GemStone/S 64 Bit™ Conversion Guide

GemStone/S 6.6
to GemStone/S 64 Bit 3.6.1
with pre-release rpm installation

April 1, 2021

INTELLECTUAL PROPERTY OWNERSHIP

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

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 GemTalk Systems 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 GemTalk Systems.

This software is provided by GemTalk Systems LLC 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 GemTalk Systems LLC or any contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

COPYRIGHTS

PATENTS

GemStone software is or has been covered by U.S. Patent Number 6,256,637 "Transactional virtual machine architecture" (1998-2018), Patent Number 6,360,219 "Object queues with concurrent updating" (1998-2018), Patent Number 6,567,905 "Generational garbage collector with persistent object cache" (2001-2021), and Patent Number 6,681,226 "Selective pessimistic locking for a concurrently updateable database" (2001-2021).

TRADEMARKS

GemTalk, **GemStone**, **GemBuilder**, **GemConnect**, and the GemTalk logo are trademarks of GemTalk Systems LLC, or of VMware, Inc., previously of GemStone Systems, Inc., in the United States and other countries.

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

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

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

Microsoft, **Windows**, and **Windows Server** 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.

Ubuntu is a registered trademark of Canonical Ltd., Inc., in the U.S. and other countries.

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

AIX, **POWER6**, **POWER7**, and **POWER8** and **VisualAge** are trademarks or registered trademarks of International Business Machines Corporation.

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

CINCOM, **Cincom Smalltalk**, and **VisualWorks** are trademarks or registered trademarks of Cincom Systems, Inc.

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. GemTalk Systems 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.

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

Preface

About This Documentation

This document explains how to convert an application running on 32-bit GemStone/S on Windows, to GemStone/S 64 Bit version 3.6.1 on a workstation running Red Hat Linux on x86_64 Compatible Systems.

This includes the rpm installation of GemStone/S v3.6.1 and the intermediate versions required for conversion. Installation on Red Hat Linux using rpm is a new capability for these GemStone versions. The specific installation details and recommendations are still under development. The installation and configuration should be implemented under the advice of GemStone Engineering; adjustments may be required. The final recommendations for rpm installation are subject to change before this capability becomes fully public.

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

Help Requests

GemTalk Technical Support is limited to customers with current support contracts. Requests for technical assistance may be submitted online (including by email), or by telephone. We recommend you use telephone contact only for urgent 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

Please include the following, in addition to a description of the issue:

- The versions of GemStone/S 64 Bit and of all related GemTalk products, and of any other related products, such as client Smalltalk products, and the operating system and version you are using.
- Exact error message received, if any, including log files and statmonitor data 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. Overview

Chapter 2. Installing required GemStone/S 64 Bit versions using rpm

Chapter 3. Converting to 2.x

Chapter 4. Converting from v2.4.x to 3.3.x

Chapter 5. Upgrade from 3.3.x to 3.6.1

Chapter

1 Overview

Overview

To upgrade from 32-bit GemStone/S to the most recent version of GemStone/S 64 Bit, v3.6.1, is a multi-step process.

- Move application to Linux.

  Since the GemStone/S 64 Bit server does not run on Windows, you must first install GemStone/S 6.x on Linux and copy the GemStone extent files to the Linux server. You will also need to copy the configuration files; these will require editing for the path and file names. This allows the upgrade process to GS64 to run.

  Prior to upgrading to GemStone/S 64 Bit, it may be helpful to verify that your application clients can interact with the Stone running on Linux.

- Convert to 2.x

  The architectures of 32-bit and 64-bit GemStone are substantially different. For this reason, the conversion cannot be done in place. The upgrade conversion process will construct a new GemStone/S 64 Bit repository, using the 6.x repository as a template.

  The 32-bit to 64-bit conversion is supported to GemStone/S 64 Bit 2.x versions. To convert to the most recent version of GemStone/S 64 Bit, you will first perform the conversion to version 2.4.8.

  Prior to the conversion and upgrade, you should determine if any of your application classes will require conversion, and decide if you want to convert instance of reimplemented classes such as SmallFloat.

- Convert to 3.3.x

  GemStone/S 64 Bit version 3.0 introduced substantial changes in classes and methods. This requires a second conversion step. This conversion is done in place; however, it requires that you recompile all classes, methods, and blocks, among other changes.

  The GemStone/S 64 Bit 2.x to 3.x conversion is supported to GemStone/S 64 Bit 3.3.x versions. To convert to the most recent version of GemStone/S 64 Bit, you will next perform the conversion to version 3.3.9.

Prior to the conversion and upgrade, you should determine your recompile strategy, and examine your application for complex blocks that cannot be converted automatically.

- Upgrade to 3.6.1

  Finally, you will upgrade from 3.3.9 to 3.6.1, which is a simple upgrade.

Conversion Strategy

While GemStone/S is similar to GemStone/S 64 Bit, there are a number of differences in code and administration. Conversion to GemStone/S 64 Bit requires modifications to your existing application.

Before conversion, read through these installation instructions, and identify the changes that you may need to make in your application.

We recommend that you perform the upgrade conversion and upgrade at least twice: first a pilot conversion, prior to performing the production conversion.

If possible, the pilot conversion should be done on a copy of your production system. During the pilot conversion, the database integrity is verified before and after conversion, and growth information on the repository is determined that will allow you to configure the final system correctly.

The GemStone/S 64 Bit repository will be substantially larger than the GemStone/S 6.x repository. Repository size increases between 20% and 200% or more are likely, depending on the nature of the data in the repository.

Installation via rpm and existing documentation

GemStone traditionally allows an application, and the configurations defaults to using, a setup with the extent files, transaction logs, and process log files located within the GemStone installation tree.

The rpm installation provides a shared GemStone installation tree that is owned by root, and while users can read and execute the GemStone executables, the installation tree is expected to be sharable between applications, and not have application-specific files located within the installation tree.

The existing public documentation, for simplicity in the examples, assumes the traditional default locations. When reading existing documentation, be aware that these defaults do not apply in an rpm installation.

Note

Installation on Red Hat Linux using rpm is a new capability for these GemStone versions. The specific installation details and recommendations are still under development. The installation and configuration should be performed under the advice of GemStone Engineering; adjustments may be required. The final recommendations for rpm installation are subject to change before this capability becomes fully public.

For the upgrade process, keep the following in mind:

- you must have a system configuration file with the extent names and tranlog directories edited to **not** reference $GEMSTONE. We recommend either entering the specific directory path and filename, or defining an instance variable such as $GEMSTONE_DATA.

- the startstone and pageaudit utilities must be instructed to use this configuration file. This is done by using the **-z** or **-e** argument to the utilities.

  Alternatively, you can define the environment variable GEMSTONE_SYS_CONF or GEMSTONE_EXE_CONF, that defines the path and filename for this configuration file. This may simplify administration for the final application. For clarity in examples in the upgrade instructions in this document, **-z** or **-e** has been used.

- The startstone and pageaudit utility must also be instructed with the location for the log file. This is done using the **-l** argument. In older versions this must be the path and filename, in later versions it may be a filename or a directory.

  Alternatively, you can define the environment variable GEMSTONE_LOG, with the location for this log file.

File permissions to allow multi user access

NetLDI

The instructions for setting up the NetLDI configuration and authentication mode make assumptions about file ownerships that are not valid in an rpm installation.

During the conversion process, it is not necessary to have the NetLDI running; you may login linked (using topaz -l) to perform audits and other validation.

From the basic rpm installation, you may login RPC provided that you start the NetLDI in guest mode with captive account.

If your security requires the NetLDI to perform UNIX authentication, you will set the s bit for the `netldid` executable during the installation process. This is only required in a system in which multiple UNIX userIds will be logging into GemStone.

Refer to the *System Administration Guide* for v3.6, chapter 4, for more information.

Extent file access

All UNIX userIds who will execute code that logs into GemStone must have write access to the extent files.

In a default rpm installation, all users who will log in should belong to a particular group, and this group must be given write access to all of the extent files.

For example

```
unix> chmod 660 extent0.dbf
unix> chgrp gsgroup extent0.dbf
```

Refer to the *System Administration Guide* for v3.6, chapter 4, for more information.

Detecting Oversize Methods

Methods larger than 65K are not supported in GemStone/S 64 Bit, and cannot be loaded or executed. If your GemStone/S version 6.x repository contains methods larger than this limit, these methods must be factored into multiple smaller methods before the repository is converted; this task cannot be done after conversion.

GemStone/S 64 Bit 2.4.8 provides a script to detect if any methods exceed the size limit. To use this script, the GemStone/S version 6.x repository that you wish to verify must be

running, and not yet converted to GemStone/S 64 Bit, and GemStone/S 64 Bit 2.4.8 must have been installed.

1. Ensure that a stone is running on the GemStone/S version 6.x repository.
2. Verify that GemStone/S 64 Bit 2.4.8 is installed
3. Start topaz, and log in to the GemStone/S version 6.x repository as SystemUser.
4. Execute the script `methodsTooLargeFor64.topaz`. For example,

   `topaz 1> input` *gs64install*`/upgrade/methodsTooLargeFor64.topaz`

   where *gs64install* is the installation directory for GemStone/S 64 Bit 2.4.8.
5. Wait for the scan to complete. This will take some time as it scans all extents for instances of GsMethod. When it completes, log out and exit topaz.
6. The script prints the results of the scan both to standard out and to a text file `methodsToLargeFor64Report.log`, in the working directory. This report provides details on any methods that are exceed the size limit, including the current size, class and selector. If there are methods that exceed the size limit, factor these into multiple smaller methods and commit.

   Execute a markForCollection and allow reclaim to complete, before rerunning the script to verify that the modified methods are now of appropriate size. Otherwise, the script may detect old instances of methods that have not yet been garbage collected.

Conversion Procedure

The following steps perform the installation, conversions, and upgrade.

1. Install the required GemStone versions on the Linux server

Chapter 2, "Installing required GemStone/S 64 Bit versions using rpm" provides configuration information for the GemStone host machine, as well as installation directions.

After your machine is configured, you will install the GemStone software using rpm. You will need to install the following versions:

- Install GemStone/S v6.6 on Linux
- Install GemStone/S 64-Bit v2.4.8 on Linux
- Install GemStone/S 64-Bit v3.3.9 on Linux
- Install GemStone/S 64-Bit v3.6.1 on Linux

2. Move the 32-bit GemStone application to Linux

With rpm package installation, the extent, transaction logs, configuration and log files may not be located in the previous default location of $GEMSTONE/data.

- Select the directory locations for the extent file, tranlogs, and log files, and copy the extent files to this location. Provided the Stone on Windows was shut down cleanly, you do not need to copy the transaction files.

  Alternately, after you have configured the server on Linux, you may restore a backup of the application into the GemStone server running on Linux.

- Create the system configuration file containing the appropriate settings for DBF_EXTENT_NAMES and STN_TRAN_LOG_DIRECTORIES, along with other configuration parameters required for your application, such as SHR_PAGE_CACHE_SIZE_KB.

- Set the environment for the v6.6 installation on linux:

```
unix> export GEMSTONE=/usr/lib/gemstone/6.6.0
unix> export PATH=$GEMSTONE/bin:$PATH
```

- Start the stone using startstone, with the **-e** or **z** and **-l** arguments:

```
unix> startstone -z configFileDir/configFileName.conf
    -l logFileDir/stoneLogfileName.log  stoneName
```

- Ensure that your application is running correctly on GemStone v6.6 on Linux. In particular, you should run `pageaudit` and `objectAudit`.

3. File out modifications to GemStone classes

File out any modifications or additions you made to methods in GemStone kernel classes.

You will need to carefully compare these changes with GemStone/S 64 Bit 3.6.1 kernel methods, to determine whether your changes are still necessary or appropriate. For a listing of GemStone/S 64 Bit Release Notes, see GemStone/S 64 Bit Release History.

4. Verify that you have no methods larger than 64K.

Before conversion, you must verify that you have no methods that are larger than 64K. Oversize methods do not cause an error in the conversion, but cannot be loaded or executed in the converted repository.

GemStone/S 64 Bit provides a tool to assist in detecting oversize methods, see “Detecting Oversize Methods” on page 9

5. Verify returned value from disallowUsedPasswords is Boolean

Verify that your repository returns a boolean for this setting. Execute:

```
AllUsers disallowUsedPasswords
```

If result is neither true nor false, then before upgrade, execute:

```
AllUsers disallowUsedPasswords: trueOrFalse.
System commitTransaction.
```

6. Remove indexes on instances of reimplemented server classes

If you have indexes that include instance of any of the following classes, these indexes will need to be removed and rebuilt following the conversion, to update information within the indexing structures.

```
DoubleByteString
DoubleByteSymbol
Float
LargeNegativeInteger
LargePositiveInteger
QuadByteString
ScaledDecimal (if you will be using the new ScaledDecimal implementation)
SmallFloat
```

If unsure, it is safer to remove the index rather than to risk incorrect indexed query results. You may remove these indexes prior to upgrade, or as part of application filein after upgrade.

7. Reset SystemUsers's password

Reset SystemUser password to the default, 'swordfish':

```
topaz 1> printit
(AllUsers userWithId: #SystemUser) password: 'swordfish' .
System commitTransaction.
%
```

The upgrade script logs in with the SystemUser account and the default password, and resets the password for DataCurator and GcUser.

Ensure that you have no users logged into your v6.6 system.

8. Perform the conversion and upgrades

- Perform the 6.6 to 2.4.8 conversion, following the instructions in "Converting to 2.x" on page 21.
- Perform the 2.4.8 to 3.3.9 conversion, following the instructions in "Converting from v2.4.x to 3.3.x" on page 29
- Perform the 3.3.9 to 3.6.1 upgrade, following the instructions in "Upgrade from 3.3.x to 3.6.1" on page 37.

Chapter

2 Installing required GemStone/S 64 Bit versions using rpm

This chapter describes the procedure for installing GemStone/S v6.6 and GemStone/S 64 Bit™ versions 2.4.8, 3.3.9, and 3.6.1 on a single host. It is expected that you install all versions at the same time.

Note that installation using rpm is a new capability for GemStone software. The specific installation details and recommendations in this document may require modification. Installation and configuration should be implemented under the advice of GemStone Engineering; adjustments may be required.

Check the System Requirements

Before you install GemStone/S 64 Bit, ensure that the following system requirements are satisfied.

Platform

- System with an x86_64-compatible processor.

RAM and Swap space

- While small installations can run on systems with only a few GB of physical RAM, increasing RAM is important for GemStone performance. Total swap space should be at least equal to the amount of RAM. Due to the way GemStone uses memory, systems with insufficient swap space allocated have a risk of memory errors even if there is available RAM.

Disk space

- Space for the installed distribution files—approximately 2GB to install all packages.
- Additional disk space as required for your repository extents and other files.

  The repository files should be located on a disk drive that does not contain swap space. Use of multiple disk drives is advisable for servers.

Operating system

- Red Hat/CentOS Linux ES 8.1
  kernel version 4.18.0-193.6.3.el8_2.x86_64 and glibc-2.28-101.el8.x86_64
- Red Hat/CentOS Linux ES 7.8
  kernel version 3.10.0-693.11.6.el7.x86_64 and glibc-2.17-292.el7.x86_64

Note that GemStone/S 64 Bit v3.6.x will not run on Red Hat Linux ES 6.x.

Debugger

A C debugger allows C-level stack traces when a GemStone error occurs, or when using the **pstack** command. While not required for GemStone execution, it is strongly recommended that gdb be installed.

C/C++ Compiler

GemStone requires a C/C++ compiler only if you are developing C or C++ code for user actions or for a C or C++ application; as described in the *GemBuilder for C* manual. This compiler is required only for development work, not for execution.

X Windows

An X Windows server allows you to use GemStone's graphical VSD application on Linux. Alternatively, GemStone statistical data may be viewed on Windows, by transferring the data files, or by mounting the file system on Windows. X Windows is not required for GemStone execution.

Configure the Operating System

The kernel must be configured to support shared memory and semaphores. See your operating system documentation for further information. These requirements apply both to server nodes and to client nodes.

1. Shared memory

The upper limit for shared memory single segment size and total usage should be set to values larger than your desired Shared Page Cache size, and not more than 75% of your real memory size.

For GemStone/S v6.6, the maximum memory limit is 4GB.

The single segment maximum size, `shmmax`, is set in bytes, and the total shared memory limit, `shmall`, is configured in pages, with a base page size of 4KB. Note that the results of ipcs may be reported in kbytes.

For example, if you have 8192 MB of real memory:

```
8192 MB * .75 = 6144 MB
6144 MB * 2^20 = 6442450944 bytes
6442450944 / 4K = 1572864
```

To set shared memory sizes, you would append the following text to the `/etc/sysctl.conf` file. The settings are read from this file during the boot process.

```
# Shared Memory setting for GemStone
kernel.shmall = 1572864
kernel.shmmax = 6442450944
```

For more details, consult your Linux operating system documentation.

2. Semaphores

You may need to increase the settings for semaphores. These settings are configured by setting `kernel.sem` to a 4-element array, with the equivalent to the old semmsl, semmns, semopm, and semmni. For example, append the following to the `/etc/sysctl.conf` file.

```
kernel.sem=1000 512000 64 2048
```

The first element sets the maximum number of semaphores per id (per semaphore set). This parameter limits the number of GemStone sessions that can log in to a particular Stone and connect to its shared page cache.

On the Stone's node, this parameter must provide **two** semaphores for each user who will log in to that Stone from any node plus an overhead of **four**. In distributed systems, nodes that have only user sessions must provide **two** semaphore for each user session on that node plus an overhead of **one**.

The number of semaphores actually requested for a particular shared page cache depends on the GemStone configuration file read by the process that starts the cache and is (SHR_PAGE_CACHE_NUM_PROCS * 2) + 1.

The second value sets the total number of semaphores in the system, which must be increased to along with the first.

3. File Descriptors

Each user session requires two file descriptors, and others are needed for extents, transaction logs, and other overhead. The default setting for `fs.file-max` setting is usually sufficient.

4. Locking the Shared Page Cache in memory

If you intend to lock the shared page cache into memory via the stone configuration option SHR_PAGE_CACHE_LOCKED, then the linux user starting the stone must either have the Linux capability CAP_IPC_LOCK, or have a RLIMIT_MEMLOCK resource limit set greater than the size of the SPC.

5. OOM Killer

If your system runs low on memory, the Linux OOM killer may select GemStone processes to terminate. To protect the shared page cache and other critical GemStone processes, each GemStone process's oom_score_adj, which is used to select processes to terminate, is adjusted. If the userid that will be running the server processes has the CAP_SYS_RESOURCE privilege, critical GemStone processes have their oom_score_adj reduced, making them safer; if the user does not have CAP_SYS_RESOURCE, then non-critical processes such as Gems have their score increased, so they will be selected rather than more critical processes.

To set CAP_SYS_RESOURCE on kernels v2.6.32 and later, set the capability on the executables:

a. Install `libcap2-bin`

b. `for i in pgsvrmain gem stoned shrpcmonitor; do sudo setcap cap_sys_resource=pe $GEMSTONE/sys/$i ; done`

c. `for i in startstone topaz; do sudo setcap cap_sys_resource=pe $GEMSTONE/bin/$i ; done`

6. PAM

If you are using UNIX authentication for GemStone logins (v 3.x), or if you run NetLDI as root with setuid (i.e. not in guest mode), you must have PAM (Pluggable Authentication Module) configured on the server. You may include a specific GemStone authorization service name, or allow the default "other" authentication definitions to be used.

PAM authentication definitions are in files under the directory `/etc/pam.d`. Alternatively, they can be lines in the configuration file `/etc/pam.conf`, but this usage is deprecated on many distributions. On these distributions, the presence of the `/etc/pam.d` directory will cause `/etc/pam.conf` to be ignored.

The specific GemStone service file names are `gemstone.gem` for user authentication, and `gemstone.netldi` for a NetLDI running with authentication.

The libraries that are specified in the stack depend on how you are configuring PAM to perform the authentication. The examples below are for PAM configured to invoke LDAP for authentication.

For GemStone UNIX authentication, which uses PAM, to authenticate via LDAP, create a file named `/etc/pam.d/gemstone.gem` with the following contents:

```
auth                required            pam_ldap.so
```

For NetLDI authentication, again using LDAP, create a file named `/etc/pam.d/gemstone.netldi` with the following contents:

```
auth                required            pam_ldap.so
```

Red Hat, by default, installs a file `/etc/pam.d/other` which disables "other" authentication. You can allow the "other" authentication stack to be used for GemStone authentication by ensuring that the file `/etc/pam.d/other` has the following contents:

```
auth                required            pam_ldap.so
```

Consult your System Administrators for more information on how authentication is handled on your system.

7. Huge Memory Pages

Details on configuring Transparent Huge Memory Pages and Huge Memory Pages is omitted from this guide. After the application is upgraded, see the *Installation Guide* for v3.6 if you wish to configure huge memory pages.

8. System clock

The system clock must be set to the correct time. When GemStone opens the repository at startup, it compares the current system time with the recorded checkpoint times as part of a consistency check. A system time earlier than the time at which the last

checkpoint was written may be taken as an indication of corrupted data and prevent GemStone from starting. The time comparisons use GMT.

9. TCP keepalive option

GemStone processes ordinarily use the TCP `keepalive` option to determine how long they will wait after communications activity ceases unexpectedly. This setting can be useful for reaping stale RPC Gems, but the operating system default may not be appropriate for this purpose. For further information, refer to your operating system documentation.

10. Unset LD_BIND_NOW

On some Linux distributions, setting the environment variable LD_BIND_NOW may result in process startup failures due to loading incorrect shared libraries.

11. Verify TCP/IP

To run GemStone, TCP/IP must be functioning, even if your machine is not connected to a network.

Verify that TCP/IP networking software is functioning:

```
unix> /bin/ping hostname
```

where *hostname* is the name of your machine. If **ping** responds with statistics, TCP/IP is functioning.

Install the GemStone Servers using rpm

1. Log in as root.
2. Perform the installation.

   The rpm installation files should be installed in this order, to avoid warnings from the installer.

   ```
   unix> rpm -i GemStone-6.6.0-1.i686.rpm
   unix> rpm -i GemStone64-2.4.8-1.x86_64.rpm
   unix> rpm -i GemStone64-3.3.9-1.x86_64.rpm
   unix> rpm -i GemStone64-3.6.1-1.x86_64.rpm
   ```

   The rpms will install the GemStone product tree to the following locations:

   ```
   /usr/lib/gemstone/6.6.0
   /usr/lib64/gemstone/2.4.8
   /usr/lib64/gemstone/3.3.9
   /usr/lib64/gemstone/3.6.1
   ```

3. The directories:

   ```
   /opt/gemstone/locks
   /opt/gemstone/log
   ```

   are created as empty directories by the installation, if they do not already exist.

   As root, ensure that the users who will be starting the Stone and NetLDI have write permission, or belong to a group with write permission, for these directories.

After the rpms have been installed, change the group of these directories

```
unix> chgrp /opt/gemstone/locks targetGroupName
unix> chgrp /opt/gemstone/log targetGroupName
```

or alternatively, change the world permissions to allow read/write:

```
unix> chmod o+rw /opt/gemstone/locks
unix> chmod o+rw /opt/gemstone/log
```

4. If you will be running the NetLDI will authentication, and if multiple UNIX userIds will be logging into GemStone, then set the s bit for the `$GEMSTONE/sys/netldid` executable. This allows an administrative UNIX userId to start a NetLDI process that will run as root.

   ```
   unix> chmod u+s $GEMSTONE/sys/netldid
   ```

5. Install the keyfiles in the respective directories.

   Each version of GemStone must have a key file for the correct version of GemStone and for the appropriate platform. The keyfile must be located where GemStone can find it on startup.

   It is recommended that the keyfiles for each version be installed in the `sys` directory of the product tree for that version. This is not required: the keyfiles can also be kept in a different location, and the specific keyfile path referenced by the KEYFILE configuration parameter in the configuration file used by the **startstone** command for that version.

   Copy the respective keyfiles to the following locations:

   ```
   /usr/lib/gemstone/6.6.0/sys/gemstone.key
   /usr/lib64/gemstone/2.4.8/sys/gemstone.key
   /usr/lib64/gemstone/3.3.9/sys/gemstone.key
   /usr/lib64/gemstone/3.6.1/sys/gemstone.key
   ```

   These keyfiles should have read-only permissions.

6. The GemStone servers are now installed.

   With an rpm installation, do **not** run the `installgs` executable. The options in this script are not appropriate for an rpm installation.

File Permissions for Multiuser environment

GemStone traditionally allows an application, and the configuration defaults to using, extent files, transaction logs, and process log files located within the GemStone installation tree.

The rpm installation provides a shared GemStone installation tree that is owned by root, and while users can read and execute the GemStone executables, the installation tree is expected to be sharable between applications, and not have application-specific files located within the installation tree. This places some constraints on how security is maintained while allowing multiple users to access the application.

Refer to the *System Administration Guide* for v3.6, chapter 4, for more information on NetLDI and extent file and executable permissions. Note that the explanations and recommendations in this chapter do not reflect the configuration with an rpm installation.

NetLDI

You will only need to run a NetLDI process if you will be logging in RPC to the GemStone server; it is not needed for the conversion or upgrade.

The NetLDI process can run in several modes; with authentication, or in guest mode with captive account (recommended). No further configuration is needed to run in guest mode with captive account; the NetLDI is started with the appropriate arguments.

If you will be using NetLDI with authentication, the root user will need to set the s bit for the `netldid` executable. See the *System Administration Guide* for v3.6 for details on NetLDI configuration.

To set the s bit, execute:

```
unix> chmod u+s /usr/lib64/gemstone/3.6.1/sys/netldid
```

Extent Files

All Linux userIds who will login to the application must have write access to the extent files.

In a default rpm installation, all users who will login to the application should belong to a particular group, and this group must be given write access to all of the extent files.

For example

```
unix> chmod 660 extent0.dbf
unix> chgrp gsgroup extent0.dbf
```

Define the NetLDI Service name, if necessary

A NetLDI is required for some local and all remote sessions to log into GemStone, and it can be resolved by name or directly by port number. If you are defining NetLDI services by name, the same NetLDI service name and port number must be defined in system services database for the Stone's node, and for all remote nodes.

NetLDI access by port number

Accessing the NetLDI by port number rather than by name avoids the need to update the services database, but does require that you explicitly specify this same port when the NetLDI is started up, and include the NetLDI port specification in login parameters. This is normally simplified by defining a GEMSTONE_NRS_ALL environment variable; for details, see the *System Administration Guide*. Using port numbers is not available in 32-bit GemStone/S v6.6.

NetLDI access via service name

The following uses the default name `gs64ldi`; the default in 32-bit GemStone/S is `netldi66`.

1. Determine whether the `gs64ldi` service is already defined. How to do this will depend on how your system is set up. The GemStone distribution includes an executable that will allow you to do this:

   ```
   unix> $GEMSTONE/install/getservbyname gs64ldi
   s_name=gs64ldi s_port = 50377 s_proto = tcp
   ```

If `gs64ldi` is defined, skip the rest of this procedure, you do not need to add an entry for it.

2. If `gs64ldi` is not defined, add an entry similar to the following to the system services database:

   ```
   gs64ldi 50377/tcp #GemStone/S 64 Bit 3.6.1
   ```

   Choose a port number that is not being used by another service. The port number should be in the range 49152 <= port <= 65535.

3. In a multiple node configuration, the UNIX system administrator should update the system services database for each machine. This includes Windows client machines as well as UNIX nodes. The port number must be the same for every machine.

Chapter

3 Converting to 2.x

This chapter describes how to convert a 32-bit GemStone/S installation to GemStone/S 64 Bit 2.4.x.

The architectures of 32-bit and 64-bit GemStone are substantially different. For this reason, the conversion cannot be done in place. The upgrade conversion process will construct a new GemStone/S 64 Bit repository, using the 6.x repository as a template.

Conversion from 32-bit GemStone/S to GemStone/S 64 Bit is a multi-step process. To avoid the risk of inconsistency, the conversion steps record their status in the $upgradeLogDirectory. If you have problems with one step of the conversion process, you may need to go back to the beginning and repeat the earlier steps, to allow the conversion to run correctly.

Before Conversion to GS/64

As described in Chapter 1, before beginning the conversion, you should ensure:

- you have filed out any modifications to GemStone classes
- you have no methods larger than 64K, and have split any larger methods into multiple smaller methods.
- the system configuration file used by the v6.6 installation does not refer to the extent file locations using the environment variable `$GEMSTONE`.

  With rpm installation it is not recommended to include $GEMSTONE in the extent file location path. For conversion, this is specifically limited; the conversion will not proceed with `$GEMSTONE` in the extent file path/s.

Conversion to GS/64

Preprocessing in Old Environment

1. Set up the GemStone/S 6.6.0 environment

The first part of the conversion runs in the GemStone/S 6.6.0 environment.

- Set GEMSTONE to `/usr/lib/gemstone/6.6.0`, and set the path to include the GemStone executable directory

```
unix> export GEMSTONE=/usr/lib/gemstone/6.6.0
unix> export PATH=$GEMSTONE/bin:$PATH
```

- Start the v6.6.0 stone, if it is not already running.

```
unix> startstone -z configFileDir/configFileName.conf
    -l logFileDir/stoneLogfileName.log  stoneName
```

2. Set GEMSTONE_64 and upgradeLogDir Environment Variables

Define an environment variable, `$GEMSTONE_64`, to point to the 2.4.8 product tree. At this point, do not reset `$GEMSTONE` or update the path to point to v2.4.8.

Also define the environment variable that will hold the upgrade log files to a temporary directory for which you have write permission.

```
unix> export GEMSTONE_64=/usr/lib64/gemstone/2.4.8
unix> export upgradeLogDir=tempDir
```

3. (Pilot conversion) Run an object and page audit

If this is the pilot conversion, to verify the integrity of your v6.x system, run an object audit and a page audit, prior to conversion.

For the page audit, you must set DBF_SCRATCH_DIR to a writable directory in your configuration file; both the -z and -l arguments are required in an rpm installation.

For more information on running object and page audits, see the *System Administration Guide.*

4. Ensure there are no dead objects

Prior to conversion, perform reclaim to ensure there are no dead objects in your repository.

5. (Optional) Set up classes for list instances

Some applications may need to perform application object conversions on objects in their repository, after the GemStone conversion has completed. For such applications, GemStone can be instructed to create lists of instances of particular classes during conversion.

This is distinct from collecting instances that refer to SmallFloats or LargeIntegers which can be detected for conversion during a later step.

Collecting instance of application classes is not enabled by default. To enable, you must define a variable in UserGlobals with a specific name and structure.

Collecting the lists of instances will slow down the conversion process; larger numbers of classes and instances of these classes will slow down the conversion more.

To collect lists of instance of classes, before invoking the **convprep6x** script, do the following:

a. Determine the list of classes for which instances are to be collected by the conversion.

b. Login to the v6.x system as SystemUser and ensure you are in transaction.

c. Create an IdentitySet containing one of more classes for which instances are to be located.

d. Store the IdentitySet in UserGlobals under the key `#ConversionClassesToFind`.

e. Commit the transaction.

During the execution of the **convprep6x** script, if the symbol `#ConversionClassesToFind` is found, information is written to the text file `$upgradeLogDir/ClassesForListInstances.txt`. Results are written to bitmap files which can be loaded into the converted GemStone/S 64 Bit v2.4.8 repository.

6. Run the GemStone/S 64 Bit 2.4.8 script convprep6x

The **convprep6x** script is in the `upgrade` directory of the 2.4.8 product tree; this should not be in your path, so it will need to be invoked with an explicit path. For example:

```
unix> $GEMSTONE_64/upgrade/convprep6x -s stoneName66
```

This script has the following options:

`convprep6x [-h] [-s` *stonename*`]`
`-h` prints this usage information.
`-s` *stoneName* sets the name of the running v6.x Stone to scan**.**

The **convprep6x** script scans the v6.x repository and produces files that are used in the second phase of conversion. These files are written to **$upgradeLogDir**. The **convprep6x** script will write progress messages to stdout. When it completes, it will report:

```
convprep6x[INFO]: ...conversion preparation complete.
```

If this output is not produced, check the log files in the **$upgradeLogDir** directory for information.

This script also shuts down the Stone. Do not restart or use these extents until the conversion is complete.

Prepare for Conversion

1. Configure GemStone/S 64 Bit 2.4.8

Create the system configuration file containing the appropriate settings for DBF_EXTENT_NAMES and STN_TRAN_LOG_DIRECTORIES, along with other configuration parameters required for your application.

The directories used for DBF_EXTENT_NAMES and STN_TRAN_LOG_DIRECTORIES must be separate from the directories used by the v6.6 installation.

While many of the settings will be the same in as in the v6.6 configuration file, you should be examine, in particular:

a. DBF_EXTENT_SIZES. The conversion process creates new v2.4.8 extents based on the v6.x extents; the v2.4.8 extents will be 20% to 200% larger. If this is the pilot, estimate the new size; if you are pre-growing extents allow for the larger growth.

   For the production conversion, set the new sizes based on growth information from the pilot conversion.

   Ensure that adequate space is available for the extents, as well as tranlogs and log files.

b. DBF_PRE_GROW. We recommend setting this to true for the conversion process, in particular for the production conversion. This ensures that you will not run out of disk space during the conversion process.

c. STN_TRAN_LOG_DIRECTORIES. For large repositories, we recommend setting the tranlogs to `/dev/null` for the conversion process. Tranlogs from the conversion process are not useful.

d. SHR_PAGE_CACHE_SIZE_KB. The shared page cache size should be increased by the same percentage as the growth in the extent files. If this is the pilot, start by estimating this to increase by 50%. It is important that the entire object table fit into the SPC. For the production conversion, use the increase computed from growth information from the pilot conversion.

   GEM_TEMPOBJ_CACHE_SIZE, GEM_PRIVATE_PAGE_CACHE_KB, and STN_PRIVATE_PAGE_CACHE_KB do not need to be increased.

2. Set the v2.4.8 Environment Variables

Set the `$GEMSTONE` and `$path` environment variables required for GemStone/S 64 Bit.

```
unix> export GEMSTONE=/usr/lib64/gemstone/2.4.8
unix> export PATH=$GEMSTONE/bin:$PATH
unix> export upgradeLogDir=tempDir
```

You will no longer need the `$GEMSTONE_64` environment variable.

The `$upgradeLogDir` environment variable that was defined in Step 2. on page 22 must be set to the same directory. Note that further conversion modified these files; if your conversion is upgraded partway, you may need to re-perform the initial preprocessing step in v6.6.

3. Copy a clean v 2.4.8 extent

Copy a clean v2.4.8 `extent0.dbf` to the location specified by the first entry in the of extent files, as it appears in the DBF_EXTENT_NAMES parameter setting in the configuration file that will be used by the 2.4.8 repository.

You must also set the permissions so it is writable by the user performing the upgrade.

New extent files for v2.4.8 will be created as part of conversion. **Do not** copy your version 6.x extent files. All 6.x extents should be available in their 6.x locations.

Perform the Conversion and Upgrade

1. Start the v 2.4.8 Stone

Start the 2.4.8 Stone on clean extents

```
unix> startstone -z confFile -l logfile stoneName248
```

2. Convert using conv6xTo2x script

Extent conversion is performed by the script **conv6xTo2x**. This script is in the v2.4.8 product tree, and (since the path has been updated) does not require a explicit path.

This script requires as an argument the configuration file used by the v**6.6** installation.

This script includes the `-L` switch that enables recording of all references to Large Integers, and the `-F` switch to record all references to Float or SmallFloat. The conversion process does not convert instances of LargePositiveIntegers or LargeNegativeIntegers that fall within the new SmallIntegers range, nor instances of Float or SmallFloat that are representable as instances of SmallDouble. This can be done as an optional manual step following conversion. Note that using the `-L` or `-F` options will slow down the conversion process, since each object must be checked for references.

This script has switches to specify the number of gems performing the conversion, and the number of OOPs per commit, which can be used to tune the conversion performance.

`conv6xTo2x -e` *oldSysConf* `[-h] [-s` *stoneName*`] [-n` *numConversionGems*`] [-o` *oopsPerCommit*`] [-F] [-L]`
`-e` *oldSysConf* specifies the configuration file to use. This configuration file must contain the list of 6.x extents to read data from. This is required; the script will not run without this information.
`-h` prints this usage information.
`-s` *stoneName* sets the name of the running stone to convert**.**
`-n` *numConversionGems* sets the number of parallel gems that will do the conversion; if this option is not used, the script will default to use one.
`-o` *oopsPerCommit* sets the number of oops per transaction; if this option is not used, the script will default to use 1000000.
`-F` During conversion, determine which objects reference one or more instances of Float or SmallFloat and write the object IDs to the binary bitmap file: `$upgradeLogDir/AllFloatRefs.bm.`
`-L` During conversion, determine which objects reference one or more instances of LargePositiveInteger or LargeNegativeInteger and write the object IDs to the binary bitmap file: `$upgradeLogDir/AllLrgIntRefs.bm`

For example,

```
unix> conv6xTo2x -e confFileForV66 -s stoneName248
```

The **conv6xTo2x** script will write progress messages to stdout. When it completes, it will report:

```
conv6xTo2x[INFO]: Successful conversion.
```

The stone will shut down at the completion of the `conv6xTo2x` step. This may take some time as the conversion gems terminate. Wait until the stone has completed and shut down before continuing.

3. Restart the 2.4.8 Stone

Once conversion is complete and the Stone is shut down, restart the Stone.

```
unix> startstone -e confFile -l logfile stoneName248
```

4. Upgrade using upgradeImageFrom6x script

The image upgrade is performed by the script `upgradeImageFrom6x`. This script includes a switch to set to size of the GEM_TEMPOBJ_CACHE_SIZE used for the upgrade process which may be needed to tune the upgrade.

`upgradeImageFrom6x [-h] [-c` *cacheSize*`] [-s` *stoneName*`]`
`-h` prints this usage information.
`-c` *cacheSize* sets the size of the GEM_TEMPOBJ_CACHE_SIZE; if this is not used, the script will default to use a value of 100000.
`-s` *stoneName* sets the name of the running stone to upgrade.

For example,

```
unix> upgradeImageFrom6x -s stoneName248
```

The script will prompt you to press the return key to begin.

The script invokes subordinate scripts to complete the upgrade. The upgrade process will take some time.

The script should complete with the message:

```
Upgrade completed. No errors detected.
```

If not, please preserve the Stone log file and the contents of `$upgradeLogDir`. Contact your internal GemStone support person or GemStone Technical Support.

5. Perform postprocessing using postconv script

To convert large objects found during the conversion process, you must run the `postconv` script. This script includes several arguments that can be used to tune performance.

`postconv [-c` *numCacheWamerGems*`][-h][-s` *stoneName*`] [-n` *numOfSessions*`] [-t` *tempObjCacheSize*`]`
`-c` *numCacheWamerGems* specifies the number of cache warmer gems to start before starting post conversion; if this is not specified, no cache warmer gems are started.
`-h` prints this usage information.
`-s` *stoneName* sets the name of the running stone to scan; if this option is not used, the script will default to **gs64stone.**
`-n` *numOfSessions* sets the number of parallel sessions to run to perform the large object conversion. If this option is not specified, the script will default to one.
`-t` *tempObjCacheSize* sets the size of GEM_TEMPOBJ_CACHE_SIZE in KB; by default, the same value as that used in the upgradeImage step, which defaults to 200000.

For example,

```
unix> postconv -s stoneName248
```

The `postconv` script will write progress messages to stdout. When it completes, it will report:

```
postconv[INFO]: Success! Conversion process from GemStone 6.3.x
or later to GemStone64 2.x completed.
```

6. (Pilot conversion) Run an object and page audit

If this is the pilot conversion, to verify the integrity of your converted system, run an object audit and a page audit to verify the integrity of the repository.

This is also recommended for the production upgrade, if time allows.

Post-upgrade Application Code Modifications

1. Perform any application specific post-processing for instances

If you set up `#ConversionClassesToFind` in your v6.x repository in order to collect instances of specified classes, the results will be written to bitmap files in the `$upgradeLogDir` directory.

There will be one file per Class, the file will have a name of the form:

<ClassName>`_`*<oldOop>*`_`*<newOop>*`.bm`

where:

<className> – Name of the class.
<oldOop> – GemStone/S v6.x OOP of the class.
<newOop> – GemStone/S 64 Bit v2.4.8 OOP of the class.

GemStone/S 64 Bit v2.4.8 contains Smalltalk methods to load bitmap files into customer-usable hidden sets. From these hidden sets, any required postconversion processing of the instances of these classes can be done.

Prior to the pilot conversion, you should determine if any application-specific post-processing is required on instances in your repository, and write and test any code required to perform the post-processing.

2. Perform any application specific post-processing for Large Integers or Floats

If you specified tracking of references to Large Integers or Floats, by using the `-L` or `-F` options during the `conv6xTo2x` script execution, the results will be written to a bitmap file:

```
$upgradeLogDir/AllLrgIntRefs.bm
$upgradeLogDir/AllFloatRefs.bm
```

GemStone/S 64 Bit v2.4.8 contains Smalltalk methods to load bitmap files into customer-usable hidden sets. From these hidden sets, you may load the individual objects and manually update the references.

This step is optional. Prior to the pilot conversion, if you determine that you should upgrade references to Large Integers and Floats, you should write and test any code required to perform the post-processing.

Chapter

4 Converting from v2.4.x to 3.3.x

This chapter describes how to convert a GemStone/S 64 Bit 2.4.x installation to GemStone/S 64 Bit version 3.3.9.

Between versions 2.x and 3.x, compiled methods have changed and version 3.x has new bytecodes to support native code. As a result, the conversion process requires that you recompile all methods in the application. This can be done either by filing in all application source code, or by iterating and recompile all methods in your application classes.

There are a number of changes you may have to make to application source code and to persistent instances as part of this conversion. GemStone kernel classes have extensive changes for version 3.0, as well as changes in v.3.1, 3.2, and 3.3. To review the changes in each version of GemStone, see the listing of Release Notes at GemStone/S 64 Bit Release History.

Upgrade Strategy

Upgrade from 2.x to 3.x is a substantial change, requiring changes to application code and, in some cases, application data objects.

We recommend that as part of validating your code changes, after making the required changes in your code, you file your application code and kernel class changes into an empty 3.3.9 installation, prior to performing the full upgrade. This will allow you to test your code changes in the 3.3.9 environment before undergoing the full application upgrade.

After verifying your application code changes, you should perform a pilot upgrade of your application, including the required modifications to application objects.

Conversion to v3.3x

Prior to Upgrade in existing application

1. Determine your application method recompile strategy

If you have source code for your application stored externally to the GemStone repository in a code management system, this can be filed in, in order to recompile all methods in your application.You should confirm that the format of your filed out code does not create new versions of your application classes on filein. See bug 42317 for details.

You may also write code to manually recompile methods in all classes; see "Recompile application code" on page 32 for details.

GemStone supports multiple versions of the same class, but tools operate on the most recent version of the classes. If you have instance of older versions of your applications classes that have not been migrated to the latest version, these class versions will not be upgraded by filein. We recommend that you migrate all instances to the most recent version of your application classes.

2. Update references to ScaledDecimal, if necessary

If your application uses the ScaledDecimal class, determine if you wish to continue to use this class under its new name FixedPoint, or if you want to use the new implementation of ScaledDecimal. If you wish to continue to use the old ScaledDecimal with its new name, you will need to modify your source code fileout so that all references to ScaledDecimal are changed to refer to FixedPoint, and all uses of the **s** ScaledDecimal literal notation are to the FixedPoint **p** literal notation.

Note that you will also have to manually update stored instances of ScaledDecimal, which is done following upgrade.

3. Save class comments for 3.x changes

If you have class comments in the Class description field, you will need to save the contents, and restore them after the upgrade. Due to structural changes in classes and changes in comment handling in earlier 3.x versions, the conversion process will remove any data stored here. Following upgrade, you should restore the class comment, which will then be upgraded for 3.x.

Prepare for Upgrade

Perform the following steps to prepare for the upgrade.

1. Shut down the 2.4.8 repository

Halt all user activity on the v2.4.8 repository, and shut down the Stone:

```
unix> stopstone stoneName248
```

where *stone*248 is the name of the version 2.4.8 stone. It is strongly recommended, but not required, that the repository be cleanly shut down before it is restarted under version 3.3.9. If the repository is not cleanly shut down, you must restart using the -N option.

2. Set the v3.3.9 Environment Variables

Set the `$GEMSTONE` and `$path` environment variables required for GemStone/S 64 Bit.

```
unix> export GEMSTONE=/usr/lib64/gemstone/3.3.9
unix> export PATH=$GEMSTONE/bin:$PATH
unix> export upgradeLogDir=tempDir
```

where *tempDir* is a temporary directory for which you have write permission.

Create the system configuration file containing the appropriate settings for DBF_EXTENT_NAMES and STN_TRAN_LOG_DIRECTORIES, along with other configuration parameters required for your application.

The directories used for DBF_EXTENT_NAMES and STN_TRAN_LOG_DIRECTORIES should be separate from the directories used by the v2.4.8 installation.

Ensure that adequate space is available for extents and transaction logs during the upgrade. You must provide space for the extents and transaction logs for both repositories, the old and the new.

3. Copy extent files

Copy your version 2.4.8 extent files into the location specified by the configuration file option `DBF_EXTENT_NAMES`. You may use the operating system **cp** command; copydbf is needed only when copying between platforms.

Perform the Upgrade

1. Convert using startstone

Conversion is done using the **-C** flag to startstone. Perform the conversion on the 2.4.8 extents you just copied:

```
unix> startstone -C -e confFile -l logfile stoneName339
```

2. Upgrade image

Ensure you are in a directory to which you have write permission, and run the upgrade script.

The upgrade is performed by the script `upgradeImage`. This script has optional switches to specify the stone name and to set to size of the GEM_TEMPOBJ_CACHE_SIZE used for the upgrade process.

`upgradeImage [-h] [-c` *tempObjCacheSize*`] [-s` *stoneName*`]`
`-h` prints this usage information.
`-c` *tempObjCacheSize* sets the size of the GEM_TEMPOBJ_CACHE_SIZE; if this is not used, the script will default to use a value of 100000.
`-s` *stoneName* sets the name of the running stone to upgrade; if this option is not used, the script will default to **gs64stone.**

For example,

```
unix> upgradeImage -s stoneName339
```

The script will prompt you to press the return key to begin.

The script invokes subordinate scripts to complete the upgrade. The upgrade process will take some time. You can examine the progress, if desired, by examining the file `$GEMSTONE/upgradeImage.out`.

The script should complete with the message:

```
Upgrade completed. No errors detected.
```

If not, please preserve the Stone log file and the contents of `$upgradeLogDir`. Contact your internal GemStone support person or GemStone Technical Support.

Post-upgrade Application Code Modifications

1. Recompile application code

The upgrade process requires all executable methods to be recompiled, to use the new bytecodes and classes introduced in version 3.0. This can be done via filein of your application code, or by iterating classes (including older versions of your classes, if you have instances of these older versions in your application), and recompiling methods. for each class.

a. Recompile by filein

Before filing in application code, you must prepare to handle any square-bracket Array constructors in your code. If your application code contains Array constructors using the syntax #[a, b, c], this is not valid syntax for version 3.x. In v3.x, you must use the syntax { a . b . c } for Array constructors.

To allow filein of application code that includes the square bracket Array constructors, in the gem that will perform the filein, prior to filein, execute:

```
System configurationAt: #GemConvertArrayBuilder put: true
```

This allows code including the old Array constructor syntax to be compiled; the resulting compiled method will be correct, and its source code will be updated to the correct syntax. This configuration variable is runtime only, so it is not necessary to unset it manually.

Now, file in all development and application code. Verify that errorcount is 0, and commit.

If you have instances of previous versions of classes, these old class versions will not be recompiled by this process. You should ensure that all application instances are migrated to current class versions before conversion, or manually recompile instances of older class versions.

b. Recompile within image

To recompile classes without filein, for each class in your repository execute

Class `recompileAllMethods`

You can collect the classes from your application symbol dictionaries using code similar to:

```
set := IdentitySet new.
set addAll: (ApplicationSymDict values select: [:ea |
      ea isKindOf: Class]).
set do: [:ea | ea recompileAllMethods]
```

The recompile method should be sent to the Class, not the Metaclass. You can verify by sending `#needsRecompileFor33` to an instance of a class or metaclass.

If you have instance of older versions of your classes, you will need to recompile methods on these older versions. For example:

```
class classHistory do: [:aClassVersion |
      aClassVersion recompileAllMethods]
```

Attempting to execute methods that have not been recompiled will result in errors.

2. Convert persistent blocks

ExecutableBlocks have been reimplemented in version 3.x; persistent instances of ExecutableBlocks are not usable in version 3.x. Executable blocks in source code are recompiled during application filein.

c. Application-specific stored blocks

If your application stores persistent blocks, you will have to locate and recompile all such blocks before they can be executed.

d. SortBlocks in SortedCollections

To convert the sortBlocks in persistent SortedCollections in your application, you may run the `postconv` script. This script only converts simple blocks; if your SortedCollection blocks are not simple (such as referring to method context), they cannot be automatically recompiled.

`postconv [-c` *numCacheWamerGems*`][-h][-s` *stoneName*`] [-r] [-n` *numSessions*`] [-t` *tempObjCacheSize*`] [-u` *userId*`]`

`-c` *numCacheWamerGems* specifies the number of cache warmer threads in a single gem to load the object table into the shared cache before starting post-conversion. If not specified, no cache warming is done.
`-h` prints this usage information.
`-s` *stoneName* sets the name of the running stone to scan.
`-n` *numSessions* specifies the number of parallel sessions which will convert the instances of SortedCollection and its subclasses. By default, use one session.
`-r` specifies to reuse an existing version of `$upgradeLogDir/AllSortedCollections.bms`, if it exists. This file contains the OOPs of all instances of SortedCollections and its subclasses. By default, the existing file is deleted and a new one created.
`-t` *tempObjCacheSize* sets the size of GEM_TEMPOBJ_CACHE_SIZE in KB; by default, 20000.

`-u` *userId* is the UserId whose SymbolList includes all subclasses of SortedCollection, for which instance's sortBlocks will be converted. If not specified, defaults to SystemUser

For example,

`unix>` **`postconv -s`** *stoneName339*

The `postconv` script will write progress messages to stdout. When it completes, it will report:

`postconv[INFO]: Congratulations!` *NNN* `SortedCollections were successfully converted. No errors were detected.`

If postconv reports errors, the results include information on the specific sortBlocks that failed recompile. These will need manual repair. Contact GemTalk Technical Support for assistance.

3. Restore and upgrade Class comments

In v3.1 and later, the class comment is stored in the class as a String, and accessed via #comment and #comment: methods. This differs from the comment handling in earlier versions.

After upgrade, you should not have class methods named #comment on any of your classes. The `upgradeComment` script both sets the comment, from either the description field or the comment method results, and deletes the class method #comment.

First, restore Class descriptions that were saved from your version 2.4.8 repository (as per Step 3. on page 30). Descriptions are restored using the `description:` method.

Then, convert all comments using the `upgradeComments` script.This converts both `description:` field comments and class #comment methods into the correct form.

`upgradeComments [-c <`*tempObjCacheSize*`][-s <`*stoneName*`>]`
`-c <`*tempObjCacheSize*`>` sets the size of temp obj cache in KB.
`-s <`*stoneName*`>` sets the name of the running stone to upgrade comments; if this option is not used, the script will default to **gs64stone.**

For example,

`unix>` **`upgradeComments`** `-s` *stoneName339*

The script should complete with the message:

`Upgrade completed. No errors detected.`

If you have a very large number of class comments, it is possible to run out of temporary object memory while upgrading comments. If so, use the `-c` argument to specify a larger temporary memory space.

If this script reports errors, you may have non-standard comment forms. Preserve the error logs and contact GemStone Technical Support.

4. Convert instances of reimplemented server classes

Optionally, you may wish to convert instances of reimplemented classes. Instances of Float, SmallFloat, LargePositiveInteger, LargeNegativeInteger, DoubleByteString, DoubleByteSymbol or QuadByteString in your version 2.4.8 repository are not converted during conversion to 3.x. The instances are auto-migrated to the new class each time they are faulted into the VM.

To avoid this overhead, you can perform a `listInstances` of the classes ObsLargePositiveInteger, ObsLargeNegativeInteger, ObsDoubleByteString, ObsDoubleByteSymbol, ObsQuadByteString, ObsFloat, and ObsSmallFloat. Send the message #convert to each instance, and commit.

5. Convert FixedPoint instances

If for Step 2. on page 30, you decided to continue to use the old ScaledDecimal class, now named FixedPoint, do not perform this step.

If your 2.4.8 application included instances of ScaledDecimal, following conversion to 3.x they are now instances of FixedPoint. If you intend to use the new ScaledDecimal implementation rather than FixedPoint, you must perform a manual step to convert each instance to a new ScaledDecimal. To do this, find all instances of FixedPoint and execute code similar to the following:

```
newScaledDecimal := aFixedPoint asScaledDecimal.
newScaledDecimal become: aFixedPoint.
```

Verify that errorcount is 0, and commit.

Chapter

5 Upgrade from 3.3.x to 3.6.1

This chapter describes how to convert a GemStone/S 64 Bit v3.3.9 repository to v3.6.1.

Upgrading from v3.3.9 to v3.6.1 is not a conversion; only `upgradeImage` is required. However, there may be changes required in application source code; GemStone kernel classes have many changes for v3.4, 3.5, and 3.6. To review the changes in each version of GemStone, see the listing of Release Notes at GemStone/S 64 Bit Release History.

Following the final upgrade to v3.6.1, you will file in any code, such as GemStone add-on products and application-specific changes to server class methods, and ensure that your application is configured correctly for general use.

Upgrade to GS/64 v3.6.1

Prepare for Upgrade

1. Shut down the 3.3.9 repository

Halt all user activity on the v3.3.9 repository, and shut down the Stone:

```
unix> stopstone stone339
```

where *stone*339 is the name of the version 3.3.9 stone.

2. Set the v3.6.1 Environment Variables

Set the `$GEMSTONE` and `$path` environment variables required for GemStone/S 64 Bit.

```
unix> export GEMSTONE=/usr/lib64/gemstone/3.6.1
unix> export PATH=$GEMSTONE/bin:$PATH
unix> export upgradeLogDir=tempDir
```

where *tempDir* is a temporary directory for which you have write permission.

Create the system configuration file containing the appropriate settings for DBF_EXTENT_NAMES and STN_TRAN_LOG_DIRECTORIES, along with other configuration parameters required for your application.

The directories used for DBF_EXTENT_NAMES and STN_TRAN_LOG_DIRECTORIES should be separate from the directories used by the v3.3.9 installation.

Ensure that adequate space is available for extents and transaction logs during the upgrade. You must provide space for the extents and transaction logs for both repositories, the old and the new.

3. Copy extent files

Copy your version 3.3.9 extent files into the location specified by the configuration file option `DBF_EXTENT_NAMES`. You may use the operating system **cp** command; copydbf is needed only when copying between platforms.

Perform the Upgrade

1. Start the Stone

Start the 3.6.1 Stone on the 3.3.9 extents you just copied:

```
unix> startstone -e confFile -l logfile stoneName361
```

2. Upgrade image

Ensure you are in a directory to which you have write permission, and run the upgrade script.

The upgrade is performed by the script `upgradeImage`. This script has optional switches to specify the stone name and to set to size of the GEM_TEMPOBJ_CACHE_SIZE used for the upgrade process.

`upgradeImage [-h] [-c` *cacheSize*`] [-s` *stoneName*`]`
`-h` prints this usage information.
`-c` *cacheSize* sets the size of the GEM_TEMPOBJ_CACHE_SIZE; if this is not used, the script will default to use a value of 100000.
`-s` *stoneName* sets the name of the running stone to upgrade; if this option is not used, the script will default to **gs64stone.**

For example,

```
unix> upgradeImage -s stoneName361
```

The script will prompt you to press the return key to begin.

The script invokes subordinate scripts to complete the upgrade. The upgrade process will take some time. You can examine the progress, if desired, by examining the file `$upgradeLogDir/upgradeImage*.out`.

The script should complete with the message:

```
Upgrade completed. No errors detected.
```

If not, please preserve the Stone log file and the contents of `$upgradeLogDir`. Contact your internal GemStone support person or GemStone Technical Support.

3. Restore System Account passwords

Log in to GemStone/S 64 Bit version 3.6.1 as DataCurator or SystemUser, and change the password for SystemUser, DataCurator, and GcUser to a secure password, such as the passwords used for these accounts in v6.x. For example:

```
topaz 1> run
(AllUsers userWithId: 'SystemUser') password: '6xPassword'.
(AllUsers userWithId: 'GcUser') password: '6xPassword'.
(AllUsers userWithId: 'DataCurator') password: '6xPassword'.
System commitTransaction
%
```

where *6xPassword* is the account password used prior to conversion.

Post Upgrade Application Code Modifications

After completing the conversions earlier in this Guide, and the final upgrade to version 3.6.1, you are ready to make the final changes for your application to be ready for users.

1. Reinstall any other GemStone products that modify kernel classes.

If you use GemConnect or GemBuilder for Java, you should reinstall the appropriate version of these products into your repository.

To install, use the procedure in the *Installation Guide* for that product.

2. File in Kernel class changes

During the pilot conversion, you have determined whether there are kernel class methods that you have modified or added in the GemStone/S v6.x repository, that are necessary and appropriate for GemStone/S 64 Bit v3.6.1, or if changes to these methods are required to make them compatible.

If the kernel class changes are applicable and compatible, file in the changes, verify that errorcount is 0, and commit.

While you may have reinstalled these methods in an earlier chapter of this Conversion Guide, they must be re-installed following the upgrade to 3.6.1.

3. Rebuild indexes, if necessary

The conversion step from 2.4.x to 3.3.x requires rebuild of indexes that refer to instances of kernel classes which have new implementations in version 3.x. These indexes **must** be rebuilt to update internal information in the Btree indexing structure, otherwise query results may be incorrect.

See “Remove indexes on instances of reimplemented server classes” on page 12 for details on this requirement.

GemStone indexing has a number of changes and refactoring in v3.3 and 3.4, and index creation methods that do not include the `withLastElementClass:` keyword have been removed. Refer to the *System Administration Guide* for v3.6 for details, and the *Release Notes* for v3.4 and earlier versions.

4. Upgrade user account passwords

Internal password formats changed in version 3.1. The new format uses updated, more secure encryption. As each user logs in after upgrade, their password will be transparently upgraded to the new format, with no extra upgrade actions required.

However, accounts that do not log in will continue to have the same passwords, which are stored in the older, less secure encryption.

For better security, you should ensure that all user accounts have logged in, or had their passwords explicitly updated by an administrator. To make these tasks easier, the following methods can be used:

```
UserProfileSet >> usersWithOldPasswordEncryption
UserProfileSet >> disableUsersWithOldPasswordEncryption
```

See image comments for these methods for more details.

5. Run an object and page audit

To verify the integrity of your upgraded system, run an object audit and a page audit.

6. Make backup

Create a full backup of the upgraded repository.

Complete configuration

1. Verify file permissions

Ensure your NetLDI mode and Extent file permissions are set correctly per the discussion on “File permissions to allow multi user access” on page 9.

2. Startup the NetLDI for v3.6.1

Start a NetLDI process, using the NetLDI service name or port, per the discussion on “Define the NetLDI Service name, if necessary” on page 19.

Ensure that users other than the administrative UNIX user can login, to verify the file and executable permissions.

3. Recompile User Actions

You will need to recompile and relink User Action libraries and GCI applications. See the *GemBuilder for C* manual for v3.6 for details.

4. Configure GBS for v3.6.1

If you are using GBS clients, ensure you are running a supported version of GBS and client Smalltalk. You must use GBS version 8.5 or later for VW, or GBS 5.4.6 or later for VA, to connect to a GemStone/S 64 Bit v3.6.1 repository.

There are changes in the library loading and naming instructions. For the requirements, see the GemStone/S 64 Bit Windows Client Installation Guide for v3.6 and the GemBuilder for Smalltalk/VA Installation Guide for v5.4.6.