GemStone®

GemBuilder® for Smalltalk Release Notes

Version 8.1

February 2016

GemTalk™ Systems

GemTalk Systems
15220 NW Greenbrier Parkway
Suite 240
Beaverton, OR 97006

Preface

About This Documentation

These Release Notes describe the changes in the GemBuilder® for Smalltalk version 8.1 release.

For information on installing or upgrading to this version of GemBuilder for Smalltalk, please refer to the *GemBuilder for Smalltalk Installation Guide* for version 8.1.

This documentation is also available on the GemStone Technical Support website.

Terminology Conventions

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

Technical Support

Support Website

gemtalksystems.com

GemTalk's website provides a variety of resources to help you use GemTalk products:

- **Documentation** for the current and for previous released versions of all GemTalk products, in PDF form.
- **Product download** for the current and selected recent versions of GemTalk software.
- **Bugnotes**, identifying performance issues or error conditions that you may encounter when using a GemTalk product.

- **TechTips**, providing information and instructions that are not in the documentation.
- **Compatibility matrices**, listing supported platforms for GemTalk product versions.

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

Help Requests

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

Requests for technical assistance may be submitted online, 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 down. The support website is the preferred way to contact Technical Support.

Website: techsupport.gemtalksystems.com

Email: techsupport@gemtalksystems.com

Telephone: (800) 243-4772 or (503) 766-4702

When submitting a request, please include the following information:

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

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

24x7 Emergency Technical Support

GemTalk offers, at an additional charge, 24x7 emergency technical support. This support entitles customers to contact us 24 hours a day, 7 days a week, 365 days a year, for issues impacting a production system. For more details, contact GemTalk Support Renewals.

Training and Consulting

GemTalk Professional Services provide consulting to help you succeed with GemStone products. Training for GemStone/S is available at your location, and training courses are offered periodically at our offices in Beaverton, Oregon. Contact GemTalk Professional Services for more details or to obtain consulting services.

Table of Contents

Chapter

1

Release Notes for GemBuilder for Smalltalk 8.1

GemBuilder for Smalltalk (GBS) version 8.1 is a new version of the GemBuilder for Smalltalk product, adding support for GemStone/S 64 Bit version 3.3, providing a redesign of the object replication infrastructure, and fixing a number of bugs. Please take time to read through these release notes before installing or upgrading, to acquaint yourself with the changes.

These release notes provide changes between the previous version of GBS, version 7.6.1, and version 8.1. Version 8.0 was a limited-distribution release, containing the first implementation of the redesigned object replication infrastructure. All changes in 8.0 are included in these release notes.

If you are upgrading from a version prior to 7.6.1, please also review the release notes for each intermediate release between your version and 7.6.1, to see the full set of changes.

This release supports both GemStone/S 64 Bit, the 64-bit GemStone/S-based object server, and with GemStone/S, the original 32-bit GemStone object server.

GBS v8.1 is supported with VisualWorks 7.10.1, and cannot be used with VisualWorks 8.x nor versions earlier than 7.9, nor with VA Smalltalk.

To install GemBuilder for Smalltalk 8.1, follow the instructions in the *GemBuilder for Smalltalk Installation Guide* for version 8.1.

Supported Platforms and Versions

The following tables describe the client Smalltalk versions and platforms supported by GBS 8.1, and the GemStone server product shared library versions that can be used with each.

For more details, including the specific required client libraries for each server product and versions, refer to the *GemBuilder for Smalltalk Installation Guide* for version 8.1.

Table 1 Supported GemStone/S 64 Bit Server versions

	VW 7.10.1 32-bit (RPC only)	**VW 7.10.1 64-bit** (RPC, and linked on UNIX)
Windows 8	3.3, 3.2.12	3.3, 3.2.12
Windows 2008 R2	3.3, 3.2.12	3.3, 3.2.12
Windows 7	3.3, 3.2.12	3.3, 3.2.12
Ubuntu Linux 12.04	3.3, 3.2.12	3.3, 3.2.12
Red Hat Linux ES 6.4	3.3, 3.2.12	3.3, 3.2.12
Red Hat Linux ES 6.5	3.3, 3.2.12	3.3, 3.2.12
Red Hat Linux ES 6.7	3.3	3.3
Red Hat Linux ES 7.1	3.3	3.3
SUSE Linux ES 12	3.3, 3.2.12	3.3, 3.2.12
Solaris 10 on SPARC	3.3, 3.2.12	3.3, 3.2.12

Table 2 Supported 32-bit GemStone/S Server versions

	VW 7.10.1 32-bit
Windows 2008 R2	6.7
Windows 7	6.7
Red Hat Linux ES 6.5	6.7
Solaris 10 on SPARC	6.7

Changes and New Feature

The following changes are in version 8.1:

Support for GemStone/S 64 Bit version 3.3

Support for the latest version of the GemStone/S 64 Bit server, v3.3, has been added.

This includes support for the new SmallFraction immediate (special) object.

Fileout of server code improved handling for extended characters

Fileout of GemStone/S 64 Bit server code via GBS now includes a header line with `fileformat utf8`; and GBS fileouts are always encoded in UTF-8 format on all platforms. This ensures that on filein, any code containing extended characters would be unambiguously decoded correctly using either server topaz commands or GBS tools.

Fileout from 32-bit GemStone/S does not include the fileformat header, and is unchanged.

API for Call Tracing that allows specification of filename

Call tracing previously did not allow you to define the name or directory for the file to which the tracing was written. The API for GbsCallTracing has been reviewed and modified. The new API includes the following class methods:

`startCallTracingToFile`
Turns on call tracing and records the traces to file with a default name, `gbstrace-`*timestamp*`.log`, in the current working directory. If call tracing was already on, a new log file is started.

`startCallTracingToFileNamed:` *aString*
Turns on call tracing and records the traces to the specified file. If the file exists, further tracing will be appended to this file.

`startCallTracingToMemoryMegabytes:` *aNumber*
Turns on call tracing and records the traces in a ring buffer of the specified size. The contents of the ring buffer can be written to a default file on request.

`startCallTracingToMemoryMegabytes:` *aNumber* `fileName:` *aString*
Turns on call tracing and records the traces in a ring buffer of the specified size. The contents of the ring buffer can be written to the specified file on request.

`stopCallTracing`
Turns off call tracing, if currently on.

`writeToFile`
Writes the current log to its file, if not already done. This is only useful when using a memory logger. It is safe to send this message regardless of the state of call tracing.

`logMessage:` *messageString*
If call tracing is active, the message is recorded, either to the log file or the ring buffer.

`isCallTracingActive`
Answers whether call tracing has been started and remains active.

In addition, GbsCallTracing has been moved to the GemStone.Gbs namespace.

Changes in Errors

The following are newError classes in GBS, supporting new server errors in v3.3:

GbsRtErrReposUpgradeNeeded
GbsStDBBadLit

The following Error classes are no longer required for server errors, and have been removed:

GbsRtErrMethodSrcTooBig
GbsRtErrNoMoreSegments
GbsRtErrClassIsNp
GbsAbortErrObjAuditFail
GbsBkupErrDisallowed
GbsAbortErrRecordDeadFail

New Object Memory statistics

GBS statistics collection now also collects VisualWorks memory-related statistics, with the type ObjectMemory. The following statistics are now available:

allocFailures
The number of OldSpace allocations that failed to find a sufficiently large data chunk on the threaded free list; hence the allocation was probably satisfied by eating into the contiguous free space between the OT and the data heap.

allocMatches
The number of OldSpace allocations that succeeded with an exact match.

allocProbes
The number of threaded data chunks examined by OldSpace allocations.

allocSplits
The number of OldSpace allocations that succeeded by splitting a data chunk on the free list.

availableContiguousFixedSpace
The size (in bytes) of the largest chunk of contiguous free space in FixedSpace.

availableContiguousOldSpace
The size (in bytes) of the largest chunk of contiguous free space in OldSpace after subtracting off the space reserved for use by the virtual machine. This figure does not include the free space on the threaded free lists. Contiguous space is consumed when an object will not fit in an entry on the threaded free lists.

availableContiguousSpace
The size (in bytes) of the largest chunk of contiguous free space in old and fixed space after subtracting off the space reserved for use by the virtual machine. This figure does not include the free space on the threaded free lists. Contiguous space is consumed when an object will not fit in an entry on the threaded free lists.

availableFreeBytes
The number of free memory bytes available in the heap.

availableFreeEdenBytes
The number of free memory bytes available in eden.

availableFreeFixedSpaceBytes
: The number of free FixedSpace bytes available for use (i.e., threadedDataBytes + contiguousFreeBytes.

availableFreeLargeSpaceBytes
: The number of free LargeSpace bytes available for use (i.e., threadedDataBytes + contiguousFreeBytes.

availableFreeOldSpaceBytes
: The number of free OldSpace bytes available for use (i.e., threadedDataBytes + contiguousFreeBytes - reservedContiguousFreeBytes.

availableFreeOldSpaceBytesLimit
: The LowSpaceSemaphore is signalled by the virtual machine when the availableFreeOldSpaceBytes drops below this limit.

compCodeCacheBytes
: The size of the compiledCodeCache in bytes.

contiguousFreeOldBytes
: The total number of contiguous free bytes in OldSpace. This figure does not include any space on the threaded free lists. Some of this space is reserved for use by the virtual machine.

dynamicallyAllocatedFootprint
: The total size (in bytes) of the dynamically allocated footprint.

edenBytes
: The size of Eden in bytes.

edenUsedBytes
: The number of used bytes in Eden.

edenUsedBytesScavengeThreshold
: The scavenger will be invoked when edenUsedBytes exceeds this threshold.

enumerationCallsPerMillisecond
: How many objects per millisecond were processed in the most recent invocation of allInstancesWeakly:. This can be used to estimate the systems object scanning performance for tasks like incremental marking.

fixedBytes
: The size of FixedSpace in bytes.

fixedSegments
: The number of FixedSpace segments.

fixedUsedBytes
: The usage of FixedSpace in bytes.

fixedUsedObjects
: The usage of FixedSpace in objects.

freePermBytes
: The number of free PermSpace bytes.

incAbortedCount
: The number of times the incremental GC aborted its mark phase.

incGCState
: The current state of the incremental garbage collector. 0 = resting, 1 = marking, 2 = aborting, 3 = sweeping.

incMarkedBytes
: The total bytes of data in the marked objects.

incMarkedObjects
: The number of objects currently marked.

incMarkedWeakBytes
: The total bytes of data in the marked objects that are weak.

incMarkedWeakObjects
: The number of marked objects that are weak.

incMarkStackOverflows
: The number of times the incremental GCs mark stack overflowed.

incNilledBytes
: The number of bytes in the weak objects examined so far.

incNilledObjects
: The number of weak objects examined so far (to see if they contain dead references that need nilling out).

incReclaimedBytes
: The bytes of data reclaimed so far in this collection.

incReclaimedObjects
: The number of objects reclaimed so far in this collection.

incSweepAllocatedBytes
: The bytes of data in the OldSpace objects allocated since the start of the incremental sweep phase.

incSweepAllocatedObjects
: The number of OldSpace objects allocated since the start of the incremental sweep phase.

incSweptObjects
: The number of objects swept so far in this collection.

incUnmarkedObjects
: The number of objects unmarked since aborting the incremental GC.

largeBytes
: The size of LargeSpace in bytes.

largeFreeBytesTenuringThreshold
: The scavenger will tenure LargeSpace objects when the free bytes in LargeSpace drops below this threshold.

largeUsedBytes
: The the number of used bytes in LargeSpace.

largeUsedObjects
: The number of objects housed in LargeSpace.

maximalFreeOldBytes
: The size in bytes of the largest chunk of contiguous free bytes in OldSpace.

nativeStackSpills
: The number of times the native stack was not large enough to hold all the internal representation of Smalltalk process frames, thus forcing a spill of context objects into new space. This number is also incremented every time a new Smalltalk process is forked. Excessively high values should be taken as a hint to either increase the native stack space size via ObjectMemory class>>sizesAtStartup:, or to adopt an implementation strategy that forks Smalltalk processes less often. See also ObjectMemory class>>sizesAtStartup for more information.

newBytesAvailableForStorage
: The number of bytes that can be used to house objects in new space.

numCompactNMethods
: The number of times the compiled code cache was compacted due to lack of space.

numDataCompactions
: The number of OldSpace data compactions that have been performed since the start of the VM.

numGCs
: The number of compacting GCs that have been performed since the start of the VM.

numGlobalGCs
: The number of global GCs that have been performed since the start of the VM.

numIncGCs
: The number of incremental GCs that have been performed since the start of the VM.

numMarkStackOverflows
: The number of times the GC mark stack overflowed. This should be avoided because it makes GC considerably slower. To address this issue, increase the size of new space (eden + survivor).

numScavenges
: The number of new space scavenges that VW has performed since the launch of the VM.

numWeakObjectListOverflows
: The number of times the weak object list overflowed during GC. This should be avoided because then not all weak objects are processed in a single GC. To address this issue, increase the size of compiled code cache space.

oldBytes
: The size of OldSpace in bytes.

oldDataBytes
: The total data bytes in OldSpace, including the threaded free data.

oldRtBytes
: The size of the old remembered table in bytes.

oldRtEntries
: The size of the old remembered table in terms of the total number of entries.

oldRtUsedEntries
: The number of actual entries in the old remembered table.

oldSegments
: The number of OldSpace segments.

oopsLeft
: Estimate of the number of additional objects that can be housed in object memory.

permBytes
: The size of PermSpace in bytes.

permDataBytes
: The total data bytes in PermSpace.

permOTEs
: The number of entries in the PermSpace OT.

reservedContiguousFreeBytes
: The bytes of contiguous free space reserved for use by the virtual machine.

rtBytes
: The size of the remembered table in bytes

rtEntries
: The size of the remembered table in terms of the total number of entries.

rtUsedEntries
: The number of actual entries in the remembered table.

stackBytes
: The size of StackSpace in bytes.

survBytes
: The size of a single SurvivorSpace in bytes.

survUsedBytes
: The number of used bytes in the occupied SurvivorSpace.

survUsedBytesTenuringThreshold
: The scavenger will start to tenure objects when survUsedBytes exceeds this threshold.

threadedDataBytes
: The number of bytes of data in OldSpace's free list.

threadedDataEntries
: The number of data chunks in OldSpace's free list.

threadedOTEntries
: The number of OTEs in OldSpaces free list.

Bugs present in 7.6.1 and fixed in 8.1

Traversal problems for objects with more than 15 named instvars

32-bit GemStone/S only

When an object has more than 15 named instance variables, if the object is split across traversal buffers, it was possible for the bytes to be packed incorrectly, corrupting the replicated object. (#44388)

Replication may modify immutable objects

Fixed with GemStone/S 64 Bit only

When immutable server literals are replicated to GBS, the replicated object was modifiable. (#44681)

Compiler errors when recompiling a method in the debugger

When a server method is recompiled in the debugger, compiler errors were displayed on the transcript rather than inline. (#44381)

This bug was only present with GemStone/S 64 Bit.

Proceeding from some compiler errors resulted in MNU

After proceeding from a compiler error, in some cases it was possible to get an MNU #oopByteAt:. Now, in these cases where you cannot reasonably proceed, a better error is signalled. (#44794)

Inspecting a GsNMethod class method resulted in BadOffset exception

Inspecting a class method did not correctly compute the size of a GsNMethod class method, resulting in a BadOffset exception. (#44312)

Inspector on IdentitySet errored on remove

When inspecting an IdentitySet, using the **Edit > Remove** menu item resulted in an MNU on #'removeAtIndex:. (#44604)

autoMarkDirty does not mark dirty for client-side class change or become

GemStone/S 64 Bit only

When a client replicate has a class change or become operation, this did not cause the client object to be marked dirty if autoMarkDirty is true. Now, the mark dirty has effect. (#45198, #33115)

This kind of change to a client object is not supported with 32-bit GemStone/S.

GbsSessionParameters removeClassConnectorsFor: may fail

Class connectors may be created with the class name as a String or as a Symbol, depending on code paths. If the class connector's #gsName is a String, removeClassConnectorsFor: would fail with an error. (#43149)

Viewing symbol lists for other users did not work correctly

The Launcher menu item **Browse > Symbol Lists** allows you to view the SymbolDictionaries of any user (as long as you have the correct permissions).

When a different user than the current user was selected, viewing an entry could result in a walkback, if the key was not also defined for the current user; or it could display the value for the current user rather than the selected user. (#45421)

Bugs introduced in v8.0 and fixed in v8.1

Finalization queue failures

Finalization queue conditions existed in which an object may fail to be mourned, with the risk of creating an unknown stub. (#45204)

Saving class methods causes hang when debugging within session critical session

When a client class method was modified, GBS needed to acquire the session semaphore for all sessions, since it was not safe to read any potential GemStone mappings for that class. If you were debugging within a session critical block, the class method save would hang. The way GBS reads mappings is now thread-safe without needing the semaphore. (#45046)

GbxUnauthorizedObjectStub objects did not get unexported

When unauthorized stub objects were enabled on the server, replicated, then GCed from the client, they were not removed from the Gem’s export set. (#45678)

Bugs included in the 8.0 Release Notes

Changes in replication infrastructure

This release includes extensive changes in the infrastructure that supports object replication, making the code more modular and extensible. The changes in the public API for replication should be transparent, although there may be minor changes in behavior.

Applications using private GBS protocol should carefully review such usages and test to ensure expected behavior.

Thread-safe replication spec set

The GbsSession variable holding the replication spec that is used for replication from the server is not thread-safe, which created race conditions when multiple sessions use more than one spec set.

To prevent these issues, a new method has been added:

```
GbsSession >> usingReplicationSpecSet: #nameOfSet do: aBlock
```

This method will add the given replication spec into the current thread’s environment under the key #gbsReplicationSpecSet, evaluate *aBlock*, then remove #gbsReplicationSpecSet from that thread's environment. Server interactions will check the thread’s environment for #gbsReplicationSpecSet, and if a replication spec is found, that will be used. Otherwise, the GbsSession variable is used, as in previous versions. (#43500)

Mappings to server objects may be lost if server map grows during server interaction

The GBS server map associates server objects with the replicated client objects (delegates). This map normally contains weak references to allow references to be garbage collected, but is set to strong during server interactions to avoid losing objects during replication. When so many objects are replicated that the server map needs to grow, the extended part is created with weak references. Under some timing conditions, VisualWorks garbage collection can remove the entries in the extended portion of the map during the server interaction. This usually results in a GbsClassGenerationError. (#44700)

Replicating earlier version of mapped class returns instance

Replicating a class that is not mapped, but whose same-named client class is already mapped to another server class, produced an instance of the client class. Now, this will produce a forwarder. (#43732)

Reverted behavior of uncached forwarders to special objects

In v7.6.1, the behavior of #asUncachedForwarder for special objects changed, to return an uncached forwarder rather than the receiver. This had some problems, for example, in creating uncached forwarders to nil. In this release, it has been reverted to the previous 7.6 behavior. (#44494)

Forwarders to large object trees may result in out of memory errors

Forwarders to server objects did not have all references removed, and therefore continued to require memory. Garbage collection removed it from the referenced set, but did not remove it from identity clamps. When many forwarders to large trees of objects were created, there was a risk of Gem out of memory errors. (#40719)

Support for private clamp-by-callback feature

GBS now includes support for the clamp-by-callback server feature. This should be used only under the guidance of GemTalk Engineering.

Failure in creating class on server when very large number of instance variables

If the number of instance variables was so many that the printString of the array of instance variable names was truncated, the class failed to be compiled on the server. (#43397)

Objects that do not inherit from Object not handled by GBS tools

GBS refinements to the VisualWorks tools sent some GBS-specific methods, which are added to Object class. Application classes that did not inherit from Object would encounter does not understand errors. (#44515)