GemStone®

GemStone/S 64 Bit Installation Guide

For Linux on x86_64 Compatible Systems

Version 2.2.3

November 2007

GemStone S 64™

INTELLECTUAL PROPERTY OWNERSHIP

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

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

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

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

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

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

COPYRIGHTS

PATENTS

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

TRADEMARKS

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

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

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

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

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

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

Linux is a registered trademark of Linus Torvalds and others.

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

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

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

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

Preface

About This Documentation

This document explains how to install GemStone/S 64 Bit version 2.2.3, and how to convert from previous GemStone/S 64 Bit versions or from earlier GemStone/S server products.

For information regarding new and modified features, please refer to the *GemStone/S 64 Bit Release Notes.*

Note that this Installation Guide has been published subsequent to the actual product release.

Terminology Conventions

This document uses the following terminology:

- The term "GemStone" is used to refer both to the product, GemStone/S 64 Bit, or previous GemStone/S server products; and to the company, GemStone Systems, Inc.

Typographical Conventions

This document uses the following typographical conventions:

- Operating system and Topaz commands are shown in **`bold`** typeface. For example:

  `copydbf`

- Smalltalk methods, GemStone environment variables, operating system file names and paths, listings, and prompts are shown in `monospace` typeface. For example:

  `markForCollection`

- Place holders that are meant to be replaced with real values are shown in *italic* typeface. For example:

  StoneName`.conf`

In formal syntax listings, these additional conventions are used:

- Literals are shown in **bold** typeface. For example:

  tcp

- Optional arguments and terms are enclosed in square brackets. For example:

  [*dbfName*]

- Braces { } mean 0 or more modifiers. For example:

  {*modifier*}

  In this example you may list as many modifiers as you wish, but they are not required.

- Alternative arguments and terms are separated by a vertical bar (pipe). For example:

  gemStoneName | *netLdiName*

  In this example you must specify one name, but not both.

Technical Support

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

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

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

The following types of information are provided at this web site:

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

Documentation for GemStone/S 64 Bit is provided in PDF format. This is the same documentation that is included with your GemStone/S 64 Bit product.

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

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

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

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

Community provides customer forums for discussion of GemStone product issues.

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

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

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

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

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

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

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

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

Email: support@gemstone.com

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

Requests for technical assistance may also be submitted by email or by telephone. We recommend you use telephone contact only for more serious requests that require immediate evaluation, such as a production system that is non-operational. In these cases, please also submit your request via the web or email, including pertinent details such error messages and relevant log files.

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

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

24x7 Emergency Technical Support

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

Training and Consulting

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

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

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

Contents

Chapter 1. Installing GemStone/S 64 Bit Version 2.2.3

Chapter 2. Upgrading from previous GemStone/S 64 Bit 2.x versions

Chapter 3. Converting from GemStone/S 64 Bit 1.x

Chapter 4. Converting from GemStone/S 6.1.x

Chapter 5. Converting from GemStone/S 6.2

Chapter 6. Configuring GBS for GemStone/S 64 Bit

Chapter 1

Installing GemStone/S 64 Bit Version 2.2.3

This chapter describes the procedure for installing GemStone/S 64 Bit version 2.2.3 on a single machine. If you have enough disk space on a single machine, we recommend that you set up GemStone this way initially to ensure that all the pieces work together. At the end of this chapter, we suggest refinements you might want to make, such as running GemStone in a network configuration.

NOTE

If you are upgrading to this release from a previous version of GemStone/S 64 Bit, or converting from a previous GemStone/S product, follow the instructions in the appropriate chapter of this Installation Guide.

Adjust the installation to meet your specific needs. The topic "What Next?" on page 1-14 provides references to procedures and related information in the *System Administration Guide for GemStone/S 64 Bit*.

Review the Installation Procedure

The following list summarizes the steps to install GemStone/S 64 Bit.

Check the System Requirements

Before you install GemStone/S 64 Bit, ensure that the following system requirements are satisfied. Systems meeting these requirements are suitable for installing GemStone/S 64 Bit and beginning development, but additional system resources may be necessary to support large applications.

Platform

- System with an x86_64-compatible processor.

RAM

- At least 1 GB (2 GB recommended)
- 10 MB for each Gem session process beyond the first two, depending on how the Gem processes are configured

Swap space

- At least 1 GB of swap space beyond other system needs (2 GB recommended). In general, your total swap space should be at least twice the RAM installed.

Disk space

- Space for the installed distribution files—you need approximately 65 MB for GemStone/S 64 Bit, and additional space for other products.
- Additional disk space as required for your repository.
- 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.

Media drive

- CD-ROM drive

Operating system

- SuSE Linux ES 10 (kernel version 2.6.16.27-0.9-smp and glibc 2.4-31.5).
- For maximum performance, install the optimized POSIX Asynchronous I/O (AIO) library. GemStone/S 64 Bit will run correctly without this library, with performance comparable to previous releases; however transaction log I/O operations, particularly on raw partitions, will be much faster if this library is used. See the section “Optimized POSIX Asynchronous I/O” on page 1-4 for details.

- 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.

  a. The maximum shared memory segment should be set to a value larger than your desired Shared Page Cache size, and not more than 75% of your real memory size.

     For example, if you have 512 MB of real memory:

     512 MB * .75 = 384 MB
     384 MB * 2**20 = 402653184 bytes

     To set shared memory, you would append the following text to the `/etc/sysctl.conf` file. The setting of the shared memory size is read from this file during the boot process.

     ```
     #
     # Shared Memory setting for GemStone
     kernel.shmall = 402653184
     kernel.shmmax = 402653184
     ```

     For more details, consult your Linux operating system documentation.

  b. You may need to increase the settings for semaphores. `semmsl` 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. (Note that `semmsl` ends with a lowercase L, not a digit.)

     On the Stone's node (and any other node where a repository file is located), this parameter must provide 1 semaphore for each user who will log in to that Stone from any node plus an overhead of 4. In distributed systems, nodes that have only user sessions must provide 1 semaphore for each user session on that node plus an overhead of 1.

     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 + 1).

  c. `semmns` sets the total number of semaphores in the system. This parameter limits the total number of GemStone sessions on a node. Keep this setting `>= semmsl` (above).

  d. A file descriptor limit of 1024 is adequate for up to about 500 GemStone users. Each user session requires two file descriptors, and others are needed for extents, transaction logs, and overhead. Consequently, you may need to set the `nfile` parameter to a larger value, such as 2048. *Use caution and increase the default setting only when necessary because doing so can have system side effects.*

- The configuration option SHR_PAGE_CACHE_SIZE_KB defines the size (in KBytes) of extent page space in the shared page cache. The maximum acceptable value for this configuration option is limited by system memory, kernel configurations, cache space allocated by SHR_PAGE_CACHE_NUM_PROCS and space allocated for other GemStone caches.

  For more general information about these and other configuration options, see Appendix A of the *System Administration Guide for GemStone/S 64 Bit*.

Optimized POSIX Asynchronous I/O

GemStone/S 64 Bit can use optimized POSIX Asynchronous I/O (AIO) functions, by installing the optimized POSIX AIO library, libposix-aio.so, into the appropriate place in your Linux installation. If this library is available, GemStone/S 64 Bit will automatically use it; if it is not available, the regular librt.so library is used instead.

Use of libposix-aio.so optimizes performance for transaction log I/O operations, particularly on raw partitions. Maximum commit performance with the regular library is not worse than in previous releases, but much slower than when using the optimized library.

To install the optimized POSIX AIO library:

a. Download libposix-aio-0.8.1.tar.gz from http://sourceforge.net/projects/paiol.

   You must use version 0.8.1 with GemStone/S 64 Bit v2.2.3; later versions may not work correctly.

b. As root, do the following:

```
cd <directory containing libposix-aio-0.8.1.tar.gz>
tar xzf libposix-aio-0.8.1.tar.gz
cd libposix-aio-0.8.1
./configure
make
cp -a src/.libs/libposix-aio.so* /usr/lib64
```

c. Verify the process completed correctly using:

```
ls -al /usr/lib64 | grep posix-aio
```

   This should result in output similar to:

```
lrwxrwxrwx  1 root root     21 Apr 13 12:10 libposix-aio.so -> libposix-aio.so.0.0.0
lrwxrwxrwx  1 root root     21 Apr 13 12:10 libposix-aio.so.0 -> libposix-aio.so.0.0.0
-rwxr-xr-x  1 root root  52942 Apr 13 12:10 libposix-aio.so.0.0.0
```

No further configuration is necessary; GemStone/S 64 Bit will automatically use libposix-aio.so if it is available.

If libposix-aio.so is not used, either if it is not available, or if the startstone -s switch is used, a message is returned to stdout on stone startup.

To verify that libposix-aio.so is being used in a running system, in the output of tools such as 'ps' or 'top', the stone process will be 'stoned_paio' if libposix-aio.so is being used, and 'stoned' if it is not being used.

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.

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.

C/C++ Compiler

- gcc / g++ 4.1.0

  linker version 2.16.91.0.5 20051219

  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. This compiler is required only for development work, not for execution.

Debugger

A C debugger can be useful to allow problem analysis by GemStone consulting or Technical Support. It also may allow you to debug your C user actions. It is not required for GemStone execution.

- gbd 6.5

Prepare for Installation

Perform the following steps to prepare the machine to receive the GemStone / S 64 Bit software. Although most steps require root login, we recommend that you perform the initial step as the GemStone administrator.

Table 1 shows the portions of the system that are affected by the installation of GemStone.

Table 1 Parts of System Affected by GemStone Installation

Location	Use
`/dev/rdsk`	Optional raw partitions for repository extents and transaction logs
`/etc/services`	Internet services database
/InstallDir/`GemStone64Bit2.2.3-x86_64.Linux`	Location of the object server software
`/opt/gemstone`	Default location for server lock files and log files for GemStone network servers (NetLDIs)
`/usr/gemstone`	Alternative location for lock and log files, for compatibility with previous releases; `/opt/gemstone` is created unless `/usr/gemstone` already exists

1. As the GemStone administrator, log in to a machine that has adequate resources to run GemStone and that owns the disk on which you are going to install the GemStone files.

NOTE

Do not copy the files as root. The ownerships that were in effect when the

distribution media was created are preserved, and this might result in file permission errors for users at your site.

1. Determine that adequate swap space is available:

   % **cat /proc/swaps**

2. Check the free disk space and determine the disk drive and partition on which you will install the GemStone software.

   To list all disk partitions, along with the amount of free space in each partition:

   % **df**

   We recommend that you avoid choosing either an NFS-mounted partition or one containing UNIX swap space for the initial installation. Mounted partitions can result in executables running on the wrong machine and in file permission problems. Existence of swap space on the same drive can dramatically slow GemStone disk accesses.

3. Select an installation directory, *InstallDir*, and make this directory the current working directory.

4. GemStone / S 64 Bit is provided as a zipped archive file with a name similar to `GemStone64Bit2.2.3-x86_64.Linux.zip`. Move this distribution file to the directory location in which GemStone will be installed, *InstallDir*.

5. Unzip the distribution file using unzip. For example:

   % **unzip GemStone64Bit2.2.3-x86_64.Linux.zip**

   The *InstallDir* now contains a GemStone directory with a name similar to `GemStone64Bit2.2.3-x86_64.Linux`.

   In addition to several subdirectories, this directory also contains two text files: `PACKING`, which lists all of the GemStone files, and `version.txt`, which identifies this particular product and release of GemStone.

6. Log in as root.

   NOTE

   Although you can complete the installation as a non-root user, we do not recommend this. During installation, GemStone system security is established through file permissions and process attributes. To ensure that the installation is successful, ***you must install as root****. If you later decide to change the security of your GemStone system, see Chapter 1 of the* System Administration Guide for GemStone / S 64 Bit, *which explains the concept of GemStone server file permissions and how to change them.*

Set the Environment

Perform the following steps to properly configure the operating environment.

1. Set the environment variable GEMSTONE.

   a. If more than one installation of any GemStone/S product resides on this machine, check for existing GemStone environment variables:

      ```
      % env | grep GEM
      ```

      All GemStone environment variables are displayed.

   b. If any environment variables exist, you must specifically unset each one.

      C shell:

      ```
      % unsetenv GEMSTONE GEMSTONE_SYS_CONF \
        GEMSTONE_EXE_CONF GEMSTONE_LOG GEMSTONE_LANG
      ```

      Bourne or Korn shell:

      ```
      $ unset GEMSTONE GEMSTONE_SYS_CONF GEMSTONE_EXE_CONF \
        GEMSTONE_LOG GEMSTONE_LANG
      ```

   c. Set the environment variable GEMSTONE to the *full pathname* (starting with a slash) of your new GemStone installation directory.

      C shell:

      % **setenv GEMSTONE** *InstallDir*/**GemStone64Bit2.2.3-x86_64.Linux**

      Bourne or Korn shell:

      ```
      $ GEMSTONE=InstallDir/GemStone64Bit2.2.3-x86_64.Linux
      $ export GEMSTONE
      ```

Create the GemStone Key File

To run GemStone, you must create a key file. Instructions and information to create this file were shipped with the distribution media. If either of these materials is missing, call GemStone Contract Administration.

1. Change the permissions on the directory `$GEMSTONE/sys` so that you can create the file:

   ```
   % cd $GEMSTONE/sys
   % chmod 755 .
   ```

7. Using a text editor and the information provided, create the key file `$GEMSTONE/sys/gemstone.key`.

8. Change the file and directory permissions so that they are no longer writable:

   ```
   % chmod 555 gemstone.key
   % chmod 555 .
   ```

Verify TCP/IP

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

1. Verify that TCP/IP networking software is functioning (**1** is the number 1):

   % **/bin/ping** *hostname*

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

Define the NetLDI Service

The `gs64ldi` service must be defined in your TCP/IP network database.

NOTE

You might need the help of your UNIX system administrator to complete this procedure.

1. Determine whether the `gs64ldi` service is already defined:

   % **ypcat services | grep gs64ldi**

   If it is defined, skip the rest of this procedure and continue with the installation at "Run the Installation Script" on page 1-9.

   If it is not defined, continue performing this procedure.

9. Determine which TCP/IP network database (local or NIS) is in use:

   % **ypwhich**

   If the program is missing or you see an error message when you run it, you can assume that your machine is using a local copy of the TCP/IP network database instead of a copy provided by NIS. Perform the following step on your local copy of the network database (the file `/etc/services`).

   If NIS is running, have your UNIX system administrator perform the following steps.

10. Add an entry similar to the following to the network database:

    ```
    gs64ldi 50377/tcp #GemStone/S 64 Bit 2.2.3
    ```

    Choose a port number that is not being used by another service. The port number should be in the range 49152 <= port <= 65535, to confirm to IANA standards (http://www.iana.org/assignments/port-numbers).

    NOTE

    If you are upgrading from a previous version, you might need to keep the NetLDI for that version running. Assign a different port number to `gs64ldi`.

11. If NIS is running, propagate the change to the network database to the rest of the network.

12. If NIS is not running, but several machines will be running GemStone, have the UNIX system administrator update the network database for each machine. Note that the port number must be the same for every machine.

Run the Installation Script

Invoke the installation script from the `install` subdirectory:

```
% cd $GEMSTONE/install
% ./installgs
```

`installgs` is an interactive script that analyzes your system configuration and makes suggestions to guide you through installing GemStone on your machine.

NOTE

You can usually terminate execution of the installation script with Ctrl-C without risk to your files. When it is not safe to do so, the message `Please do not interrupt` *appears on the screen. If this happens, wait for the message* `now it is OK to interrupt` *before you interrupt the script. You can run the script again from the beginning as many times as necessary.*

Decisions to Make During Installation

During installation, you are asked several questions. The entire installation dialog is not reproduced here, but the main points are addressed. Some questions may not be asked, depending on answers to previous questions.

Whenever you are asked to answer "yes" or "no," answer with **y** or **n**. When the script offers a default answer in square brackets (such as "`[y]`"), press Enter to accept the default.

Verify the Release Tree?

Do you want the installation script to verify your GemStone release tree?

This process takes a few minutes, but it's a good idea to ensure that the files were transferred from your distribution media without error.

Default: Verify the installation tree.

The GemStone installer will attempt to verify some system requirements on your system. Some of the verifications may issue warnings of the form:

```
./installgs[52]: test: Specify a parameter with this command.
./installgs[52]: -q:  not found.
./installgs[260]: test: Specify a parameter with this command.
```

Unless an explicit installation failure is returned, the installation will complete successfully, and the above warnings may be ignored.

Do you want the installation script to set up directories for server lock files and NetLDI logs?

The default location for server lock files and NetLDI log files is `/opt/gemstone`, although for compatibility with earlier products `/usr/gemstone` is used only if it exists. If neither directory exists, the installation script offers to create `/opt/gemstone` and the subdirectories `locks` and `log`. Then, the script offers to grant world access (777) to these directories.

If you answer **no** to creating the directories, you must create them (or provide a symbolic link) before starting the server.

Do you want the installation script to set the owner and group for all the files in the GemStone distribution?

If you answer **yes**, the script will prompt you for the owner and group you want to use. Refer to Chapter 1 of the *System Administration Guide for GemStone/S 64 Bit* for more information about setting owner and group permissions.

If you answer **no**, the permissions will remain the same as when the files were extracted from the distribution media.

Do you want the installation script to protect the repository file?

The default, which we recommend, gives only the owner read and write access (600) through ordinary UNIX commands. Other users can read and write the repository through a GemStone session. If you choose not to protect the repository, the setuid bit is cleared from all executables, which causes them to run under ownership of the user who invokes them.

Default: Set the repository permission to 600, and leave the setuid bit applied.

Allow NetLDI to Run as Root?

Do you want the installation script to allow non-root users to start a NetLDI that runs as root?

The NetLDI is a network server that permits remote processes to interact with the repository. There are two ways to set up a NetLDI so that it can provide services to all GemStone users: it can run as root, or it can run in guest mode with a captive account.

- To run NetLDIs as root, accept the default “yes” response. Ownership of the NetLDI executable is changed to root, and the setuid bit is set. Any GemStone user will be able to start a NetLDI process that is accessible to all GemStone users because it will always run as root. For certain services, users will need to authenticate themselves by supplying a password. Alternatively, answer “no” but log in as root before starting the NetLDI.

  If the NetLDI uses a port number less than 1024, it must run as root.

- To run NetLDIs in guest mode with a captive account, answer “no” to the prompt, because those modes are not permitted if the NetLDI runs as root. “Guest mode” means that GemStone users do not have to supply a UNIX password to use NetLDI services. The “captive account” is an account that owns all processes the NetLDI starts; typically, it is the GemStone administrative account that owns the files. You must start the NetLDI while logged in as that account.

Default: Change ownership of the `netldi` executable to root, and set its setuid bit.

Set up an Extent?

Do you want the installation script to set up an extent now?

GemStone is distributed with a read-only copy of the initial repository in `$GEMSTONE/bin/extent0.dbf`. Before you can start GemStone, this file must be

copied to a suitable location and made writable. The script offers to copy the file to its default location of `$GEMSTONE/data`.

If you are a new GemStone user, we recommend that you answer **y**. If you are an existing GemStone user, you might prefer to answer **n**, then copy the extent to a different location yourself. (If you choose a location other than the default, you must edit your configuration file before starting GemStone. For information, see the *System Administration Guide for GemStone/S 64 Bit*.)

Default: Place a writable copy of `extent0.dbf` in `$GEMSTONE/data`.

Start a NetLDI?

Do you want the installation script to start a NetLDI?

If you prefer, you can start these processes manually at any time.

Almost every host needs a NetLDI. You must start a NetLDI when the Stone repository monitor or Gem session processes will run on this machine.

You can start a NetLDI that runs as root by answering **yes** to this prompt and the confirmation that follows. However, if you want to start the NetLDI in guest mode with a captive account, you must do that after completing the installation. For more information about guest mode with captive account, see Chapter 3 of the *System Administration Guide for GemStone/S 64 Bit*.

Default: Do not start a NetLDI at this time.

Start an Object Server?

As root, you cannot start an object server, but the script offers to start one as another user. You will start the server later in the installation, so answer **no**.

Default: Do not start an object server at this time.

Log out as root.

Change System Passwords and Add Users

After installing GemStone/S 64 Bit, you must change the passwords for the three administrative users: DataCurator, SystemUser, and GcUser. (The initial password for each is `swordfish`.) The DataCurator account is used to perform system administration tasks. The SystemUser account ordinarily is used only for performing GemStone system upgrades. The GcUser account is used by the garbage collection task, which runs automatically as a separate login. Access to each of these accounts should be restricted.

Chapter 6 of the *System Administration Guide for GemStone/S 64 Bit* tells you how to change the passwords and set up accounts for other GemStone users.

You must then establish GemStone accounts for each of your system's users.

The script `makeusers` in the `$GEMSTONE/install` directory helps you change the system passwords and add users. Note that the `makeusers` script assumes that the system passwords are as yet unmodified from their initial values of `swordfish`. Because `makeusers` modifies the system passwords, you can run the script only once on a given repository. To change system passwords or add more users at a later time, refer to the procedures in the *System Administration Guide for GemStone/S 64 Bit*.

NOTE

The actions described in this procedure affect only the repository whose name you enter while running the script makeusers.

1. Ensure that the GemStone environment variables are set for version 2.2.3:

   ```
   % env | grep GEM
   ```

   If necessary, set the environment variables as detailed in "Set the Environment" on page 1-7.

13. Invoke the script `gemsetup`.

    C shell:

    ```
    % source $GEMSTONE/bin/gemsetup.csh
    ```

    Bourne or Korn shell:

    ```
    $ . $GEMSTONE/bin/gemsetup.sh
    ```

    The script defines the GemStone environment for users by modifying the `PATH` and `MANPATH` variables to include `$GEMSTONE/bin` and `$GEMSTONE/doc`, respectively.

14. Start the repository monitor (Stone). You must do this from a working directory for which you have write privilege. For instance:

    ```
    % cd $HOME
    % startstone
    ```

15. Invoke the script `makeusers`:

    ```
    % cd $GEMSTONE/install
    % ./makeusers
    ```

    The script prompts you for the name of your Stone (the default is `gs64stone`), new administrative passwords, and names of the users you want to create. The password for new users will be set to `gemstone`.

    The script writes the information you supply to a file that can be read by Topaz, an interactive GemStone interface that you can use for system administration. Topaz deletes the file when it finishes with it. By default, this file is `newusers.topaz`, created in your `$HOME` directory. If you specify another file, be sure to give a full pathname. (You cannot create a file in the `install` directory unless you made the directory writable during installation.)

16. When the script `makeusers` has finished, run the linked version of Topaz and read in the file that the script created.

    a. Invoke linked Topaz:

       ```
       % topaz -l
       ```

    b. Read in the file containing the new users:

       ```
       topaz> input $HOME/newusers.topaz
       ```

       Topaz reads in the file, displays GemStone's new settings, deletes `$HOME/newusers.topaz`, and logs you out of GemStone.

c. Exit Topaz:

```
topaz> exit
```

Have Users Execute gemsetup

The directory $GEMSTONE/bin contains two files, gemsetup.sh and gemsetup.csh, to help set a user's environment. These files define the GemStone environment for users by modifying the PATH and MANPATH variables to include $GEMSTONE/bin and $GEMSTONE/doc, respectively.

After GemStone/S 64 Bit 2.2.3 has been installed, you should notify each GemStone user of the installation and explain how to use the gemsetup files.

NOTE

This procedure applies to users ONLY. Each user must perform this procedure before running GemStone.

1. Set the environment variable GEMSTONE to the *full pathname* (starting with a slash) of the GemStone/S 64 Bit 2.2.3 directory.

   C shell:

   % **setenv GEMSTONE** *InstallDir*/**GemStone64Bit2.2.3-x86_64.Linux**

   Bourne or Korn shell:

   $ **GEMSTONE=***InstallDir*/**GemStone64Bit2.2.3-x86_64.Linux**$ **export GEMSTONE**

17. Invoke the script gemsetup.

    C shell:

    % **source $GEMSTONE/bin/gemsetup.csh**

    Bourne or Korn shell:

    $ **. $GEMSTONE/bin/gemsetup.sh**

18. If you will use GemStone frequently, consider adding to your login shell's initialization file (.cshrc or .profile) the environment variable GEMSTONE and the command **gemsetup**. This way, the GemStone environment is automatically configured every time you log in or create a login shell. It overrides any other GemStone path settings.

    NOTE

    If you use the Korn shell and your .profile contains commands that are not POSIX-compliant, you might encounter errors when a shell is initialized. To remedy this situation, place the non-compliant commands within a conditional, such as the following:

    ```
    hash -r 2>/dev/null
    status=$?
    if [ $status -ne 0 ]; then
    ```

```
    # Place Korn-shell-specific initialization here
fi
```

Install the default TimeZone

GemStone / S 64 Bit is shipped with a default time zone of US Pacific. If you are in another Time Zone, edit the file `installtimezone.txt` in the GemStone `upgrade` directory, then file it in as SystemUser.

What Next?

This chapter has guided you through installation of GemStone / S 64 Bit 2.2.3 in an initial configuration that is sufficient to create a basic repository and begin setting up user accounts. The objective has been to get a simple, default configuration up and running.

You might consider performing the following tasks:

- To modify the initial object server configuration to one that is more efficient for your particular needs, refer to Chapter 1 of the *System Administration Guide for GemStone/S 64 Bit*. This chapter contains sample configurations, from small to large, and also contains detailed information about how to tailor these configurations to your own system.
- To modify the configuration of Gem session processes and to ensure that users have the necessary permissions to access the shared page cache and the extents, refer to Chapter 2 of the *System Administration Guide for GemStone/S 64 Bit*.
- If you are going to operate in a network environment, Chapter 3 of the *System Administration Guide for GemStone/S 64 Bit* has additional information about the GemStone network server (NetLDI), how to handle user authentication, how to share software over the network, and how to set up some common configurations.
- To start and stop the GemStone object server, refer to instructions in Chapter 4 of the *System Administration Guide for GemStone/S 64 Bit*.

Chapter 2

Upgrading from previous GemStone/S 64 Bit 2.x versions

This chapter describes how to upgrade an existing GemStone/S 64 Bit 2.x installation to GemStone/S 64 Bit version 2.2.3. GemStone/S 64 Bit version 2.2.3 supports upgrade from GemStone/S 64 Bit version2.1.4, and 2.2 and later. If you are upgrading from GemStone/S 64 Bit 1.x, which requires conversion, or from a different GemStone/S server product, see the appropriate chapter for conversion instructions.

As part of the upgrade process, you:

- Modify application classes and methods that run in GemStone/S 64 Bit so they work properly with the version 2.2.3 classes and methods.
- Modify your application's client code as necessary. This code may be in C, C++, or client Smalltalk.
- File out the changes to GemStone kernel classes.
- Upgrade your repository per the instructions starting on page 2-2.
- File in your GemStone kernel class changes.
- Recompile and relink any C or C++ user actions or client applications.
- Reload client Smalltalk images.

Upgrade Strategy

We recommend that you perform the upgrade twice: first a pilot upgrade and then the production upgrade. With this strategy, you can keep your version 2.x production system running while you familiarize yourself with the upgrade process. The pilot upgrade may be performed on any GemStone/S 64 Bit 2.x test repository.

Upgrade Procedure

The following list summarizes the steps necessary to perform the upgrade to GemStone/S 64 Bit version 2.2.3.

Prepare for Upgrade

Perform the following steps to prepare for the upgrade.

1. File out any modifications or additions you made to GemStone/S 64 Bit kernel class methods by using the Topaz command **fileout**. For more information about fileout, see the *GemStone Topaz Programming Environment*.

   You will need to carefully compare these changes with GemStone/S 64 Bit 2.2.3 kernel methods to determine whether your changes are still necessary or appropriate.

2. Install GemStone/S 64 Bit 2.2.3 to a new installation directory, separate from the installation directory for version 2.2.2, as described in Chapter 1 of this Installation Guide.

3. Configure GemStone/S 64 Bit 2.2.3 the way you expect to use it — that is, with the appropriate extent locations and sizes.

   If your transaction logs are on raw partitions, for best performance you may wish to install the optimized POSIX AIO library; see "Optimized POSIX Asynchronous I/O" on page 1-4 for more information.

4. Ensure that adequate space is available for extents, transaction logs, and a backup during the upgrade:

   ```
   % df
   ```

   You must provide space for the following:

   - Your version 2.2.2 extents and transaction logs.
   - Your version 2.2.3 extents and transaction logs.

5. Log in to the version 2.2.2 system and reset the SystemUser password to 'swordfish':

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

   The upgrade script logs in with the SystemUser account and the default password.

6. Halt all user activity on the repository you are going to upgrade:

   a. Log in to Topaz as DataCurator.

   b. Force all other users off the system:

   ```
   topaz 1> printit
   System stopUserSessions.
   %
   ```

   CAUTION

   You MUST file out any changes that you have made to the GemStone/S 64 Bit kernel classes in order to preserve these changes in version 2.2.3. Also, consider saving important modified files, such as configuration files, that will be overwritten during the upgrade.

7. File out any modifications or additions you made to GemStone/S 64 Bit version 2.2.2 kernel class methods by using the Topaz command **fileout**. For more information about fileout, see the *GemStone/S 64 Bit Topaz Programming Environment*.

8. You must now shut down the Stone:

   % **stopstone** *stone222*

   where *stone222* is the name of the version 2.2.2 stone on this machine. The repository must be cleanly shut down before it can be restarted under version 2.2.3

9. Set up the version 2.2.3 environment.

   Set the environment variables required for the upgrade.

   C shell:
   % **setenv GEMSTONE** *InstallDir223*
   % **set path = ($GEMSTONE/bin $path)**
   % **setenv upgradeLogDir** *tempDir*

   Bourne or Korn shell:
   $ **GEMSTONE=***InstallDir223*
   $ **export GEMSTONE**
   $ **export PATH=$GEMSTONE/bin:$PATH**
   $ **upgradeLogDir=***tempDir*
   $ **export upgradeLogDir**

   where *InstallDir223* is the GemStone/S 64 Bit version 2.2.3 installation and *tempDir* is a temporary directory for which you have write permission.

   NOTE

   Use a separate log directory for each repository you convert. A repository may contain multiple extents.

10. Copy your version 2.2.2 extent files into the location specified by the configuration file option `DBF_EXTENT_NAMES`:

    a. Using a text editor, open the configuration file that the version 2.2.2 repository uses.

    b. Locate the last occurrence of the option `DBF_EXTENT_NAMES`, and note its value, a list of `.dbf` files.

c. Copy each `.dbf` file to the noted location in the version 2.2.3 installation. For example:

```
% cp InstallDir222/data/extent0.dbf 223location
% cp InstallDir222/data/extent1.dbf 223location
% cp InstallDir222/data/extent2.dbf 223location
```

where *223location* is the location specified by `DBF_EXTENT_NAMES` in the configuration file that will be used in version 2.2.3.

11. Before upgrading, ensure that there are no transaction logs from a previous version of GemStone / S 64 Bit in any of the transaction log locations specified in the configuration file that will be used by version 2.2.3. Transaction logs from earlier versions are not compatible with version 2.2.3. If the transaction log directories will be reused for version 2.2.3, any transaction logs should be deleted or copied elsewhere.

Perform the Upgrade

1. Start the 2.2.3 Stone on the 2.2.2 extents you just copied:

   ```
   % startstone stoneName223
   ```

2. 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,

   ```
   % upgradeImage -s stoneName223
   ```

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

3. 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.

Restore Your Site-Specific Settings and Back Up the Repository

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

   If you use GemConnect, you must install it again at this time. Use the procedure in the installation guide for this product.

2. Log in to GemStone/S 64 Bit version 2.2.3 as DataCurator.
3. Change the password for SystemUser, which you changed to `swordfish` prior to the conversion, back to its previous value. Also, change the password for GcUser, which was reset by the conversion process, back to the version 2.2.2 value:

   ```
   topaz 1> printit
   (AllUsers userWithId: 'SystemUser') password: '222Password'.
   (AllUsers userWithId: 'GcUser') password: '222Password'.
   System commitTransaction
   %
   ```

   where *222Password* is the account password used in GemStone/S 64 Bit version 2.2.2.

   The upgraded repository is now usable. Other users can log in to assist with the following steps.
4. If you have modified any kernel class methods of the previous version, carefully compare your changes with version 2.2.3 kernel methods to see whether your changes are still necessary or appropriate. If so, file in the changes and commit.
5. Create a full backup of the upgraded repository. For details, see the *System Administration Guide for GemStone/S 64 Bit*.
6. If you are using GBS clients, configure these to use the version 2.2.3 client libraries. See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details.

Chapter 3

Converting from GemStone/S 64 Bit 1.x

This chapter describes how to convert an existing GemStone/S 64 Bit 1.x installation to GemStone/S 64 Bit version 2.2.3. Upgrading from GemStone/S 64 Bit 2.x does not require conversion; if you are upgrading from version 2.x, or converting from another GemStone/S server product, see the appropriate Chapter of this Installation Guide.

GemStone/S 64 Bit version 2.2.3 includes substantial changes from version 1.x. For details on these changes, see the *Porting Guide for GemStone/S 64 Bit 1.1.14*.

GemStone/S 64 Bit version 2.2.3 supports conversion from GemStone/S 64 Bit versions 1.1.14 and later.

As part of the conversion process, you:

- Modify classes and methods that run in GemStone so they work properly with the GemStone/S 64 Bit classes and methods.
- Modify your application's client code as necessary. This code may be in C, C++, or client Smalltalk.

  NOTE

  In order to use GemBuilder for Smalltalk (GBS) with the GemStone/S 64 Bit v2.2.3 server, you must install GBS version 7.1.2 or later, and then follow the configuration instructions in Chapter 6 of this Installation Guide.

- File out the changes to GemStone kernel classes.
- Convert and upgrade your repository in a multi-stage process, per the instructions starting on page 3-3.
- File in your GemStone kernel class changes.
- Recompile and relink any C user actions or client applications.
- Recompile and relink any C or C++ user actions or client applications.
- Reload client Smalltalk images.

The architecture of the repository in GemStone/S 64 Bit 2.2.3 is significantly different from GemStone/S 64 Bit 1.x. For that reason, the upgrade from v1.x to v2.2.3 is a conversion that cannot be done in place. The upgrade conversion process will construct a new GemStone/S 64 Bit repository, using the 1.x repository as a template. When the conversion is completed, both the 1.x and 2.2.3 repositories will exist. This means that enough disk space must be available to accommodate both repositories.

The GemStone/S 64 Bit 2.2.3 repository will be substantially larger than the GemStone/S 64 Bit 1.x repository. Repository size increases between 20% and 200% or more are likely, depending on the nature of the data in the repository.

The conversion process will not substantially modify the original 1.x repository. The 1.x repository will remain in a fully usable and coherent state after the conversion is completed.

Conversion Strategy

We recommend that you perform the upgrade conversion twice: first a pilot conversion and then the production conversion. With this strategy, you can keep your version 1.x production system running while you familiarize yourself with the conversion process. The steps are the same for the pilot and the production conversions; where there are difference, this is noted in the instructions.

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 version 2.2.3 system correctly.

Conversion Procedure

The conversion procedure from GemStone/S 64 Bit 1.x to GemStone/S 64 Bit 2.2.3 is performed in several stages. The first stage executes in the existing 1.x environment, that is, with $GEMSTONE pointing to the GemStone/S 64 Bit v1.x product tree. Do not reset $GEMSTONE until specified in the instructions. The following phases of conversion execute in the GemStone/S 64 Bit environment.

The following list summarizes the steps necessary to perform the conversion to GemStone/S 64 Bit v2.2.3.

NOTE

The following instructions use the version number 1.2 to represent any GemStone/S 64 Bit 1.x version, 1.1.14 or later. Any differences in the upgrade proceedure are noted in the instructions.

Preprocessing in Old Environment

1. Install GemStone / S 64 Bit 2.2.3 to a new installation directory, separate from the installation directory for 1.2, as described in Chapter 1 of this Installation Guide.

   Define a new environment variable, **$GEMSTONE_22**, to point to this product tree. At this point, do not reset $GEMSTONE or update the path to point to v2.2.3.

   For example, using the default GemStone / S 64 Bit 2.2.3 installation location from Chapter 1:

   C shell:

   ```
   % setenv GEMSTONE_22 <InstallDir>/GemStone64Bit2.2.3-x86_64.Linux
   ```

   Bourne or Korn shell:

   ```
   $ GEMSTONE_22=<InstallDir>/GemStone64Bit2.2.3-x86_64.Linux
   $ export GEMSTONE_22
   ```

2. Check the system configuration file used by your existing version 1.2 installation. If the environment variable $GEMSTONE is used in the paths for extent file locations in the DBF_EXTENT_NAMES, this must be changed to use the absolute path. The conversion will not proceed with $GEMSTONE in the extent file path / s.

3. If this is the pilot conversion, to verify the integrity of your v1.2 system, run an object audit and a page audit prior to conversion. For the page audit, which requires the stone to be shut down, you may choose to run this following the `convprep` step (step on page 3-4).

   For instructions in running object and page audits, see the *System Administration Guide for GemStone/S 64 Bit*

4. Optional: Set up classes for list instances.

   Some customers may need to perform application level object conversions on 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 behavior is not enabled by default. To enable, you must define a UserGlobal 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 **`convprep1x`** script, do the following:

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

   b. Login to the v1.2 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 **convprep1x** 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 final converted GemStone/S 64 Bit v2.2.3 repository.

5. Log in to the v1.2 system and reset the SystemUser password to 'swordfish':

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

The conversion scripts log in with the SystemUser account and the default password.

6. Halt all user activity on the repository you are going to convert:

   a. Log in to Topaz as DataCurator.

   b. Force all other users off the system:

```
topaz 1> printit
System stopUserSessions.
%
```

CAUTION

You MUST file out any changes that you have made to the GemStone/S 64 Bit kernel classes in order to preserve these changes in version 2.2.3. Also, consider saving important modified files, such as configuration files, that will be overwritten during the conversion.

7. File out any modifications or additions you made to GemStone/S 64 Bit kernel class methods by using the Topaz command **fileout**. For more information about fileout, see the *GemStone Topaz Programming Environment*.

8. Logout, and exit topaz.

9. Set the environment variable that will hold the upgrade log files.

   C shell:

   % **setenv upgradeLogDir** *tempDir*

   Bourne or Korn shell:

   $ **upgradeLogDir=***tempDir*
   $ **export upgradeLogDir**

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

NOTE

Use a separate log directory for each repository you convert.

Run the GemStone/S 64 Bit 2.2.3 script **convprep1x**. This script is in the upgrade directory of the 2.2.3 product tree; this should not be in your path, so it will need to be invoked with an explicit path. For example,

% **$GEMSTONE_22/upgrade/convprep1x**

This script has the following options:

```
convprep1x [-h] [-s <stonename>]
```
`-c` prewarm the cache using startcachewarmers (default: no prewarm)
`-h` prints this usage information.
`-n <numcachewarmers>` where <numcachewarmers> is the number of cachewarmer gems to start. Defaults to 1 if no number specified; has no effect if `-c` is not specified.
`-s <stoneName>` sets the name of the running v1.2 Stone to scan; if this option is not used, the script will default to **gs64stone.**

The **`convprep1x`** script scans the v1.2 repository and produces files that are used in the second phase of conversion. These files are written to **`$upgradeLogDir`**. The **`convprep1x`** script will write progress messages to stdout. When it completes, it will report:

```
convprep1x[INFO]: ...conversion preparation complete.
```

This script also shuts down the Stone.

Prepare for Conversion

1. Set up the configuration for GemStone/S 64 Bit 2.2.3. In particular, you should examine and update:

   a. DBF_EXTENT_SIZES. The conversion process creates new v2.2.3 extents based on the v1.2 extents; the v2.2.3 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.

   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. To ensure the performance of the v2.2.3 system equals that of the v1.2 system, the shared page cache size, for all shared caches including the gem server SPC, 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. Ensure that you have adequate disk space for the v2.2.3 extents in addition to the v1.2 extents, as well as for tranlogs (if not using `/dev/null`) and a backup following the conversion.

   For the pilot conversion, allow for a growth of as much as 200%. In other words, if your repository extents total size currently is 1GB, ensure you have at least 2GB free space for the

extents.

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

   C shell:
   % **setenv GEMSTONE** *InstallDir223*
   % **set path = ($GEMSTONE/bin $path)**

   Bourne or Korn shell:
   $ **GEMSTONE=***InstallDir223*
   $ **export GEMSTONE**
   $ **export PATH=$GEMSTONE/bin:$PATH**

   where *InstallDir223* is the GemStone/S 64 Bit version 2.2.3 installation. You will no longer need the $GEMSTONE_**22** environment variable.

   The $upgradeLogDir environment variable that was defined in the section 'Preprocessing in Old Environment' must be available and set to the same directory.

4. Copy a clean v2.2.3 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 2.2.3 repository. For example,

   % **cp $GEMSTONE/bin/extent0.bdf $GEMSTONE/data**
   % **chmod a+w $GEMSTONE/data/extent0.dbf**

   New extent files for v2.2.3 will be created as part of conversion. **Do not** copy your version 1.2 extent files. All 1.2 extents should be available in their 1.2 locations.

Perform the Conversion and Upgrade

1. Start the 2.2.3 Stone

   % **startstone** *stoneName223*

2. Extent conversion is performed by the script **conv1xTo2x**. This script is in the v2.2.3 product tree, and does not require a explicit path.

   This script requires as an argument the configuration file used by the v1.2 installation. This configuration file must not have the extent file names specified using $GEMSTONE.

   It also has optional switches to specify the Stone name, the number of gems performing the conversion, and the number of OOPs per commit.

   This script also includes the -L switch that enables recording of all references to Large Integers. The conversion process does not convert instances of LargePositive- or LargeNegativeIntegers that fall within the new SmallIntegers range. This can be done as an optional manual step following conversion. Note that using the -L option will slow down the conversion process, since each object must be checked for Large Integer references.

   If you are upgrading from 1.2 or later, you may also use the -F switch that enables recording of all references to SmallFloats or Floats. Instances of these classes that are within the range of the new SmallDouble class are not converted by the conversion process. Using the -F switch collects these instances for subsequent manual conversion, as with Large Integers.

Use of this switch requires appropriate code in the originating repository, so this flag does not work for repositories prior to 1.2. Similarly to `-L`, this will slow down conversion.

`conv1xTo2x -e <oldSysConf> [-h] [-s <stoneName>] [-n <numConversionGems>] [-o <oopsPerCommit>] [-L] [-F]`
`-e <oldSysConf>` specifies the configuration file to use. This configuration file must contain the list of 1.2 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; if this option is not used, the script will default to **gs64stone.**
`-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.
`-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: `$upgradeLogDir/AllLrgIntRefs.bm.`
`-F` During conversion, determine which objects reference one or more instances of Float for SmallFloat and write the object IDs to the binary bitmap file: `$upgradeLogDir/AllFloatRefs.bm.` (only for use when upgrading from 1.2 or later)

For example,

% **conv1xTo2x** -e *installLocation12/**data/system.conf*** **-s** *stoneName223*

The **conv1xTo2x** script will write progress messages to stdout. When it completes, it will report:

```
conv1xTo2x[INFO]: Successful conversion.
```

The `conv1xTo2x` script shuts down the stone on completion.

3. Restart the 2.2.3 Stone

   % **startstone** *stoneName223*

4. Run the script that performs the kernel class filein to upgrade the image.

   The image upgrade is performed by the script `upgradeImageFrom1x`. 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.

   `upgradeImageFrom1x [-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,

   % **upgradeImageFrom1x -s** *stoneName223*

   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. To convert large objects found during the conversion process, run the `postconv` script. This script has optional flags.

   `postconv [-h][-s <stoneName>][-n <numberOfSessions>]`
   `-h` prints this usage information
   `-s <stoneName>` sets the name of the running stone; if this option is not used, the script will default to **gs64stone.**
   `-n <numberOfSessions>` 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 <tmpObjCacheSize>` 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 100000.
   `-c <numCacheWamerGems>` specified the number of cache warmer gems to start before starting post conversion; if this is not specified, no cache warmer gems are started.

   For example,

   % **`postconv -s`** *stoneName223*

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

   ```
   postconv[INFO]: conversion process complete
   ```

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

7. If you set up `#ConversionClassesToFind` in your v1.2 repository, be sure to remove this variable to avoid retaining references. Log in as SystemUser, execute:

   ```
   UserGlobals removeKey: #ConversionClassesToFind.
   ```

   and commit.

8. Optional. If you set up `#ConversionClassesToFind` in your v1.2 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 64 Bit v1.2 OOP of the class.
   <*newOop*> – GemStone/S 64 Bit v2.2.3 OOP of the class.

GemStone/S 64 Bit v2.2.3 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.

9. Optional: if you specified tracking of references to Large Integers or Floats, by using the `-L` or `-F` option during the `conv1xTo2x` script execution, the results will be written to bitmap files:

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

   GemStone/S 64 Bit v2.2.3 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 from Large Integers to SmallIntegers, or from Float or SmallFloat to SmallDouble.

Restore Your Site-Specific Settings and Back Up the Repository

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

   If you use GemConnect or GemBuilder for Java, you must install it again at this time. Use the procedure in the installation guide for this product.

2. Log in to GemStone/S 64 Bit version 2.2.3 as DataCurator.

3. Change the password for SystemUser, which you changed to `swordfish` prior to the conversion, back to its previous value. Also, change the password for GcUser, which was reset by the conversion process, back to the version previous value:

   ```
   topaz 1> printit
   (AllUsers userWithId: 'SystemUser') password: '12Password'.
   (AllUsers userWithId: 'GcUser') password: '12Password'.
   System commitTransaction
   %
   ```

   where *12Password* is the account password used in the original GemStone/S 64 Bit v1.2 repository.

   The converted repository is now usable. Other users can log in to assist with the following steps.

4. If you have modified any kernel class methods of the previous version, carefully compare your changes with version 2.2.3 kernel methods to see whether your changes are still necessary or appropriate. If so, file in the changes and commit.

5. Create a full backup of the converted repository. For details, see the *System Administration Guide for GemStone/S 64 Bit*.

6. If your transaction logs are on raw partitions, for best performance you may wish to install the optimized POSIX AIO library; see “Optimized POSIX Asynchronous I/O” on page 1-4 for more information.

7. If you are using GBS clients, configure these to use the version 2.2.3 client libraries. See Chapter 6, ’Configuring GBS for GemStone/S 64 Bit’ for details

Chapter 4

Converting from GemStone/S 6.1.x

This chapter describes how to convert an existing GemStone/S installation to GemStone/S 64 Bit version 2.2.3. This chapter covers conversion from 6.1.x; if you are converting from 6.2, see the appropriate chapter of this Installation Guide. The process is very similar, but separate conversion and upgrade scripts are provided.

GemStone/S 64 Bit version 2.2.3 supports conversion directly from GemStone/S version 6.1.5 or later. The instructions in this chapter use "6.1.5" for any 6.1.x release, 6.1.5 or later. If you are running an earlier version of GemStone/S, you must first upgrade to version 6.1.5 following the instructions in that release.

While GemStone/S is similar to GemStone/S 64 Bit, there are differences in code and administration. Conversion to GemStone/S 64 Bit requires modifications to your existing application. Refer to *Porting Guide for GemStone/S 6.1.5* for details on the changes.

As part of the conversion process, you:

- Modify classes and methods that run in GemStone so they work properly with the GemStone/S 64 Bit classes and methods.
- Modify your application's client code as necessary. This code may be in C, C++, or client Smalltalk.

  NOTE

  In order to use GemBuilder for Smalltalk (GBS) with the GemStone/S 64 Bit v2.2.3 server, you must install GBS version 7.1.2 or later, and then follow the configuration instructions in Chapter 6 of this Installation Guide.

- Ensure you have no methods whose compiled size is larger than 64K. Methods larger than 64K will not fail during conversion, but cannot be loaded or executed in GemStone/S 64 Bit. For more information, see "Detecting Oversize Methods" on page 4-10.
- Ensure that you have the TimeZone2007 patch installed into your GemStone/S repository.

- File out the changes to GemStone kernel classes.
- Convert and upgrade your repository in a multi-stage process per the instructions starting on page 4-3.
- File in your GemStone kernel class changes.
- Recompile and relink any C user actions or client applications.
- Recompile and relink any C or C++ user actions or client applications.
- Reload client Smalltalk images.

The architecture of the repository in GemStone/S 64 Bit 2.2.3 is substantially different from GemStone/S 6.1.5. For that reason, the upgrade from v6.1.5 to v2.2.3 is a conversion that cannot be done in place. The upgrade conversion process will construct a new GemStone/S 64 Bit repository, using the 6.1.5 repository as a template. When the conversion is completed, both the 6.1.5 and 2.2.3 repositories will exist. This means that enough disk space must be available to accommodate both repositories.

The GemStone/S 64 Bit 2.2.3 repository will be substantially larger than the GemStone/S 6.1.5 repository. Repository size increases between 20% and 200% or more are likely, depending on the nature of the data in the repository.

The conversion process will not substantially modify the original 6.1.5 repository. The 6.1.5 repository will remain in a fully usable and coherent state after the conversion is completed.

Conversion Strategy

We recommend that you perform the upgrade conversion twice: first a pilot conversion and then the production conversion. With this strategy, you can keep your version 6.1.5 production system running while you familiarize yourself with the conversion process. The steps are the same for the pilot and the production conversions; where there are difference, this is noted in the instructions.

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 version 2.2.3 system correctly.

Conversion Procedure

The conversion procedure from GemStone/S 6.1.5 to GemStone/S 64 Bit 2.2.3 is performed in several stages. The first stage executes in the existing 6.1.5 environment, that is, with $GEMSTONE pointing to the GemStone/S v6.1.5 product tree. Do not reset $GEMSTONE until specified in the instructions. The following phases of conversion execute in the GemStone/S 64 Bit environment.

The following list summarizes the steps necessary to perform the conversion to GemStone/S 64 Bit v2.2.3.

Preprocessing in Old Environment

1. File out any modifications or additions you made to GemStone / S kernel class methods by using the Topaz command **fileout**. For more information about fileout, see the *GemStone Topaz Programming Environment*.

   You will need to carefully compare these changes with GemStone / S 64 Bit 2.2.3 kernel methods to determine whether your changes are still relevant or appropriate.

2. Verify that you have no methods larger than 64K. This must be done prior to conversion. Oversize methods do not cause an error in the conversion, but cannot be loaded or executed in the converted repository. GemStone / S 64 Bit 2.2.3 provides a tool to assist in detecting oversize methods - see “Detecting Oversize Methods” on page 4-10

3. Verify that you have the TimeZone2007 patch installed. This patch provides a new implementation of TimeZone, supporting the 2007 Daylight Savings Time changes in the US. You must install this patch prior to upgrading, even if you do not require the TimeZone changes. The patch is available for download at:

```
http://support.gemstone.com/gemstone_s/downloads/patches/timezone/index.html
```

   You do not need to execute the script `timezone.txt`, nor install a new TimeZone instance, in order to perform the conversion.

   NOTE

   Attempts to convert without previously installing the TimeZone2007 patch may coredump.

4. Install GemStone / S 64 Bit 2.2.3 to a new installation directory, separate from the installation directory for 6.1.5, as described in Chapter 1 of this Installation Guide.

   Define a new environment variable, `$GEMSTONE_22`, to point to this product tree. At this point, do not reset `$GEMSTONE` or update the path to point to v2.2.3.

   For example, using the default GemStone / S 64 Bit 2.2.3 installation location from Chapter 1:

   C shell:

   ```
   % setenv GEMSTONE_22 <223Location>/GemStone64Bit2.2.3-
   x86_64.Linux/
   ```

   Bourne or Korn shell:

   ```
   $ GEMSTONE_22=<223Location>/GemStone64Bit2.2.3-x86_64.Linux/
   $ export GEMSTONE_22
   ```

5. Check the system configuration file used by your existing version 6.1.5 installation. If the environment variable `$GEMSTONE` is used in the paths for extent file locations in the DBF_EXTENT_NAMES, this must be changed to use the absolute path. The conversion will not proceed with `$GEMSTONE` in the extent file path / s.

6. If this is the pilot conversion, to verify the integrity of your v6.1.5 system, run an object audit and a page audit prior to conversion. For the page audit, which requires

the stone to be shut down, you may choose to run this following the **convprep61** step (step 14 on page 4-5).

For instructions in running object and page audits, see the *System Adminitration Guide.*

7. Prior to conversion, ensure there are no dead objects in your repository.
8. Optional: Set up classes for list instances.

   Some customers may need to perform application level object conversions on 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 behavior is not enabled by default. To enable, you must define a UserGlobal 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 **convprep61** script, do the following:

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

   b. Login to the v6.1.5 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 **convprep61** 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 final converted GemStone/S 64 Bit v2.2.3 repository.

9. Log in to the v6.1.5 system and reset the SystemUser password to '`swordfish`':

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

   The conversion scripts log in with the SystemUser account and the default password.

10. Halt all user activity on the repository you are going to convert:

    a. Log in to Topaz as DataCurator.

    b. Force all other users off the system:

```
topaz 1> printit
System stopOtherSessions.
%
```

CAUTION

You MUST file out any changes that you have made to the GemStone/S kernel classes in order to preserve these changes in version 2.2.3. Also, consider saving

important modified files, such as configuration files, that will be overwritten during the conversion.

11. File out any modifications or additions you made to GemStone/S kernel class methods by using the Topaz command **fileout**. For more information about fileout, see the *GemStone Topaz Programming Environment*.
12. Logout, and exit topaz.
13. Set the environment variable that will hold the upgrade log files.

    C shell:

    % **setenv upgradeLogDir** *tempDir*

    Bourne or Korn shell:

    $ **upgradeLogDir=***tempDir*
    $ **export upgradeLogDir**

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

    NOTE
    Use a separate log directory for each repository you convert.

14. Run the GemStone/S 64 Bit 2.2.3 script **convprep61**. This script is in the upgrade directory of the 2.2.3 product tree; this should not be in your path, so it will need to be invoked with an explicit path. For example,

    % **$GEMSTONE_22/upgrade/convprep61**

    This script has the following options:

    convprep61 [-h] [-s <*stonename*>]
    -h prints this usage information.
    -s <*stoneName*> sets the name of the running v6.1.5 Stone to scan; if this option is not used, the script will default to **gemserver61.**

    The **convprep61** script scans the v6.1.5 repository and produces files that are used in the second phase of conversion. These files are written to **$upgradeLogDir**. The **convprep61** script will write progress messages to stdout. When it completes, it will report:

    convprep61[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.

Prepare for Conversion

1. Set up the configuration for GemStone / S 64 Bit 2.2.3. In particular, you should examine and update:

   a. DBF_EXTENT_SIZES. The conversion process creates new v2.2.3 extents based on the v6.1.5 extents; the v2.2.3 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.

   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. To ensure the performance of the v2.2.3 system equals that of the v6.1.5 system, the shared page cache size, for all shared caches including the gem server SPC, 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. Ensure that you have adequate disk space for the v2.2.3 extents in addition to the v6.1.5 extents, as well as for tranlogs (if not using `/dev/null`) and a backup following the conversion.

   For the pilot conversion, allow for a growth of as much as 200%. In other words, if your repository extents total size currently is 1GB, ensure you have at least 2GB free space for the 2.2.3 extents.

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

   C shell:
   % **`setenv GEMSTONE`** *InstallDir223*
   % **`set path = ($GEMSTONE/bin $path)`**

   Bourne or Korn shell:
   $ **`GEMSTONE=`***InstallDir223*
   $ **`export GEMSTONE`**
   $ **`export PATH=$GEMSTONE/bin:$PATH`**

   where *InstallDir223* is the GemStone / S 64 Bit version 2.2.3 installation. You will no longer need the `$GEMSTONE_22` environment variable.

   The `$upgradeLogDir` environment variable that was defined in the section 'Preprocessing in Old Environment' (see step 13 on page 4-5) must be available and set to the same directory.

4. Copy a clean v2.2.3 `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 2.2.3 repository. For example,

   ```
   % cp $GEMSTONE/bin/extent0.bdf $GEMSTONE/data
   % chmod a+w $GEMSTONE/data/extent0.dbf
   ```

   New extent files for v2.2.3 will be created as part of conversion. **Do not** copy your version 6.1.5 extent files. All 6.1.5 extents should be available in their 6.1.5 locations.

Perform the Conversion and Upgrade

1. Start the 2.2.3 Stone

   % **startstone** *stoneName22*

2. Extent conversion is performed by the script **conv61To2x**. This script is in the v2.2.3 product tree, and does not require a explicit path.

   This script requires as an argument the configuration file used by the v6.1.5 installation. This configuration file must not have the extent file names specified using $GEMSTONE.

   It also has optional switches to specify the Stone name, the number of gems performing the conversion, and the number of OOPs per commit.

   This script also includes the `-L` switch that enables recording of all references to Large Integers. The conversion process does not convert instances of LargePositive- or LargeNegativeIntegers that fall within the new SmallIntegers range. This can be done as an optional manual step following conversion. Note that using the `-L` option will slow down the conversion process, since each object must be checked for Large Integer references.

   `conv61To2x -e <oldSysConf> [-h] [-s <stoneName>] [-n <numConversionGems>] [-o <oopsPerCommit>] [-L]`
   `-e <oldSysConf>` specifies the configuration file to use. This configuration file must contain the list of 6.1.5 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; if this option is not used, the script will default to **gs64stone.**
   `-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.
   `-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,

   % **conv61To2x** `-e` *installLocation61***/data/system.conf -s** *stoneName22*

   The **conv61To2x** script will write progress messages to stdout. When it completes, it will report:

   ```
   conv61To2x[INFO]: Successful conversion.
   ```

The conv61To2x script shuts down the stone on completion.

3. Restart the 2.2.3 Stone

   % **startstone** *stoneName22*

4. Run the script that performs the kernel class filein to upgrade the image.

   The image upgrade is performed by the script upgradeImageFrom61. 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.

   upgradeImageFrom61 [-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,

   % **upgradeImageFrom61 -s** *stoneName22*

   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. To convert large objects found during the conversion process, run the postconv script. This script has optional flags to specify stone name and the number of sessions to use.

   postconv [-h][-s <*stoneName*>][-n <*numberOfSessions*>]
   -h prints this usage information
   -s <stoneName> sets the name of the running stone; if this option is not used, the script will default to **gs64stone.**
   -n <numberOfSessions> 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 <tmpObjCacheSize> 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 100000.
   -c <numCacheWamerGems> specified the number of cache warmer gems to start before starting post conversion; if this is not specified, no cache warmer gems are started.

   For example,

   % **postconv -s** *stoneName22*

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

   postconv[INFO]: conversion process complete

6. If this is the pilot conversion, run an object audit and a page audit to verify the integrity of the repository following the conversion.
7. If you set up #ConversionClassesToFind in your v6.1.5 repository, be sure to remove this variable to avoid retaining references. Log in as SystemUser, execute:

   ```
   UserGlobals removeKey: #ConversionClassesToFind.
   ```

   and commit.
8. Optional. If you set up #ConversionClassesToFind in your v6.1.5 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 v6.1.5 OOP of the class.
   <*newOop*> – GemStone/S 64 Bit v2.2.3 OOP of the class.

   GemStone/S 64 Bit v2.2.3 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.
9. Optional: if you specified tracking of references to Large Integers, by using the -L option during the conv61To2x script execution, the results will be written to a bitmap file:

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

   GemStone/S 64 Bit v2.2.3 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 from Large Integers to SmallIntegers.

Restore Your Site-Specific Settings and Back Up the Repository

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

   If you use GemConnect or GemBuilder for Java, you must install it again at this time. Use the procedure in the installation guide for that product.
2. Log in to GemStone/S 64 Bit version 2.2.3 as DataCurator.
3. Change the password for SystemUser, which you changed to swordfish prior to the conversion, back to its previous value. Also, change the password for GcUser, which was reset by the conversion process, back to the version previous value:

   ```
   topaz 1> printit
   (AllUsers userWithId: 'SystemUser') password: '61Password'.
   (AllUsers userWithId: 'GcUser') password: '61Password'.
   System commitTransaction
   %
   ```

   where *61Password* is the account password used in the original GemStone/S v6.1.5 repository.

The converted repository is now usable. Other users can log in to assist with the following steps.

4. If you have modified any kernel class methods of the previous version, carefully compare your changes with version 2.2.3 kernel methods to see whether your changes are still necessary or appropriate. If so, file in the changes and commit.
5. Create a full backup of the converted repository. For details, see the *System Administration Guide for GemStone/S 64 Bit*.
6. If you are using GBS clients, configure these to use the version 2.2.3 client libraries. See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details.
7. If your transaction logs are on raw partitions, for best performance you may wish to install the POSIX AIO library; see "Optimized POSIX Asynchronous I/O" on page 1-4 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 6.1.5 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 2.2.3 provides a script to detect if any methods exceed the size limit. To use this script, the GemStone/S version 6.1.5 repository that you wish to verify must be running, and not yet converted to GemStone/S 64 Bit, and GemStone/S 64 Bit 2.2.3 must have been installed.

1. Ensure that a stone is running on the GemStone/S version 6.1.5 repository.
2. Verify that GemStone/S 64 Bit 2.2.3 is installed
3. Start topaz, and log in to the GemStone/S version 6.1.5 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 2.2.3.
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.
7. 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.

Chapter 5

Converting from GemStone/S 6.2

This chapter describes how to convert an existing GemStone/S installation to GemStone/S 64 Bit version 2.2.3. This chapter covers conversion from 6.2; if you are converting from 6.1.x, see the appropriate chapter of this Installation Guide. The process is very similar, but separate conversion and upgrade scripts are provided.

While GemStone/S is similar to GemStone/S 64 Bit, there are differences in code and administration. Conversion to GemStone/S 64 Bit requires modifications to your existing application. Refer to *Porting Guide for GemStone/S 6.1.5* for details on the changes.

As part of the conversion process, you:

- Modify classes and methods that run in GemStone so they work properly with the GemStone/S 64 Bit classes and methods.
- Modify your application's client code as necessary. This code may be in C, C++, or client Smalltalk.

  NOTE

  In order to use GemBuilder for Smalltalk (GBS) with the GemStone/S 64 Bit v2.2.3 server, you must install GBS version 7.1.2 or later, and then follow the configuration instructions in Chapter 6 of this Installation Guide.

- Ensure you have no methods whose compiled size is larger than 64K. Methods larger than 64K will not fail during conversion, but cannot be loaded or executed in GemStone/S 64 Bit. For more information, see "Detecting Oversize Methods" on page 5-10.
- File out the changes to GemStone kernel classes.
- Convert and upgrade your repository in a multi-stage process per the instructions starting on page 5-3.
- File in your GemStone kernel class changes.
- Recompile and relink any C user actions or client applications.

- Recompile and relink any C or C++ user actions or client applications.
- Reload client Smalltalk images.

The architecture of the repository in GemStone/S 64 Bit 2.2.3 is substantially different from GemStone/S 6.2. For that reason, the upgrade from v6.2 to v2.2.3 is a conversion that cannot be done in place. The upgrade conversion process will construct a new GemStone/S 64 Bit repository, using the 6.2 repository as a template. When the conversion is completed, both the 6.2 and 2.2.3 repositories will exist. This means that enough disk space must be available to accommodate both repositories.

The GemStone/S 64 Bit 2.2.3 repository will be substantially larger than the GemStone/S 6.2 repository. Repository size increases between 20% and 200% or more are likely, depending on the nature of the data in the repository.

The conversion process will not substantially modify the original 6.2 repository. The 6.2 repository will remain in a fully usable and coherent state after the conversion is completed.

Conversion Strategy

We recommend that you perform the upgrade conversion twice: first a pilot conversion and then the production conversion. With this strategy, you can keep your version 6.2 production system running while you familiarize yourself with the conversion process. The steps are the same for the pilot and the production conversions; where there are difference, this is noted in the instructions.

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 version 2.2.3 system correctly.

Conversion Procedure

The conversion procedure from GemStone/S 6.2 to GemStone/S 64 Bit 2.2.3 is performed in several stages. The first stage executes in the existing 6.2 environment, that is, with $GEMSTONE pointing to the GemStone/S v6.2 product tree. Do not reset $GEMSTONE until specified in the instructions. The following phases of conversion execute in the GemStone/S 64 Bit environment.

The following list summarizes the steps necessary to perform the conversion to GemStone/S 64 Bit v2.2.3.

Preprocessing in Old Environment

1. File out any modifications or additions you made to GemStone / S kernel class methods by using the Topaz command **fileout**. For more information about fileout, see the *GemStone Topaz Programming Environment*.

   You will need to carefully compare these changes with GemStone / S 64 Bit 2.2.3 kernel methods to determine whether your changes are still relevant or appropriate.

2. Verify that you have no methods larger than 64K. This must be done prior to conversion. Oversize methods do not cause an error in the conversion, but cannot be loaded or executed in the converted repository. GemStone / S 64 Bit 2.2.3 provides a tool to assist in detecting oversize methods - see “Detecting Oversize Methods” on page 5-10

3. Install GemStone / S 64 Bit 2.2.3 to a new installation directory, separate from the installation directory for 6.2, as described in Chapter 1 of this Installation Guide.

   Define a new environment variable, `$GEMSTONE_22`, to point to this product tree. At this point, do not reset `$GEMSTONE` or update the path to point to v2.2.3.

   For example, using the default GemStone / S 64 Bit 2.2.3 installation location from Chapter 1:

   C shell:

   ```
   % setenv GEMSTONE_22 <223Location>/GemStone64Bit2.2.3-
   x86_64.Linux/
   ```

   Bourne or Korn shell:

   ```
   $ GEMSTONE_22=<223Location>/GemStone64Bit2.2.3-x86_64.Linux/
   $ export GEMSTONE_22
   ```

4. Check the system configuration file used by your existing version 6.2 installation. If the environment variable `$GEMSTONE` is used in the paths for extent file locations in the DBF_EXTENT_NAMES, this must be changed to use the absolute path. The conversion will not proceed with `$GEMSTONE` in the extent file path / s.

5. If this is the pilot conversion, to verify the integrity of your v6.2 system, run an object audit and a page audit prior to conversion. For the page audit, which requires the stone to be shut down, you may choose to run this following the **`convprep62`** step (step 13 on page 5-5).

   For instructions in running object and page audits, see the *System Adminitration Guide.*

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

7. Optional: Set up classes for list instances.

   Some customers may need to perform application level object conversions on 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 behavior is not enabled by default. To enable, you must define a UserGlobal 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 **`convprep62`** script, do the following:

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

b. Login to the v6.2 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 **`convprep62`** 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 final converted GemStone/S 64 Bit v2.2.3 repository.

8. Log in to the v6.2 system and reset the SystemUser password to '`swordfish`':

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

The conversion scripts log in with the SystemUser account and the default password.

9. Halt all user activity on the repository you are going to convert:

a. Log in to Topaz as DataCurator.

b. Force all other users off the system:

```
topaz 1> printit
System stopOtherSessions.
%
```

CAUTION

You MUST file out any changes that you have made to the GemStone/S kernel classes in order to preserve these changes in version 2.2.3. Also, consider saving important modified files, such as configuration files, that will be overwritten during the conversion.

10. File out any modifications or additions you made to GemStone/S kernel class methods by using the Topaz command **fileout**. For more information about fileout, see the *GemStone Topaz Programming Environment*.

11. Logout, and exit topaz.

12. Set the environment variable that will hold the upgrade log files.

C shell:

% **`setenv upgradeLogDir`** *tempDir*

Bourne or Korn shell:

$ **`upgradeLogDir=`***tempDir*
$ **`export upgradeLogDir`**

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

NOTE

Use a separate log directory for each repository you convert.

13. Run the GemStone / S 64 Bit 2.2.3 script **convprep62**. This script is in the upgrade directory of the 2.2.3 product tree; this should not be in your path, so it will need to be invoked with an explicit path. For example,

    % **$GEMSTONE_22/upgrade/convprep62**

    This script has the following options:

    convprep62 [-h] [-s <*stonename*>]
    -h prints this usage information.
    -s <*stoneName*> sets the name of the running v6.2 Stone to scan; if this option is not used, the script will default to **gemserver62.**

    The **convprep62** script scans the v6.2 repository and produces files that are used in the second phase of conversion. These files are written to **$upgradeLogDir**. The **convprep62** script will write progress messages to stdout. When it completes, it will report:

    convprep62[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.

Prepare for Conversion

1. Set up the configuration for GemStone / S 64 Bit 2.2.3. In particular, you should examine and update:

   a. DBF_EXTENT_SIZES. The conversion process creates new v2.2.3 extents based on the v6.2 extents; the v2.2.3 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.

   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. To ensure the performance of the v2.2.3 system equals that of the v6.2 system, the shared page cache size, for all shared caches including the gem server SPC, 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. Ensure that you have adequate disk space for the v2.2.3 extents in addition to the v6.2 extents, as well as for tranlogs (if not using `/dev/null`) and a backup following the conversion.

   For the pilot conversion, allow for a growth of as much as 200%. In other words, if your repository extents total size currently is 1GB, ensure you have at least 2GB free space for the 2.2.3 extents.

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

   C shell:
   % **`setenv GEMSTONE`** *InstallDir223*
   % **`set path = ($GEMSTONE/bin $path)`**

   Bourne or Korn shell:
   $ **`GEMSTONE=`***InstallDir223*
   $ **`export GEMSTONE`**
   $ **`export PATH=$GEMSTONE/bin:$PATH`**

   where *InstallDir223* is the GemStone/S 64 Bit version 2.2.3 installation. You will no longer need the `$GEMSTONE_22` environment variable.

   The `$upgradeLogDir` environment variable that was defined in the section 'Preprocessing in Old Environment' (see step 12 on page 5-4) must be available and set to the same directory.

4. Copy a clean v2.2.3 `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 2.2.3 repository. For example,

   % **`cp $GEMSTONE/bin/extent0.bdf $GEMSTONE/data`**
   % **`chmod a+w $GEMSTONE/data/extent0.dbf`**

   New extent files for v2.2.3 will be created as part of conversion. **Do not** copy your version 6.2 extent files. All 6.2 extents should be available in their 6.2 locations.

Perform the Conversion and Upgrade

1. Start the 2.2.3 Stone

   % **`startstone`** *stoneName22*

2. Extent conversion is performed by the script **`conv62To2x`**. This script is in the v2.2.3 product tree, and does not require a explicit path.

   This script requires as an argument the configuration file used by the v6.2 installation. This configuration file must not have the extent file names specified using $GEMSTONE.

It also has optional switches to specify the Stone name, the number of gems performing the conversion, and the number of OOPs per commit.

This script also 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 LargePositive- 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.

conv62To2x -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 6.2 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; if this option is not used, the script will default to **gs64stone.**
-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 for 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,

% **conv62To2x** -e *installLocation62/**data/system.conf*** **-s** *stoneName22*

The **conv62To2x** script will write progress messages to stdout. When it completes, it will report:

```
conv62To2x[INFO]: Successful conversion.
```

The conv62To2x script shuts down the stone on completion.

3. Restart the 2.2.3 Stone

   % **startstone** *stoneName22*

4. Run the script that performs the kernel class filein to upgrade the image.

   The image upgrade is performed by the script upgradeImageFrom62. 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.

   upgradeImageFrom62 [-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,

% **upgradeImageFrom62 -s** *stoneName223*

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. To convert large objects found during the conversion process, run the `postconv` script. This script has optional flags to specify stone name and the number of sessions to use.

   `postconv [-h][-s <stoneName>][-n <numberOfSessions>]`
   `-h` prints this usage information
   `-s <stoneName>` sets the name of the running stone; if this option is not used, the script will default to **gs64stone.**
   `-n <numberOfSessions>` 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 <tmpObjCacheSize>` 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 100000.
   `-c <numCacheWamerGems>` specified the number of cache warmer gems to start before starting post conversion; if this is not specified, no cache warmer gems are started.

   For example,

   % **postconv -s** *stoneName22*

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

   ```
   postconv[INFO]: Success! Conversion process from GemStone 6.2.x
   to GemStone64 2.x completed.
   ```

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

7. If you set up `#ConversionClassesToFind` in your v6.2 repository, be sure to remove this variable to avoid retaining references. Log in as SystemUser, execute:

   ```
   UserGlobals removeKey: #ConversionClassesToFind.
   ```

   and commit.

8. Optional. If you set up `#ConversionClassesToFind` in your v6.2 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 v6.2 OOP of the class.
<*newOop*> – GemStone/S 64 Bit v2.2.3 OOP of the class.

GemStone/S 64 Bit v2.2.3 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.

9. Optional: if you specified tracking of references to Large Integers or Floats, by using the `-L` or `-F` options during the `conv62To2x` script execution, the results will be written to a bitmap file:

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

GemStone/S 64 Bit v2.2.3 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.

Restore Your Site-Specific Settings and Back Up the Repository

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

   If you use GemConnect or GemBuilder for Java, you must install it again at this time. Use the procedure in the installation guide for that product.

2. Log in to GemStone/S 64 Bit version 2.2.3 as DataCurator.

3. Change the password for SystemUser, which you changed to `swordfish` prior to the conversion, back to its previous value. Also, change the password for GcUser, which was reset by the conversion process, back to the version previous value:

```
topaz 1> printit
(AllUsers userWithId: 'SystemUser') password: '62Password'.
(AllUsers userWithId: 'GcUser') password: '62Password'.
System commitTransaction
%
```

   where *62Password* is the account password used in the original GemStone/S v6.2 repository.

   The converted repository is now usable. Other users can log in to assist with the following steps.

4. If you have modified any kernel class methods of the previous version, carefully compare your changes with version 2.2.3 kernel methods to see whether your changes are still necessary or appropriate. If so, file in the changes and commit.

5. Create a full backup of the converted repository. For details, see the *System Administration Guide for GemStone/S 64 Bit*.

6. If you are using GBS clients, configure these to use the version 2.2.3 client libraries. See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details.

7. If your transaction logs are on raw partitions, for best performance you may wish to install the POSIX AIO library; see "Optimized POSIX Asynchronous I/O" on page 1-4 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 6.2 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 2.2.3 provides a script to detect if any methods exceed the size limit. To use this script, the GemStone/S version 6.2 repository that you wish to verify must be running, and not yet converted to GemStone/S 64 Bit, and GemStone/S 64 Bit 2.2.3 must have been installed.

1. Ensure that a stone is running on the GemStone/S version 6.2 repository.
2. Verify that GemStone/S 64 Bit 2.2.3 is installed
3. Start topaz, and log in to the GemStone/S version 6.2 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 2.2.3.
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.
7. 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.

Chapter 6

Configuring GBS for GemStone/S 64 Bit

This chapter describes how to configure your GemBuilder for Smalltalk (GBS) application to run with GemStone/S 64 Bit version 2.2.3.

NOTE

GemStone/S 64 Bit version 2.2.3 requires the use of GBS version 7.1.2 or later. Versions earlier than 7.1.1 cannot log in to GemStone/S 64 Bit 2.2.3; version 7.1.1 may be usable, but some GBS behavior such as the Segment Tool will be incorrect for versions prior to 7.1.2. For upgrade instructions, refer to the GemBuilder for Smalltalk Installation Guide.

For the supported versions of VisualWorks client Smalltalk, see "*Supported platforms*" on page 6-2

In addition to using the appropriate version of GBS, you must use GemStone/S 64 Bit 2.2.3 client libraries to be able to log in to the GemStone/S 64 Bit 2.2.3 server. Applications using GBS version 7.x can log in to either GemStone/S, GemStone/S 2G, or GemStone/S 64 Bit repositories (but not simultaneously), provided that the appropriate client libraries for each are used.

GemStone/S 64 Bit 2.2.3 supports only RPC logins from GBS; linked logins are not supported.

On Unix platforms, GemStone/S 64 Bit 2.2.3 provides both 64-bit libraries and 32-bit libraries. To load these libraries into the VisualWorks 32-bit process, the 32-bit libraries must be used. 64-bit and 32 bit libraries have identical names, and are provided in separate directories to distinguish them.

The following sections describes the procedure for installing client libraries and getting GBS to recognize them.

Supported platforms

GBS version 7.1.2 is supported with GemStone/S 64 Bit version 2.2.3, on the following VisualWorks and OS platforms:

VisualWorks 7.5, with the 7.5 Object Engine, is supported on the following platforms:

- Window 2000, SP1 or later, and Windows XP, SP1 or later
- Solaris 2.9 and 2.10
- HP-UX 11.11
- SUSE Linux ES 10

VisualWorks 7.4.1, with the 7.4d Object Engine, is supported on the following platforms:

- Window 2000, SP1 or later, and Windows XP, SP1 or later
- Solaris 2.9 and 2.10
- HP-UX 11.11

VisualWorks 7.4, with the 7.4 Object Engine, is supported on the following platforms:

- Window 2000, SP1 or later, and Windows XP, SP1 or later
- Solaris 2.9 and 2.10
- HP-UX 11.11 (with the 7.4d Object Engine)

VisualWorks 5i.1 Envy, with the 5i.4c Object Engine, is supported on the following platforms:

- Window 2000, SP1 or later, and Windows XP, SP1 or later

GemStone/S 64 Bit 2.2.3 Windows Client Installation

To allow GemStone/S 64 Bit clients to connect and perform some operations on Windows, GemStone/S 64 Bit provides a separate Windows installation, with the name GemBuilder for C. This installation provides:

- The ability to run topaz on Windows, to login RPC to a GemStone/S 64 Bit server on UNIX.
- The client libraries to be used by GBS for RPC logins.

If you do not require topaz on Windows, you may wish to extract only the client libraries from the Windows installation, and move these files to a directory on the path or to the GBS installation. See “Installing the GemStone/S 64 Bit 2.2.3 Libraries onto Windows Clients” on page 6-3

To install the GemStone / S 64 Bit Windows Client product, use the following procedure. The procedure is similar regardless of what platform the GemStone / S 64 Bit server is running on.

1. The GemStone / S 64 Bit client installation for Windows is provided as a zipped archive with a name similar to `GemBuilderC2.2.3-x86.Windows_NT.zip`. Move this distribution file to the directory location in which this will be installed.
2. Create a directory named `GemBuilderC2.2.3-x86.Windows_NT`, into which the client will be installed, and make this directory the working directory. For example:

   ```
   C:\InstallDir> mkdir GemBuilderC2.2.3-x86.Windows_NT
   C:\InstallDir> cd GemBuilderC2.2.3-x86.Windows_NT
   ```

3. Unzip the distribution file using unzip. For example,

   ```
   C:\InstallDir\GemBuilderC2.2.3-x86.Windows_NT> unzip
   GemBuilderC2.2.3-x86.Windows_NT.zip
   ```

   In addition to several subdirectories, `C:\InstallDir\GemBuilderC2.2.3-x86.Windows_NT` also contains two text files: `PACKING.txt`, which lists all of the GemStone files, and `version.txt`, which identifies the product and release.
4. Add the directory

   `C:\InstallDir\GemBuilderC2.2.3-x86.Windows_NT\bin`

   to the machine search path.

Installing the GemStone/S 64 Bit 2.2.3 Libraries onto Windows Clients

This section describes the procedure for installing client libraries on Windows and getting GBS to recognize them.

1. Quit any running client Smalltalk VM that is using GBS.
2. Your client libraries may be placed in the directory in which the client Smalltalk executable resides, or in any directory that is on the path, %PATH%.

   NOTE

   In step 3., <GBS> refers to the directory in which the client Smalltalk executable resides, or any directory that is on the path that will be used by the GBS application, and <SRC> refers to the GemStone/S 64 Bit Windows client installation directory (for example, `installDir/GemBuilderC2.2.3-x86.Windows_NT/bin`).

3. To copy the files from the GemStone / S 64 Bit Windows client directory <*SRC*> to the GemBuilder installation directory, or another directory on the Windows path, <*GBS*>, execute the following:

   ```
   C:\> copy <SRC>\libgcirpc64-223.dll <GBS>
   C:\> copy <SRC>\libgcirpc.dll       <GBS>
   C:\> copy <SRC>\libgs64-223.dll     <GBS>
   ```

4. Ensure that there are no other copies of these libraries on the search path.

5. If the GBS configuration parameter *libraryName* is not set, or is set to `libgcirpc.dll`, installation is complete. The correct client library will be loaded when the client Smalltalk VM is restarted and the application logs in to GemStone.

   If you are converting from a different GemStone server product, your GBS application is likely to have set *libraryName* to the client library used with that product. This parameter needs to be updated for GemStone / S 64 Bit. If you are converting, continue with these instructions.

6. Start up the client Smalltalk VM.

7. Clear the library name. There are several ways to do this. Using the GBS settings dialog, select the tab titled “Server Communication”, and clear the field libraryName. You may also execute:

   ```
   GbsConfiguration current libraryName: ''
   ```

8. Save your image, exit, and restart, to allow the correct libraries to be loaded.

Installing the GemStone/S 64 Bit 2.2.3 Libraries onto Unix Clients

GemStone / S 64 Bit supports GBS clients running on Solaris, Linux and HP-UX with 32-bit VisualWorks. Since the 64-bit libraries shipped with GemStone / S 64 Bit cannot be loaded into a 32-bit application, GemStone / S 64 Bit includes 32-bit versions of these libraries. These libraries can be found in the lib32 subdirectory of the GemStone / S 64 Bit installation, `$GEMSTONE/lib32/`.

1. Log in to your GBS client machine as the user who is the owner of the GBS installation files.

2. Your client libraries may be placed in any directory that is on the machine search path `$LD_LIBRARY_PATH`, or you may add the directory containing the client libraries to the `$LD_LIBRARY_PATH`. Ensure that there are no other copies of the client libraries on the search path. (In the following instructions, the term “path” means the path specified by `$LD_LIBRARY_PATH`)

   Alternatively, you can place the shared libraries in any location, and in the GBS configuration parameter *libraryName*, specify the absolute path to the client library file.

   NOTE

   In step 3., <GBS> refers to your GemBuilder installation directory, or any directory that is on the path that will be used by the GBS application, and <SRC> refers to the lib32 subdirectory of your installation (for example, `installDir/GemStone64Bit2.2.3-sparc.Solaris/lib32`).

3. To copy the files from the GemStone / S 64 Bit server source directory <*SRC*> to the GemBuilder installation directory, or another directory, <*GBS*>, execute the following:

   % **cp** <*SRC*>**/libgcirpc64-223.so** <*GBS*>

   Note that the suffix on HP-UX is `.sl`, not `.so`; adjust the instructions for this platform.

4. Make the destination directory the current working directory, and create a symbolic link to the library:

   ```
   % cd <GBS>
   % ln -s libgcirpc64-223.so libgcirpc.so
   ```

5. If the client libraries are in a directory on the path, and the GBS configuration parameter *libraryName* is not set, or is set to `libgcirpc.so`, installation is complete. The correct client library will be loaded after the client Smalltalk VM is restarted, on login.

   If you are converting from a different GemStone server product, your GBS application is likely to have set *libraryName* to the client library used with that product. This parameter needs to be updated for GemStone / S 64 Bit. If you are converting, continue with these instructions.

6. Start up the client Smalltalk VM.

7. Set the library name. There are several ways to do this. Using the GBS settings dialog, select the tab titled “Server Communication”, and fill in the appropriate library name in the libraryName field. You may also execute:

   ```
   GbsConfiguration current libraryName: '<libraryName>'
   ```

   If the clientLibraries are in a directory on the path, clear the libraryName field, or set it to an empty string. The default client library search will find and load the correct library.

   If the client libraries are not in a directory on the path, *libraryName* must be set to the absolute path and name of the client library `libgcirpc.so`.

8. Save your image, exit, and restart, to allow the correct libraries to be loaded.