GemStone®

GemStone/S 64 Bit Porting Guide

for GemStone/S 64 Bit 1.x Applications

Version 2.2.1

July 2007

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

Preface

About This Manual

This manual describes the changes, and addresses issues in porting applications from GemStone/S 64 Bit 1.1.14 to version 2.2.1. GemStone/S 64 Bit versions 2.0, 2.1, and 2.2 introduced architectural changes and major new functionality. The information in this document is also covered in more detail in the release notes for each version.

While this manual addresses the major issues and changes to consider when converting, 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 64 Bit 1.1.14.

Terminology Conventions

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

Other Useful Documents

This document provides a summary of changes between GemStone/S 64 Bit 1.1.14 and GemStone/S 64 Bit 2.2.1. 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 *Programming Guide for GemStone/S 64 Bit* 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

GemStone Server Changes

This chapter summarizes the differences between GemStone / S 64 Bit 1.1.14 and GemStone / S 64 Bit version 2.2.1.

The most significant architectural difference is the change from 32-bit OOPs to 64-bit OOPs. With this change there have also been related fundamental changes in various size limits and ranges. GemStone / S 64 Bit version 2.2.1 also adds a number of new features and changes. 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.

Architectural Overview

GemStone / S 64 Bit 2.2.1 has been modified to use 64-bit OOPs, avoiding the 2 billion object limit of 32-bit OOPs in earlier versions. 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, 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 10.

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.

GemStone / S 64 Bit 2.2.1 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 version 1.1.14 POM OOP to the equivalent OOP in GemStone/S 64 Bit 2.2.1, the formula is

gs642xOOP `= (64 *` *gs641xOOP*`) - 63`

Special Objects

SmallInteger range change

In version 1.1.14, 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 2.2.1, 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 version 1.x that are within the new GemStone/S 64 Bit 2.x 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 2.x. 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.

Existing instances of SmallFloat can be converted to SmallDoubles by adding 0.0.

Memory Use

The changes in process memory use in GemStone / S 64 Bit 2.x create configuration issues for gem sessions to avoid out of memory errors, while not overallocating memory.

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, a new configuration option has been added, GEM_TEMPOBJ_INITIAL_SIZE (page 29) to specify an initial size, in addition to the maximum specified by GEM_TEMPOBJ_CACHE_SIZE. Only the initial amount of memory is allocated to begin with, and as memory becomes close to full, the cache will be enlarged.

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 version 2.2.1, 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*.

Segments and Security

GemStone / S 64 Bit versions 1.x did not support Segments. Segment protocol has been reimplemented in GemStone / S 64 Bit 2.2.1, with similar behavior 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. Creating a new user via the GBS tools requires a separate operation to create and commit a default segment, which must be pasted as a additional step.
- Login requires that the user logging in have read authorization for both SystemSegment and DataCuratorSegment.
- The meanings of authorization errors has changed, and they may now return the segment ID, rather than the segment itself.

Operations that added privilege requirements in GemStone / S 64 Bit, such as #otherPassword for password changes, still require that privilege, in addition to the appropriate segment authorization.

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.

In GemStone / S 64 Bit 2.2.1, a DataCuratorGroup is provided to simplify granting privileges for Segment creation and other restricted activities.

GemStone/S 64 Bit 2.2.1 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.*

Session Control Changes

To bypass the normal `stopSession:` timeout, particularly 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:` *seconds*

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 2.2.1, 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.

Translating OOP Value to OOP Number

GemStone / S 64 Bit 2.2.1 includes the methods:

```
System class >> _oopHighWaterMark
System class >> _oopNumberHighWaterMark
```

In earlier versions, these returned approximations rather than exact number, for performance reasons. Now, these methods return the exact number, requiring a call the the Stone to complete. New methods have been added to 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 internally for various object tracking tasks. In GemStone / S 64 Bit 2.2.1, hidden sets are also available for customer use. 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.

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 previous products and versions, this signal was sent to administrative sessions only and did not require specific enabling. In GemStone / S 64 Bit 2.2.1, 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.

Indexing

GemStone / S 64 Bit 1.x did not support Indexing. Indexing has been reintroduced in GemStone / S 64 Bit 2.x; the interface 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.

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:
```

IndexManager

GemStone/S 64 Bit indexing 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*.

The IndexManager also provides an optimization for the fast collection sorting mechanism. Rather than using `Collection >> sortWithBlock`, use the method:

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

Using this method, you 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.

ANSI Exception Handling

The ANSI Exception handling framework is now 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 2.2.1 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:
```

Locking

Application write lock

GemStone/S 64 Bit 2.2.1 includes 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

Transaction conflicts

A new transaction conflict type, Write-Dependency, has been added. For more information, see page 16.

Reduced conflict retries now retry multiple times

Reduced conflict logic provides the capability of retrying operations that failed to commit. In earlier GemStone servers, 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 2.2.1, 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 2.2.1 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 abort now fail for uncommitted changes

There are a number of methods, such as `listInstances`, `markForCollection`, etc., which perform an abort and in earlier versions would lose modified data. In version 2.2.1, if there are modifications to persistent objects that would be lost due to the abort, the method now fails to abort and instead returns the new error 2412, `#rtErrAbortWouldLoseData`.

Also, if the session is in manual transaction mode and is in a transaction when the method that aborts is executed, a new transaction is begun before returning. This leaves the session in the same state as it was in when the method was invoked.

Other Changes

Gem-to-gem signal buffer

In previous releases, the gem to gem signal buffer held only one signal; attempts to send another signal before the previous signal was receiver would error.

In GemStone/S 64 Bit 2.2.1, the Stone allows up to 50 pending gem-to-gem signal messages per session before the sending session will receiver the buffer is full error. The limit of 50 applies to signals from all sessions, not a particular sender.

Method behavior changes

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 "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.

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 #....

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.

Only RPC logins are supported from GBS.

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

Comparison operators

GemStone/S 64 Bit version 2.2.1 includes an expanded set and range of Specials, which are immediate objects - that is, the value is encoded directly rather than as an OOP lookup. Immediate objects are inherently canonical, allowing use of == rather than =. However, when these are replicated to the GBS client, they may or may not be immediate; for example, large SmallIntegers on the server may be replicated as client large integers. Client side code that compares numbers that may not be immediate, should always compare using =, rather than ==, even if the server code may safely use ==.

Chapter

2

GemBuilder for C

UserAction Compile and Link Changes

Specific compile and link command lines have changed, as have the recommended compiler / linker versions. Refer to the *GemBuilder for C* manual for GemStone / S 64 Bit for details.

Static user actions, which were not supported in GemStone / S 64 Bit 1.x, are now supported. The details are similar to GemStone / S, but now 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.

Added GCI functions and macros

The following GCI functions and macros have been added in GemStone / S 64 Bit 2.2.1:

`GciClampedTravRefs`
`GciI32ToOop`
`GCI_I64_IS_SMALL_INT` (replaces `GCI_LONG_IS_SMALL_INT`)
`GciInitAppName_` (variant of `GciInitAppName`)
`GciNbClampedTravRefs`
`GciNbStoreTravDoTravRefs`
`GciOldOopToNewOop`
`GciOopToI32`
`GciOopToI32_`
`GciServerIsBigEndian`
`GciSetTraversalBufSwizzling`
`GciStoreTravDoTravRefs`

Removed GCI functions and macros

The following GCI functions and macros have been removed from GemStone / S 64 Bit 2.2.1:

`GciConvertOldOop`
`GciDoubleEncodedLongToOop`
`GciErrMsgSymToFile` (undocumented API)
`GciErrMsgSymToText` (undocumented API)
`GciFetchIdxOop` (undocumented API)
`GciFetchIdxOops` (undocumented API)
`GciFilteredDirtyObjs`
`GCI_IS_REPORT_CLAMPED`
`GCI_LONG_IS_SMALL_INT` (replaced by `GCI_I64_IS_SMALL_INT`)
`GciLongToOop` (use `GciI64ToOop` instead)
`GCI_LONG_TO_OOP`
`GciNetOobHandler` (undocumented API)
`GciOldEncodedLongToOop`
`GciOldEncodedOopToOop`
`GCI_OOP_IS_CHAR16`
`GciOopToDoubleEncodedLong`
`GciOopToLong` (use `GciOopToI64` or `GciOopToI32` instead)
`GCI_OOP_TO_LONG`
`GciOopToUnsignedLong`
`GciPushErrHandler`
`GciReplaceOopsInNsc` (use `GciReplaceVaryingOops`)
`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 2.2.1. 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 64 Bit 1.1.14 name	GemStone/S 64 Bit 2.2.1 name
`GciFetchBytes`	`GciFetchBytes_`
`GciFetchChars`	`GciFetchChars_`
`GciFetchSize`	`GciFetchSize_`
`GciFetchVaryingSize`	`GciFetchVaryingSize_`
`GciInt64ToOop`	`GciI64ToOop`
`GciObjRepSize`	`GciObjRepSize_`
`GciOopToInt64`	`GciOopToI64, GciOopToI64_`
`GciProcessDeferredUpdates`	`GciProcessDeferredUpdates_`

Table 1 Renamed GCI Functions

`GciSetCacheName`	`GciSetCacheName_` This function has a change in behavior, and now returns a Boolean.

Changes in GCI functions

int to int64

Many arguments and return types that were of type `int` in GemStone / S 64 Bit 1.1.14 are now `int64`.

Some shared counter related functions that formerly used `int` now use `int64_t`.

ArraySizeType

In GemStone / S 64 Bit 1.1.14 and earlier, 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 version 2.x.

Traversal buffers

In GemStone / S 64 Bit 2.2.1, 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:

`GciClampedTraverseObjs`
`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 2.2.1. 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
`GciAlteredObjs`	Fewer arguments; since symbol canonicalization is no longer required, the related arguments have been removed.
`GciFetchObjInfo` `GciFetchObjectInfo`	May return a different number of elements in the results for OOP objects, since `sizeof(OopType)` has changed from 4 to 8.
`GciFltToOop`	Now returns the OOP of either a SmallDouble or a Float object.
`GciOopToChar16`	Now converts a JisCharacter object to a 16-bit C character value.
`GciPointerToByteArray`	May return a different type. If the argument is a 64-bit pointer aligned on an 8 byte boundary, or is a 32-bit pointer, the result will be a SmallInteger; otherwise the result will be a ByteArray.

Changes to GCI Structured Types

GemStone / S 64 Bit 2.2.1 includes 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.

To accommodate the new range of SmallIntegers and other changes, many structured data types have been modified. This includes replacing `ArraySizeType` with `int` or `int64`, and use of the new `GciTravBufType`. 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 3 Topaz Changes

New Command Line Argument

In GemStone / S 64 Bit 2.2.1, 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 2.2.1 provides the 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 2.2.1, EXPECTVALUE now accepts additional arguments. For more information, see the *GemStone/S 64 Bit Topaz Programming Environment Manual*.

Concurrent Error Handling

In GemStone / S 64 Bit 2.2.1, 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.

GemStone / S 64 Bit 2.2.1 also includes these related new or modified 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

4

Changes in Configuration Parameters

This chapter presents the following information:

- A list of configuration options which have updated maximum, minimum, or default values between version 1.1.14 and 2.2.1
- A list of version 1.1.14 configuration options that are not present in GemStone/S 64 Bit version 2.2.1 (page 28)
- A list of configuration options that were added between version 1.1.14 and 2.2.1(page 28)

Changes to Existing Stone Configuration Options

The following Stone configuration options have changed between version 1.1.14 and version 2.2.1:

DBF_EXTENT_SIZES
Maximum increased from 16384 to 33554432.

GEM_PRIVATE_PAGE_CACHE_KB
Minimum increased from 64 to 128.

GEM_SEND_STN_MSGS_VIA_PGSVR
Default changed from false to true.

GEM_TEMPOBJ_CACHE_SIZE
Minimum increased from 1000 to 2000.

STN_MAX_AIO_REQUESTS
Default increased from 33 to 128.
Minimum increased from 22 to 100.

STN_MAX_SESSIONS
Maximum increased from 4096 to 10000.

STN_PRIVATE_PAGE_CACHE_KB
Minimum increased from 64 to 128.

STN_TRAN_LOG_SIZES
With version 2.2.1, we recommend tranlog sizes of 100 or larger. The default provided in `system.conf` has changed from 10, 10 to 100, 100.

Removed Configuration Options

The following 1.1.14 configuration options are not present in version 2.2.1:

STN_AIO_WAIT_TIME
STN_MAX_SLEEP_TIME
STN_NUM_LOOPS_PER_NET_POLL
STN_OOB_SOCKET_POLL_INTERVAL

Added Configuration Options

GemStone/S 64 Bit 2.2.1 includes the following new configuration options:

STN_ALLOCATE_HIGH_OOPS

If this value is greater than 0, Stone skips the specified number of non-special object identifiers and starts to allocate object identifiers (GCI OopTypes) for non-special objects at the specified oopNumber + 1.

This option is intended for testing conversion of GCI applications and user actions from GemStone/S 64 Bit v1.x to GemStone/S 64 Bit v2.x. Do not set this option in a production environment. Large values can cause a long delay during the first startstone after the value is changed in the configuration file.

Cannot be changed at runtime.
Default: 0
Minimum: 0
Maximum: 549755813888

STN_COMMITS_ASYNC

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

Default: FALSE

STN_OBJ_LOCK_TIMEOUT

Time in seconds that a session is allowed to wait to obtain one of the special single object write locks. See `System Class >> waitForRcWriteLock:` and `System Class >> waitForApplicationWriteLock:queue:autoRelease:` for more details.

Runtime equivalent: #StnObjLockTimeout
Default: 0 (stone waits forever)
Minimum: 0
Maximum: 86400

STN_PAGE_REMOVAL_THRESHOLD

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

Runtime equivalent: #StnPageRemovalThreshold
Default: 40
Minimum: 0
Maximum: 1792

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.

Chapter 5

Changes in Errors

While most error numbers have remained the same between releases, errors have been added and modified.

New Errors

The following error numbers / mnemonics were not defined in GemStone / S 64 Bit 1.1.14, but are in GemStone / S 64 Bit 2.2.1.

Table 1 New Errors in GemStone/S 64 Bit

Error	Error Name	Definition
1037	#StDBErrStmtNoEffect	Statement has no effect.
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.
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.

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

Error	Error Name	Definition
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.
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.
4043	GS_ERR_SHRPC_LOST	Network connection to the shrpcmonitor was lost.
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.
4147	AUTH_ERR_IN_LOGIN	Fatal read authorization error during login. Args: (1) details
4148	ERR_LGC_NET_SHUTDOWN	An end of file was received over the GCI network, indicating that the network is being shut down.

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

Error	Error Name	Definition
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.

Changed Errors

Error number with new meaning

In the following case, GemStone / S 64 Bit 1.1.14 and GemStone / S 64 Bit 2.2.1 errors with the same number are not identical.

Error #	GemStone/S 64 Bit 1.1.14	GemStone/S 64 Bit 2.2.1
4047	BKUP_ERR_RESTORE_COMMIT_FAILED	GS_ERR_SHRPC_LOSTOT_TIMEOUT LostOt timeout detected when accessing the shared cache

Renumbered and renamed error

The following error is now a fatal error with a different number, and also is differently named:

RT_ERR_LOSTOT_HANDLING_FAILED 2380

is equivalent to:

GS_ERR_LOSTOT_HANDLING_FAILED 4058

Changes in arguments

The following errors have changes in arguments:

REP_ERR_MAX_EXTENTS 2016
This error now also can be returned if keyfile limits would be exceeded. Previously, 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.

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.

BKUP_ERR_READ_FAILED 2307
Previously, 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
Previously, 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.

RT_ERR_OBJ_MUST_BE_COMMITTED 2405
This error has an additional second argument, a details string.

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

BKUP_ERR_RESTORE_SUCCESSFUL 4046
The arguments have been reordered. Previously, 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.

Chapter

6

Changes in Cache Statistics

Cache statistics are visible using VSD or via the Smalltalk programmatic interface. In addition to added and removed statistics reflecting new and obsolete features, some commonly used statistics have required changes to reflect the 64-bit change.

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 2.2.1, 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.

Added and Removed Cache Statistics

To support the new features, cache statistics have been added, and irrelevant ones removed.

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