GemStone®

GemStone/S Release Notes

Version 6.5.7.4

December 2010

GEMSTONE S™

INTELLECTUAL PROPERTY OWNERSHIP

COPYRIGHTS

PATENTS

TRADEMARKS

GemStone/S 6.5.7.4 Release Notes

Overview

GemStone/S 6.5.7.4 is a new version of the GemStone Smalltalk object server. This special build release includes new feature to perform repository operations in with object validation, and a workaround for a GcGem crash.

These release notes provide changes between the previous version of GemStone/S, version 6.5.7.3, and version 6.5.7.4. If you are upgrading from a version prior to 6.5.7.3, please also review the release notes for each intermediate release to see the full set of changes.

No separate Installation Guide is provided with this release. For installation instructions, use the Installation Guide for version 6.5.7.

Supported Platforms

GemStone/S version 6.5.7.4 is supported on the following platforms:

▸ Solaris 8, 9 and 10 on SPARC
▸ Red Hat Linux ES 5.0

For more information and detailed requirements for each supported platforms, please refer to the GemStone/S v6.5.7 Installation Guide for that platform.

Changes in this release

"Safe" repository operations

GemStone/S 6.5.7.4 includes new versions of several repository methods, which operate in safe mode - each object is checked that it is valid. By checking each object, any invalid objects in the system are skipped and the operation can proceed without interruption. These safe operations will take longer to complete than the originals.

Garbage Collection with validation

The following new methods perform the repository scan operation in safe mode. As objects on data pages are scanned, each object is validated by lookup up in the object table. If the object table disagrees with the location, and the page is not on the shadow list, the page is reported at the end.

```
Repository >> safeMarkGcCandidates

Repository >> safeFindDisconnectedObjects

Repository >> safeMarkForCollection

Repository >> safeMarkForCollectionWait: waitTimeSeconds
```

For safeMarkForCollection* or safeFindDisconnectedObjects, at the start a message is printing indicating safe mode:

```
[Info]: Safe mode is enabled.  All objects scanned will be
validated with the object table.
```

If invalid objects are found during the scan, a message is printed at the end of the scan:

```
[Warning]: GemStone detected NN pages which are missing from
the scavengable pages list and/or contain one or more shadow or
dead objects.
<list of page IDs>
```

In addition to new Repository protocol, new user actions have been added to the offlinegc.c code to invoke these methods:

```
uaRunSafeFdcAndWriteOopsToFile

uaLoadOopsAndRunSafeMgc
```

Backups with validation

The following new methods perform a backup in safe mode; each object is checked that it is valid, before it is written to the backup file. In regular backups, only objects that are on pages on the shadowed pages list are checked.

The following methods have been added to perform safe backups; they are otherwise equivalent to the corresponding regular backup methods. See method comments in the image for details.

```
Repository >> safeFullBackupTo: fileOrDevice MBytes: mByteLimit

Repository >> safeFullBackupTo: fileOrDevice

Repository >> safeFullBackupCompressedTo: fileOrDevice
```

```
Repository >> safeFullBackupCompressedTo: fileOrDevice MBytes:
mByteLimit
```

On completion, the safe full backup methods print out the following message, to indicate how many invalid objects were not included in the backup:

```
[Info]: Safe full backup statistics: 0 invalid objects on 0
pages were omitted from the backup.
```

These values are also reported in session cache statistics for the backup session:

SessionStat19 - invalid oop count

SessionStat20 - invalid page count

GcGem coredump reclaiming missing dead object

There is an unknown set of conditions under which the system can get into a state in which a dead object, waiting to be reclaimed, is not on the data page indicated by the object table. The dead object is unreachable and no further information is available.

If this corruption exists, the GcGem crashed when it attempted to reclaim the missing dead object. Now, the GcGem will remove the OOP from the object table and from the dead set, but not allow it to be reused. This orphans the OOP, but allows reclaim to continue. Subsequent FDC/MGC runs will not collect this OOP. (#41114)

Risk of objErrAlreadyExists during dead object reclaim

During the time period when an Epoch GcGem is reclaiming dead objects, a session using GBS to create new objects containing circular references may inadvertently fail with error 2105, #objErrAlreadyExists. (#41132)

GcGem may incorrectly vote many objects as not dead

In voting after MFC, the OOPs of GsMethods in the code cache are voted down. However, the OOP computation was incorrect, so if an object existed with the OOP that was computed, it would be voted down. In the rare case that this object existed and referenced many other objects, it could result in very few objects promoted to dead. (#41135)

Extra internal calls impact performance of scan operations

Operations that scan the object table to produce a list of data pages, including backup, markGcCandidatesFromFile: (MGC), list instances, etc., called a function to update internal information on extents. These updates are not needed and unnecessarily slowed performance. (#41040)