs is now **EpochScannedKobjs**.

FreeOopCount is now **FreeOopsK**.

PossibleDeadSize is now **PossibleDeadKobjs**.

Most of the cases in which the statistic **ProgressCount** was modified now use a new statistic, **ProgressKobjs**, which is in units of 1K (1024). The statistic **ProgressCount** still exists and is used. Here is a summary of where these statistics are used.

ProgressCount	ProgressKobjs
▸ During objectAudit, set to number of live data pages at end of OT scan, then decremented as data pages are scanned. ▸ During _findPagesContainingOops:, set to total number of pages in repository and decremented as pages are processed. ▸ During _readObjectTableFromOopNum:toOopNum:..., incremented as OT pages are read, is reset to zero, then incremented again as dataPages (if any) are read. ▸ During readPageRangeForGem:... incremented as pages are read, then reset to zero.	▸ During markForCollection, incremented as live objects marked during the marking phase, reset to zero, and incremented as possible dead objects are computed. ▸ During epochGc, incremented as live objects within epoch are marked, then reset to zero. ▸ During fullBackup and restoreFromBackup, tracks number of objects written to or restored from the backup file/s. ▸ During objectAudit, incremented as OT is scanned, then decremented as data pages are scanned. ▸ During reading and writing an FDC file of oops, incremented as oops are read or written.

PageWrites for AIO page servers now records the total AIO writes.

UserTime and **SysTime** statistics on HP-UX now have nanosecond resolution, which is converted to milliseconds for VSD. These statistics on Solaris already had finer resolution.

TimeInScavenges has been changed to measure in real milliseconds instead of CPU seconds.

The following cache statistics have been modified so that the calculation is more accurate:

TimeWaitingForStone (Gem)
TimeWaitingForCommit (Gem)
TimeProcessingCommit (Gem)
TimeStoneCommit (Gem)

In GemStone/S 6.1.5, the elapsed time in ns was measured, and the statistic was updated every time. This understated the statistic when any samples were less than 1 ms. In GemStone/S 64 Bit, a counter is kept of the cumulative ns spent in the operation, and the total elapsed time is computed from that counter

Added and Removed Cache Statistics

Due to system changes, many 6.1.5 cache statistics are no longer relevant and have thus been removed.

GemStone/S 64 Bit provides new cache statistics to support the current architecture and functionality.

For a comprehensive list of cache statistics available in GemStone/S 64 Bit, see the "Monitoring GemStone" chapter of the *System Administration Guide for GemStone/S 64 Bit*.

Index

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W