GemStone®

GemStone/S 64 Bit Installation Guide

for Solaris on Sun SPARC
or x86 _64 Compatible Systems

Version 3.1

July 2012

vmware®

GEMSTONE S™ 64

INTELLECTUAL PROPERTY OWNERSHIP

This documentation is furnished for informational use only and is subject to change without notice. VMware, 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 VMware, 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 VMware, 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 VMware, Inc.

This software is provided by VMware, 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 VMware, 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 software is covered by U.S. Patent Number 6,256,637 “Transactional virtual machine architecture”, Patent Number 6,360,219 “Object queues with concurrent updating”, Patent Number 6,567,905 “Generational garbage collector with persistent object cache”, and Patent Number 6,681,226 “Selective pessimistic locking for a concurrently updateable database”. GemStone software may also be covered by one or more pending United States patent applications.

TRADEMARKS

VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions.

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

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

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

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

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

Microsoft, **MS**, **Windows, Windows XP, Windows 2003**, **Windows 7** and **Windows Vista** 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.

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

AIX, **POWER5**, **POWER6**, and **POWER7** are trademarks or registered trademarks of International Business Machines Corporation.

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

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, VMware 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.

VMware, Inc.
15220 NW Greenbrier Parkway
Suite 150
Beaverton, OR 97006

Preface

About This Documentation

This document explains how to install GemStone/S 64 Bit version 3.1, and how to upgrade from previous GemStone/S 64 Bit versions. This document is also available on the GemStone Technical Support website.

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

Terminology Conventions

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

Technical Support

GemStone Website

http://support.gemstone.com

GemStone's Technical Support website provides a variety of resources to help you use GemStone products:

- **Documentation** for released versions of all GemStone products, in PDF form.
- **Downloads** and **Patches**, including past and current versions of GemBuilder for Smalltalk.
- **Bugnotes**, identifying performance issues or error conditions you should be aware of.
- **TechTips**, providing information and instructions that are not otherwise included in the documentation.
- **Compatibility matrices**, listing supported platforms for GemStone product versions.

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

Help Requests

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

Requests for technical support should be submitted online or by telephone. We recommend you use telephone contact only for serious requests that require immediate attention, such as a production system down. The support website is the preferred way to contact Technical Support.

Website: http://techsupport.gemstone.com

Email: techsupport@gemstone.com

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

When submitting a request, please include the following information:

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

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

24x7 Emergency Technical Support

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

Training and Consulting

Consulting is available to help you succeed with GemStone products. Training for GemStone software is available at your location, and training courses are offered periodically at our offices in Beaverton, Oregon. Contact your GemStone account representative for more details or to obtain consulting services.

Contents

Chapter 1

Installing GemStone/S 64 Bit Version 3.1

This chapter describes the procedure for installing GemStone/S 64 Bit version 3.1 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, follow the instructions in the appropriate later chapter of this Installation Guide.

Adjust the installation to meet your specific needs. The topic "What Next?" on page 19 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.

Platforms

- Sun SPARC server, one or more CPUs,
  or
  x86 64-bit system, one or more 2.6 GHZ CPUs.
- Multiple CPUs improve performance.

RAM

- At least 1 GB Physical RAM installed.

  1 GB is sufficient for only for very small systems. In most cases you will need much more, depending on the size of the cache and the number of Gem sessions.

Swap space

- Total swap space should be at least equal to the amount of RAM. We recommend installing twice as much swap space as RAM.

  Due to the way GemStone uses memory, systems with insufficient swap space allocated have a risk of memory errors even if there is available RAM.

Disk space

- Space for the installed distribution files—you need approximately 450 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.

Operating system

- Solaris 10 for SPARC
  or
  Solaris 10 for x86
- For this release of GemStone/S 64 Bit, the following Solaris patches were installed for the specified Operating System and Platform:

  Solaris 10 on SPARC:
  No patches are required

  Solaris 10 on x86:
  kernel version Generic_127128-11 or later, with required patches
- 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 (nodes that will run Gem processes).

The following suggested settings in `/etc/system` are typical of those used at GemStone, and most are intentionally set high to provide flexibility. These settings may not be suitable for your hardware configuration and system load.

On Solaris 10, Sun recommends configuring shared memory and semaphores at the project level, rather than system wide. This provides greater flexibility, as well as avoiding the need to reboot after editing `/etc/system`. However, if the parameter settings are present in `/etc/system`, they are used as the default project settings. If you are not partitioning your system using projects, you can continue to configure these settings using `/etc/system`.

a. Shared memory maximum

   In earlier versions of Solaris, `shmsys:shminfo_shmmax` set the maximum shared memory segment size. The default for maximum shared memory segment is not adequate to run GemStone/S 64 Bit; we recommend a value larger than your desired Shared Page Cache size, and not more than 75% of your real memory.

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

   8192 MB * .75 = 6144 MB
   6144 MB * 2**20 = 6442450944 bytes

   To set this parameter using legacy-style Solaris configuration, in `/etc/system` include the following:

   ```
   set shmsys:shminfo_shmmax=6442450944
   ```

   The equivalent parameter in Solaris 10 is `project.max-shm-memory`, which controls project total memory use rather than segment size. You may configure this using the `prctl` command on your project. For example, to limit memory to 6 GB:

   ```
   prctl -n project.max-shm-memory -v 6gb -r -i project "user.root"
   ```

b. Number of semaphore identifiers

   In earlier versions of Solaris, `semsys:seminfo_semmni` set the number of semaphore identifiers in the system. This parameter limits the number of GemStone shared page caches on the node because each shared page cache uses one identifier. This parameter must be large enough to allow for one shared page cache for each Stone running on the node, plus one for each cache running on a remote node.

   The equivalent project parameter in Solaris 10 is `project.max-sem-ids`. The default is 128. This should be adequate for most installations.

c. Maximum number of semaphores per id

   In earlier versions of Solaris, `semsys:seminfo_semmsl` set 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, this parameter must provide **two** semaphores for each user who will log in to that Stone from any node plus an overhead of **four**. In distributed systems, nodes that have only user sessions (Gems), but no Stones, must provide **two** semaphores for each user session on that node plus an overhead of **one**.

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

For example, to set this parameter to 500 using legacy-style Solaris configuration, in `/etc/system` include the following:

```
set semsys:seminfo_semmsl=500
```

In Solaris 10, the equivalent project parameter is `process.max-sem-nsems`. The default is 512 which is adequate for smaller-sized systems. For larger systems, use a command similar to this example, which sets the value to 1024:

```
prctl -n project.max-sem-nsems -v 1024 -r -i project "user.root"
```

d. If you are adjusting settings using `/etc/system`, save the file, and reboot the system using **`boot -r`**.

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

SPARC:

- Solaris Studio 11
  CC: Sun C++ 5.8 Patch 121017-05 2006/08/30

x86:

- Solaris Studio 12 update 1
  CC: Sun C++ 5.10 SunOS_i386 128229-09 2010/06/24

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

SPARC:

- Sun Dbx Debugger 7.5 Patch 121023-02 2006/05/26

x86:

- Sun Dbx Debugger 7.7 SunOS_i386 2009/06/03

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.

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.

These are the portions of the system that are affected by the installation of GemStone:

`/dev/rdsk`
: Optional raw partitions for repository extents and transaction logs.

`/etc/services`
: Internet services database, for NetLDI name lookup.

/InstallDir/`GemStone64Bit3.1.0-sparc.Solaris`
/InstallDir/`GemStone64Bit3.1.0-i386.Solaris`
: Location of the object server software.

`/opt/gemstone`
: Default location for server lock files, host name id file, and log files for GemStone network servers (NetLDIs). See the *System Administration Guide* for more information.

`/tmp/gemstone`
: Pipe file for the Stone repository monitor.

`/usr/gemstone`
: Alternative location for lock and log files, for compatibility with previous products; `/opt/gemstone` is created unless `/usr/gemstone` already exists. See the *System Administration Guide* for more information.

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.

2. Determine that adequate swap space is available:

   ```
   % /usr/sbin/swap -s
   ```

3. 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.

4. Select an installation directory, *InstallDir*, and make this directory the current working directory.
5. GemStone/S 64 Bit is provided as a zipped archive file with a name similar to `GemStone64Bit3.1.0-sparc.Solaris.zip` or `GemStone64Bit3.1.0-i386.Solaris.zip`.
6. Move this distribution file to the directory location in which GemStone will be installed, *InstallDir*.
7. Unzip the distribution file using unzip. For example:

   ```
   % unzip GemStone64Bit3.1.0-sparc.Solaris.zip
   ```

8. The *InstallDir* now contains a GemStone directory with a name similar to `GemStone64Bit3.1.0-sparc.Solaris` or `GemStone64Bit3.1.0-i386.Solaris`.

   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.
9. 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/GemStone64Bit3.1.0-sparc.Solaris
```

or

```
% setenv GEMSTONE InstallDir/GemStone64Bit3.1.0-i386.Solaris
```

Bourne or Korn shell:

```
$ GEMSTONE=InstallDir/GemStone64Bit3.1.0-sparc.Solaris
$ export GEMSTONE
```

or

```
$ GEMSTONE=InstallDir/GemStone64Bit3.1.0-i386.Solaris
$ export GEMSTONE
```

Create the GemStone Key File

To run GemStone, you must have a key file. Keyfile information is provided by GemStone Contract Administration. Contact your GemStone account representative or keyfiles@gemstone.com if you have questions about your keyfile, or if you need an evaluation keyfile.

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

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

2. Using a text editor, create the key file, pasting in the provided keyfile information:

   `$GEMSTONE/sys/gemstone.key.`

3. 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 (each **1** is the number 1):

   ```
   % /usr/sbin/ping hostname 1 1
   ```

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

Define the NetLDI Service

The NetLDI service, by default `gs64ldi`, should be defined in your TCP/IP network database. This is required for certain kinds of local sessions, or any client on a remote machine, to log into GemStone. For clients on remote machines, the same NetLDI service name and port number must be defined on the remote machine.

NOTE

If you are upgrading from a previous version, you might need to keep the NetLDI for that version running. You may need to select a distinct name and port for the NetLDI for GemStone/S 64 Bit 3.1, rather than using the default `gs64ldi`.

1. 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.

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

2. Determine whether the `gs64ldi` service is already defined. How to do this will depend on how your system is set up.

   - If you are using NIS:
     ```
     % ypcat services | grep gs64ldi
     ```
   - If you are using ldap:
     ```
     % getent services | grep gs64ldi
     ```
   - If your machine is using a local copy of the TCP/IP network database:
     ```
     % grep gs64ldi /etc/services
     ```

   If `gs64ldi` is defined, skip the rest of this procedure and continue with the installation at “Run the Installation Script” on page 15.

   If it is not defined, continue performing this procedure.

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

   ```
   gs64ldi 50377/tcp #GemStone/S 64 Bit 3.1
   ```

   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).

4. If NIS is running, propagate the change to the network database to the rest of the network.
5. 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.

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 the environment variable GEMSTONE_GLOBAL_DIR is defined to point to a valid directory, this overrides the default server lock files and log file location; however, all Gemstone processes that will interact on this machine must have this environment variable set to the same directory.

If these directories do not exist, the installation script offers to create `/opt/gemstone` and the subdirectories `locks` and `log`. Then, the script offers to set access (770) 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

Log out as user root. The rest of the installation is done as the GemStone administrative user.

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 chapter entitled User Accounts and Security in the *System Administration Guide for GemStone/S 64 Bit* tells you how to change the passwords and set up accounts for other GemStone users, and how to create new GemStone user accounts. These functions can also be done using GemBuilder for Smalltalk tools; see the *GemBuilder for Smalltalk Users's Guide* for more information.

Users must 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 3.1 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 3.1 directory.

   C shell:

   ```
   % setenv GEMSTONE InstallDir/GemStone64Bit3.1.0-sparc.Solaris
   ```

   or:

   ```
   % setenv GEMSTONE InstallDir/GemStone64Bit3.1.0-i386.Solaris
   ```

   Bourne or Korn shell:

   ```
   $ GEMSTONE=InstallDir/GemStone64Bit3.1.0-sparc.Solaris
   $ export GEMSTONE
   ```

   or:

   ```
   $ GEMSTONE=InstallDir/GemStone64Bit3.1.0-i386.Solaris
   $ export GEMSTONE
   ```

2. Invoke the script gemsetup.

   C shell:

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

   Bourne or Korn shell:

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

3. 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.

   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 3.1 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 3.x versions

This chapter describes how to upgrade an existing GemStone/S 64 Bit 3.0.x installation to GemStone/S 64 Bit version 3.1. GemStone/S 64 Bit version 3.1 supports upgrade from GemStone/S 64 Bit versions 3.0 and 3.0.1. For upgrading from GemStone/S 64 Bit 2.x versions, which require conversion, see Chapter 3 on page 27 or Chapter 5 on page 41.

If you are using GemBuilder for Smalltalk (GBS), you must also upgrade your GBS version and the client libraries that are used by GBS. Older versions of GBS cannot log into v3.1 due to changes in shared library naming and requirements. See Chapter 6 for supported versions of GBS for use with GemStone/S 64 Bit 3.1, and instructions on installing updated client libraries.

For applications using GemConnect or GemBuilder for Java, these products will need to be reinstalled following the upgrade process.

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 production system running while you familiarize yourself with the upgrade process.

Upgrade Procedure

The following list summarizes the steps necessary to perform the upgrade to GemStone/S 64 Bit version 3.1.

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 3.1 kernel methods to determine whether your changes are still necessary or appropriate.

2. Install GemStone/S 64 Bit 3.1 to a new installation directory, separate from the installation directory for version 3.0.1, as described in Chapter 1 of this Installation Guide.

3. Configure GemStone/S 64 Bit 3.1 the way you expect to use it — that is, with the appropriate extent locations and sizes.

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

   - Your version 3.0.1 extents and transaction logs.
   - Your version 3.1 extents and transaction logs.

5. Log in to the version 3.0.1 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 3.1. 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 3.0.1 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 stone301
   ```

   where *stone301* is the name of the version 3.0.1 stone on this machine. The repository must be cleanly shut down before it can be restarted under version 3.1

9. Set up the version 3.1 environment.

   Set the environment variables required for the upgrade.

   C shell:

   ```
   % setenv GEMSTONE InstallDir31
   % set path = ($GEMSTONE/bin $path)
   % setenv upgradeLogDir tempDir
   ```

   Bourne or Korn shell:

   ```
   $ GEMSTONE=InstallDir31
   $ export GEMSTONE
   $ export PATH=$GEMSTONE/bin:$PATH
   $ upgradeLogDir=tempDir
   $ export upgradeLogDir
   ```

   where *InstallDir31* is the GemStone/S 64 Bit version 3.1 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 3.0.1 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 3.0.1 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 3.1 installation. For example:

    ```
    % cp InstallDir301/data/extent0.dbf 31location
    % cp InstallDir301/data/extent1.dbf 31location
    % cp InstallDir301/data/extent2.dbf 31location
    ```

    where *31location* is the location specified by `DBF_EXTENT_NAMES` in the configuration file that will be used in version 3.1.

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 3.1. Transaction logs from earlier versions are not compatible with version 3.1. If the transaction log directories will be reused for version 3.1, any transaction logs should be deleted or copied elsewhere.

Perform the Upgrade

1. Start the 3.1 Stone on the 3.0.1 extents you just copied:

   ```
   % startstone stoneName31
   ```

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` *stoneName31*

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.

4. The format for class comments has changed between versions 3.0.x and v3.1. Now, comments are stored as Strings in a new field in the class, rather than as class methods or as instances of GsClassDocumentation in the description field.

   Kernel class comments are upgraded automatically. To upgrade application class comments, use the script `upgradeComments`.

   You must upgrade comments to the new scheme. In particular, you should not have class methods named #comment on any of your classes. The `upgradeComments` script both sets the comment and deletes the class method #comment.

   For example,

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

   For example,

   `% upgradeComments -s` *stoneName31*

   The script should complete with the message:

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

Seaside Upgrade

If you are using Seaside/GLASS, you will need to perform another step to upgrade your Seaside image. This step upgrades seaside, and reloads your Mettachello code from your respository.

For details, see Chapter 4, starting on page 37. When you have completed the Seaside upgrade, continue with the upgrade process and perform the following steps.

Restore Your Site-Specific Settings

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

   If you use GemConnect or GemBuilder for Java, you must reinstall the appropriate versionof these products into your repository at this time. To install, use the procedure in the installation guide for the specific product.

2. Log in to GemStone/S 64 Bit version 3.1 as DataCurator, and change the password for SystemUser, which you changed to `swordfish` prior to the conversion, back to its previous value. Also, change the passwords for GcUser and DataCurator, which were reset by the conversion process, back to the version 3.0.1 value:

   ```
   topaz 1> printit
   (AllUsers userWithId: 'SystemUser') password: '301Password'.
   (AllUsers userWithId: 'GcUser') password: '301Password'.
   (AllUsers userWithId: 'DataCurator') password: '301Password'.
   System commitTransaction
   %
   ```

   where *301Password* is the account password used in GemStone/S 64 Bit version 3.0.1.

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

3. If you have modified any kernel class methods of the previous version, carefully compare your changes with version 3.1 kernel methods to see whether your changes are still necessary or appropriate. If so, file in the changes and commit.

Backup repository and configure GBS

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

2. If you are using GBS clients, ensure you have upgraded to a supported version. You must use GBS version 7.5 or later to connect to a GemStone/S 64 Bit v3.1 repository.

   Configure GBS to use the version 3.1 client libraries. Note that the required libraries, library naming conventions, and the process GBS uses to identify the correct library to load have changed in version 7.5. Read the updated instructions carefully.

   See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details.

Chapter 3

Converting from GemStone/S 64 Bit 2.4.x versions

This chapter describes how to upgrade an existing GemStone/S 64 Bit 2.4.x installation to GemStone/S 64 Bit version 3.1.

GemStone/S 64 Bit version 3.1 supports upgrade from GemStone/S 64 Bit versions 2.4.4 and later only, and from version 2.2.6. To upgrade from version 3.0.x, see Chapter 2 on page 21; for upgrade from version 2.2.6, see Chapter 5 on page 41.

If you are upgrading from a different version of GemStone/S 64 Bit, or from 32-bit GemStone/S, you will first need to upgrade to version 2.4.4 or later, following the instructions in the Installation Guide for v2.4.

Between versions 2.x and 3.x, compiled methods have changed and version 3.x has new bytecodes to support native code. As a result, the conversion process requires that you file in all application source code so it can be recompiled. Alternatively, you may iterate and recompile all methods in your application individually. There are a number of changes you may have to make to application source code and to persistent instances; please review the upgrade process carefully.

New keyfiles are required with GemStone/S 64 Bit version 3.1. Keyfiles for GemStone/S 64 Bit 2.x will not work with version 3.x. If you need a keyfile to evaluate version 3.1, contact GemStone Technical Support, or email keyfiles@gemstone.com.

If you are using GemBuilder for Smalltalk (GBS), you must also upgrade GBS to version 7.5, and update the client libraries that are used by GBS. Older versions of GBS cannot log into 3.1 due to changes in the shared library naming and requirements. See Chapter 6 on page 49 for using GBS with GemStone/S 64 Bit 3.1, and instructions on installing updated client libraries.

For applications using GemConnect or GemBuilder for Java, updated versions of these products will also need to be reinstalled following the upgrade process.

Upgrade Strategy

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

We recommend that as part of validating your code changes, after making the required changes in your code, you file your application code and kernel class changes into an empty 3.1 installation, prior to performing the full upgrade. This will allow you to test your code changes in the 3.1 environment before undergoing the full application upgrade.

After verifying your application code changes, you should perform a pilot upgrade of your application, including the required modifications to application objects. This will allow you to uncover any issues prior to converting your production system.

Upgrade Procedure

The following list summarizes the steps necessary to perform the upgrade to GemStone/S 64 Bit version 3.1.

NOTE

The following instructions use the version number 2.4.5.1 to represent any version from which upgrade/conversion is supported. The procedure is the same regardless of which version you have.

Prior to Upgrade in existing application

1. If you do not already have source code for your application stored externally to the GemStone repository in a code management system, file out all application source code using the Topaz command **fileout**. Filein of application code is required to recompile methods.

   You should confirm that the format of your filed out code does not create new versions of your application classes on filein.

2. File out any modifications or additions you made to GemStone/S 64 Bit kernel class methods. You will need to carefully compare these changes with GemStone/S 64 Bit 3.1 kernel methods to determine whether your changes are appropriate. GemStone kernel classes have extensive changes for version 3.0, as well as changes in v.3.1. See the Release Notes for version 3.0 for details of the changes, and read these carefully before filing 2.x code modifications into your version 3.1 repository.

   You should also review the changes documented in the 3.0.1 and 3.1 Release Notes, as there are further, though less sweeping, changes in these releases.

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

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

4. If you have indexes that include instance of any of the following classes, these indexes will need to be removed and rebuilt in version 3.1 to update information within the indexing structures.

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

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

5. The Class description variable is used in v2.x to hold class comments. If you have defined class comments using GBS browsers, depending on the version of GBS, class comments may be stored in the description variable or as class method named comment that returns a string containing the comment text.

   If there is comment information in the description field, you will need to save the contents elsewhere. Due to structural changes in classes, the conversion process will remove any data stored here. Following upgrade, you can restore the class comment, which will then be upgraded for the new handling of comments in v3.1.

Prepare for Upgrade

Perform the following steps to prepare for the upgrade.

1. Install GemStone/S 64 Bit 3.1 to a new installation directory, separate from the installation directory for version 2.4.5.1, as described in Chapter 1 of this Installation Guide.
2. Configure GemStone/S 64 Bit 3.1 the way you expect to use it — that is, with the appropriate extent locations and sizes.
3. Ensure that adequate space is available for extents, transaction logs, and a backup during the upgrade. You must provide space for the following:
   - Your version 2.4.5.1 extents and transaction logs.
   - Your version 3.1 extents and transaction logs.
4. Log in to the version 2.4.5.1 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.

5. 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.
%
```

6. You may now shut down the Stone:

   % **stopstone** *stone2451*

   where *stone2451* is the name of the version 2.4.5.1 stone on this machine. It is strongly reocmmended, but not required, that the repository be cleanly shut down before it is restarted under version 3.1. If the repository is not cleanly shut down, you must restart using the -N option.

7. Set up the version 3.1 environment.

   Set the environment variables required for the upgrade.

   C shell:
   % **setenv GEMSTONE** *InstallDir31*
   % **set path = ($GEMSTONE/bin $path)**
   % **setenv upgradeLogDir** *tempDir*

   Bourne or Korn shell:
   $ **GEMSTONE=***InstallDir31*
   $ **export GEMSTONE**
   $ **export PATH=$GEMSTONE/bin:$PATH**
   $ **upgradeLogDir=***tempDir*
   $ **export upgradeLogDir**

   where *InstallDir31* is the GemStone/S 64 Bit version 3.1 installation and *tempDir* is a temporary directory for which you have write permission.

8. Copy your version 2.4.5.1 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 2.4.5.1 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 3.1 installation. For example:

   % **cp** *InstallDir2451***/data/extent0.dbf** *31location*
   % **cp** *InstallDir2451***/data/extent1.dbf** *31location*
   % **cp** *InstallDir2451***/data/extent2.dbf** *31location*

   where *31location* is the location specified by `DBF_EXTENT_NAMES` in the configuration file that will be used in version 3.1.

   If version 3.1 will run on a platform with a different byte ordering than the version 2.4.5.1 repository, or if you are using raw partitions, you will need to use copydbf.

9. 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 3.1. If the transaction log directories will be reused for version 3.1, any transaction logs should be deleted or copied elsewhere.

Perform the Upgrade

1. Conversion is done using the -C flag to startstone. Perform the conversion on the 2.4.5.1 extents you just copied:

   % **startstone** -C *stoneName31*

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

   For example,

   % **upgradeImage** -s *stoneName31*

   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.

Seaside Upgrade

If you are using Seaside/GLASS, you will need to perform another step to upgrade your Seaside image. This step upgrades seaside, and reloads your Mettachello code from your respository.

For details, see Chapter 4, starting on page 37. When you have completed the Seaside upgrade, continue with the upgrade process and perform the following steps.

Restore Your Site-Specific Settings

Log in to GemStone/S 64 Bit version 3.1 as DataCurator, and change the password for SystemUser, which you changed to `swordfish` prior to the conversion, back to its previous value. Also, change the passwords for GcUser and DataCurator, which were reset by the conversion process, back to the version 2.4.5.1 value:

```
topaz 1> printit
(AllUsers userWithId: 'SystemUser') password: '2451Password'.
(AllUsers userWithId: 'GcUser') password: '2451Password'.
(AllUsers userWithId: 'DataCurator') password: '2451Password'.
System commitTransaction
%
```

where *2451Password* is the account password used in GemStone/S 64 Bit version 2.4.5.1.

Post-upgrade Application Code Modifications

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

   If you use GemConnect or GemBuilder for Java, you must reinstall the appropriate version into your repository at this time. Version 3.1 requires new versions of these products. Contact GemStone Technical Support for more information. To install, use the procedure in the installation guide for the specific product.

2. Recompile application code by filein

   The upgrade process requires filein of all customer code, in order to recompile to the new bytecodes and classes introduced in version 3.0. You should have fileins available, from step 1 on page 28.

   Seaside images will not need to perform this step.

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

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

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

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

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

3. Convert persistent blocks

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

a. Application-specific stored blocks

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

b. SortBlocks in SortedCollections

To convert the sortBlocks in persistent SortedCollections in your application, you may run the `postconv` script. This script has optional flags to specify stone name and the number of sessions to use.

`postconv [-c <numCacheWamerGems>][-h][-s <stoneName>] [-r] [-n <numberOfSessions>] [-t <tempObjCacheSize>] [-u <userId>]`

`-c <numCacheWamerGems>` specifies the number of cache warmer threads in a single gem to load the object table into the shared cache before starting post-conversion. If not specified, no cache warming is done.
`-h` prints this usage information.
`-s <stoneName>` sets the name of the running stone to scan; if this option is not used, the script uses **gs64stone.**
`-n <numberOfSessions>` specifies the number of parallel sessions which will convert the instances of SortedCollection and its subclasses. By default, use one session.
`-r` specifies to reuse an existing version of `$upgradeLogDir/AllSortedCollections.bms`, if it exists. This file contains the OOPs of all instances of SortedCollections and its subclasses. By default, the existing file is deleted and a new one created.
`-t <tempObjCacheSize>` sets the size of GEM_TEMPOBJ_CACHE_SIZE in KB; by default, 20000.
`-u <userId>` is the UserId whose SymbolList includes all subclasses of SortedCollection, for which instance's sortBlocks will be converted. If not specified, defaults to SystemUser

For example,

`% postconv -s` *stoneName31*

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

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

4. Restore and upgrade Class comments.

In v3.1, the class comment is stored in the class as a String, and accessed via #comment and #comment: methods. This differs from the comment handling in 2.x or 3.0.x.

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

Since Seaside handles class comments in a way that is consistent with new 3.1 behavior, Seaside images do not need to perform this step.

First, restore Class descriptions that were saved from your version 2.4.5.1 repository (as per step 5 on page 29). Descriptions are restored using the description: method.

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

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

For example,

`% upgradeComments -s stoneName31`

The script should complete with the message:

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

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

5. Convert reimplemented classes

   Optionally, you may wish to convert instances of reimplemented classes. Instances of Float, SmallFloat, LargePositiveInteger, LargeNegativeInteger, DoubleByteString, DoubleByteSymbol or QuadByteString in your version 2.4.5.1 repository are not converted during conversion to 3.x. The instances auto-migrated to the new class each time they are faulted into the VM.

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

6. Convert FixedPoint instances

   If for step 3 on page 28, you decided to continue to use the old ScaledDecimal class, now named FixedPoint, do not perform this step.

   If your 2.4.5.1 application included instances of ScaledDecimal, following conversion to 3.x they are now instances of FixedPoint. If you intend to use the new ScaledDecimal implementation rather than FixedPoint, you must perform a manual step to convert each instance to a new ScaledDecimal. To do this, find all instances of FixedPoint and execute code similar to the following:

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

   Verify that errorcount is 0, and commit.

7. Rebuild indexes, if necessary.

   If your application includes indexes on instances of kernel classes which have new implementations in version 3.x, these indexes must be rebuilt to update internal information in the Btree indexing structure, otherwise query results may be incorrect.

   See step 4 on page 29 for more details on this requirement.

8. If you have modified any kernel class methods of the previous version or if you have added methods to kernel classes, carefully compare your changes with the changes in version 3.1 to see whether your changes are still necessary or appropriate. Carefully review the Release Notes for v3.0 and v3.1, as well as examining code in the image. If you changes are still applicable, file in the changes and commit.

Backup repository and configure GBS

1. Create a full backup of the upgraded repository. For details, see the *System Administration Guide for GemStone/S 64 Bit*.
2. If you are using GBS clients, ensure you have upgraded to a supported version. You must use GBS version 7.5 or later to connect to a GemStone/S 64 Bit v3.1 repository.

   Configure GBS to use the version 3.1 client libraries. Note that the required libraries, library naming conventions, and the process GBS uses to identify the correct library to load have changed in version 7.5. Read the updated instructions carefully.

   See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details.

Chapter 4

Upgrading Seaside/GLASS Applications

This chapter describes the additional upgrade step that applies when upgrading a Seaside/GLASS application that is based on GemStone/S 64 Bit 2.4.x or 3.0.x to GemStone/S 64 Bit version 3.1.

The complete upgrade process is described in the relevant chapter of this Installation Guide; Chapter 3 for version 2.4.x and Chapter 2 for upgrade from 3.0.x. You will need to follow the steps in the relevant chapter, which will note the point at which the Seaside upgrade takes place.

Due to the differences between Seaside and GemStone, not all steps of the upgrade process apply. The upgrade instructions note the differences.

Prepare for upgrading Seaside/GLASS applications

Before Seaside Upgrade

After you have completed the GemStone/S 64 Bit image upgrade, as described in Chapter 2 or Chapter 3, you can upgrade your Seaside application.

Do not perform this upgrade if you have not already completed the upgrade process through the `upgradeImage` step.

You should also have confirmed that your application code has been updated as required. This is particularly important when upgrading from version 2.4.x. The GemStone/S 64 Bit v3.0 release included extensive changes that impact most application code.

Configure Upgrade

Before upgrading your Seaside application, you will need to configure your environment to correctly find and load your application code. The following variables in the UserGlobals of DataCurator should be reviewed and set as necessary:

`#BootstrapSymbolDictionary`
The symbolDictionary in which the root of your application-specific Seaside classes are located. By default, UserGlobals.

`#BootstrapSymbolDictionaryName`
The name of the symbolDictionary in which the root of your application-specific Seaside classes are located. By default, the name of what is set for #BootstrapSymbolDictionary

`#BootstrapRepositoryDirectory`
The directory in which your base GemStone classes exist. By default, `GsPackageLibrary getMonticelloRepositoryDirectory`. This normally resolves to `$GEMSTONE/seaside/monticello/repository`.

`#BootstrapApplicationLoadSpecs`
A collection of 4-element arrays. Each array includes:

- the name of a configuration
- the version of the configuration
- an array of names of what to load from the configuration
- the monticello directory.

The minimum required, and the default, is:

```
{ {'ConfigurationOfGLASS' . '1.0-beta.8.7.2' . #('default') .
BootstrapRepositoryDirectory } }
```

`#BootstrapExistingConfigurationList`
A collection of configurations in your 2.x or 3.x repository, that will be deleted and reloaded from the repository. By default, everything in the #BootstrapSymbolDictionary whose key begins with 'ConfigurationOf'. If you name your configurations differently, you will need to customize this list.

`#BootstrapApplicationPostloadClassList`
The set of classes that should be initialized after the reload. If the class is on this list, it will be sent #initialize after load, which may cause data loss. By default, an empty collection.

Example Seaside Upgrade Script

The following script is an example of an seaside upgrade customization. In this example, `$MYREPOS/monticello` is the path of the application specific monticello repository.

Example 4.1 Example Seaside upgrade setup script

```
set user DataCurator pass swordfish
login
run
UserGlobals
  at: #BootstrapRepositoryDirectory
  put: GsPackageLibrary getMonticelloRepositoryDirectory.
UserGlobals
  at: #BootstrapApplicationPostloadClassList
  put: #( #MBConfigurationRoot).
true
%
run
UserGlobals
  at: #BootstrapApplicationLoadSpecs
  ifAbsent: [
     UserGlobals
        at: #BootstrapApplicationLoadSpecs
        put: {
           { 'ConfigurationOfGLASS' . '1.0-beta.8.7.2' . #('default') .
                 BootstrapRepositoryDirectory } .
           { 'ConfigurationOfSeaside30'. '3.0.7'. #( 'Base' 'JQuery-UI'
                 'Seaside-Adaptors-Swazoo' ) .
                 '$MYREPOS/monticello'      } .
           { 'ConfigurationOfSqueakSource' . '3.0-beta.2' . #('All').
                 '$MYREPOS/monticello'      } .
           { 'ConfigurationOfMetacelloBrowser' . '1.61' . #('default') .
                 '$MYREPOS/monticello'      } .
           { 'ConfigurationOftODE' . '0.1' . #('default') .
                 '$MYREPOS/monticello'      }.
        }.
     ].
true
%
commit
logout
```

Perform the Upgrade

The Seaside upgrade is performed by the script `upgradeSeasideImage`, which is located in the `$GEMSTONE/seaside/bin` subdirectory .

Prior to executing `upgradeSeasideImage`, you should have up Global variables to customize the upgrade.

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

For example,

```
% $GEMSTONE/seaside/bin/upgradeSeasideImage -s stoneName31
```

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

Complete the Upgrade Process

Return to the appropriate chapter describing the upgrade process and follow the remaining steps.

Chapter 5

Converting from GemStone/S 64 Bit 2.2.6

This chapter describes how to upgrade an existing GemStone/S 64 Bit 2.2.6 installation to GemStone/S 64 Bit version 3.1. If you are upgrading from version 3.0.x or from 2.4.x,see Chapter 2 on page 21 or Chapter 3 on page 27.

Between versions 2.x and 3.x, compiled methods have changed and version 3.x has new bytecodes to support native code. As a result, the conversion process requires that you file in all application source code so it can be recompiled. There are a number of changes you may have to make to application source code and to persistent instances; please review the upgrade process carefully.

New keyfiles are required with GemStone/S 64 Bit version 3.1. Keyfiles for GemStone/S 64 Bit 2.x will not work with version 3.x. If you need a keyfile to evaluate version 3.1, contact GemStone Technical Support, or email keyfiles@gemstone.com.

If you are using GemBuilder for Smalltalk (GBS), you must also upgrade GBS to version 7.5 or later, and update the client libraries that are used by GBS. Older versions of GBS cannot log into 3.1 due to changes in the shared library naming and requirements. See Chapter 6 on page 49 for using GBS with GemStone/S 64 Bit 3.1, and instructions on installing updated client libraries.

For applications using GemConnect or GemBuilder for Java, updated versions of these products will also need to be reinstalled following the upgrade process.

Upgrade Strategy

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

We recommend that as part of validating your code changes, after making the required changes in your code, you file your application code and kernel class changes into an empty 3.1 installation, prior to performing the full upgrade. This will allow you to test your code changes in the 3.1 environment before undergoing the full application upgrade.

After verifying your application code changes, you should perform a pilot upgrade of your application, including the required modifications to application objects. This will allow you to uncover any issues prior to converting your production system.

Upgrade Procedure

The following list summarizes the steps necessary to perform the upgrade to GemStone/S 64 Bit version 3.1.

Prior to Upgrade in existing application

1. If you do not already have source code for your application stored externally to the GemStone repository in a code management system, file out all application source code using the Topaz command **fileout**. Filein of application code is required to recompile methods.

   You should confirm that the format of your filed out code does not create new versions of the application classes on filein.

2. File out any modifications or additions you made to GemStone/S 64 Bit kernel class methods. You will need to carefully compare these changes with GemStone/S 64 Bit 3.1 kernel methods to determine whether your changes are appropriate. GemStone kernel classes have extensive changes for version 3.0, as well as changes in 3.1; and consider changes in version 2.3 and 2.4. Review the Release Notes for these major versions carefully before filing 2.x code modifications into your version 3.1 repository.

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

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

4. You must remove all indexes in your application. There are changes in the indexing internal structures to accommodate QuadByteStrings, introduced in v2.3, and these structures for existing indexes are not modified by upgrade.

5. The Class description variable is used in v2.x to hold class comments. If you have defined class comments using GBS browsers, depending on the version of GBS, class comments may be stored in the description variable or as class method named comment that returns a string containing the comment text.

   If there is comment information in the description field, you will need to save the contents elsewhere. Due to structural changes in classes, the conversion process will remove any data stored here. Following upgrade, you can restore the class comment, which will then be upgraded for the new handling of comments in v3.1.

Install 3.1 and set up environment for preprocessing

1. Install GemStone/S 64 Bit 3.1 to a new installation directory, separate from the installation directory for version 2.2.6, as described in Chapter 1 of this Installation Guide.
2. Configure GemStone/S 64 Bit 3.1 the way you expect to use it — that is, with the appropriate extent locations and sizes.
3. Ensure that adequate space is available for extents, transaction logs, and a backup during the upgrade. You must provide space for the following:
   - Your version 2.2.6 extents and transaction logs.
   - Your version 3.1 extents and transaction logs.
4. Set up environment variable for preprocessing

   Define a new environment variable, `$GEMSTONE_3X`, to point to the GemStone/S 64 Bit 3.1product tree. At this point, do not reset `$GEMSTONE` or update the path to point to v3.1.

   For example, using the default GemStone/S 64 Bit 3.1 installation location from Chapter 1:

   C shell:

   ```
   % setenv GEMSTONE_3X <InstallDir>/GemStone64Bit-3.1<platform>
   ```

   Bourne or Korn shell:

   ```
   $ GEMSTONE_3X=<InstallDir>/GemStone64Bit-3.1<platform>
   $ export GEMSTONE_3X
   ```
5. 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.
6. Log in to the version 2.2.6 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.

7. 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.
   %
   ```

8. Run the GemStone/S 64 Bit 3.1 script **convprep22**. This script is in the `bin` directory of the 3.1 product tree; this should not be in your path, so it will need to be invoked with an explicit path. For example,

   % **$GEMSTONE_3X/bin/convprep22 -s** *stoneName226*

   This script has the following options:

   `convprep22 [-c <tempObjCacheSize>] [-s <stonename>]`
   `-c <tempObjCacheSize>` sets the size of temp obj cache to use, in KB. Default: 100000.
   `-s <stoneName>` sets the name of the running v2.2.6 Stone to scan; if this option is not used, the script will default to **gs64stone.**

   When it completes, it will report:

   `Conversion preparation complete. No errors detected.`

   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 Upgrade

1. Set the GemStone environment variables.

   C shell:
   % **setenv GEMSTONE** *InstallDir31*
   % **set path = ($GEMSTONE/bin $path)**

   Bourne or Korn shell:
   $ **GEMSTONE=***InstallDir31*
   $ **export GEMSTONE**
   $ **export PATH=$GEMSTONE/bin:$PATH**

   where *InstallDir31* is the GemStone/S 64 Bit version 3.1 installation.

2. Copy your version 2.2.6 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 2.2.6 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 3.1 installation. For example:

```
% cp InstallDir226/data/extent0.dbf 31location
% cp InstallDir226/data/extent1.dbf 31location
% cp InstallDir226/data/extent2.dbf 31location
```

where *31location* is the location specified by DBF_EXTENT_NAMES in the configuration file that will be used in version 3.1.

If version 3.1 will run on a platform with a different byte ordering than the version 2.2.6 repository, or if you are using raw partitions, you will need to use copydbf.

3. 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 3.1. If the transaction log directories will be reused for version 3.1, any transaction logs should be deleted or copied elsewhere.

Perform the Upgrade

1. Conversion is done using the -C flag to startstone. Perform the conversion on the 2.2.6 extents you just copied:

```
% startstone -C stoneName31
```

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 stoneName31
```

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

Log in to GemStone/S 64 Bit version 3.1 as DataCurator, and change the password for SystemUser, which you changed to `swordfish` prior to the conversion, back to its previous value. Also, change the passwords for GcUser and DataCurator, which were reset by the conversion process, back to the version 2.2.6 value:

```
topaz 1> printit
(AllUsers userWithId: 'SystemUser') password: '226Password'.
(AllUsers userWithId: 'GcUser') password: '226Password'.
(AllUsers userWithId: 'DataCurator') password: '226Password'.
System commitTransaction
%
```

where *226Password* is the account password used in GemStone/S 64 Bit version 2.2.6.

Post-upgrade Application Code Modifications

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

   If you use GemConnect or GemBuilder for Java, you must reinstall the appropriate version into your repository at this time. Version 3.1 requires new versions of these products. Contact GemStone Technical Support for more information. To install, use the procedure in the installation guide for the specific product.

2. Recompile application code by filein

   The upgrade process requires filein of all customer code, in order to recompile to the new bytecodes and classes introduced in version 3.1. You should have fileins available, from step 1 on page 42.

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

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

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

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

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

3. Convert persistent blocks

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

   a. Application-specific stored blocks

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

b. SortBlocks in SortedCollections

To convert the sortBlocks in persistent SortedCollections in your application, you may run the `postconv` script. This script has optional flags to specify stone name and the number of sessions to use.

`postconv [-c <numCacheWamerGems>][-h][-s <stoneName>] [-r] [-n <numberOfSessions>] [-t <tempObjCacheSize] [-u <userId>]`

`-c <numCacheWamerGems>` specifies the number of cache warmer threads in a single gem to load the object table into the shared cache before starting post-conversion. If not specified, no cache warming is done.
`-h` prints this usage information.
`-s <stoneName>` sets the name of the running stone to scan; if this option is not used, the script uses **gs64stone.**
`-n <numberOfSessions>` specifies the number of parallel sessions which will convert the instances of SortedCollection and its subclasses. By default, use one session.
`-r` specifies to reuse an existing version of `$upgradeLogDir/AllSortedCollections.bms`, if it exists. This file contains the OOPs of all instances of SortedCollections and its subclasses. By default, the existing file is deleted and a new one created.
`-t <tempObjCacheSize>` sets the size of GEM_TEMPOBJ_CACHE_SIZE in KB; by default, 20000.
`-u <userId>` is the UserId whose SymbolList includes all subclasses of SortedCollection, for which instance's sortBlocks will be converted. If not specified, defaults to SystemUser

For example,

`% **postconv -s** *stoneName31*`

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

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

4. Restore and upgrade Class comments.

In v3.1, the class comment is stored in the class as a String, and accessed via #comment and #comment: methods. This differs from the comment handling in 2.x or 3.0.x.

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

First, restore Class descriptions that were saved from your version 2.2.6 repository (as per step 5 on page 42). Descriptions are restored using the description: method.

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

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

For example,

```
% upgradeComments -s stoneName31
```

The script should complete with the message:

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

5. Convert FixedPoint instances

   If for step 3 on page 42, you decided to continue to use the old ScaledDecimal class, now named FixedPoint, do not perform this step.

   If your 2.2.6 application included instances of ScaledDecimal, following conversion to 3.x they are now instances of FixedPoint. If you intend to use the new ScaledDecimal implementation rather than FixedPoint, you must perform a manual step to convert each instance to a new ScaledDecimal. To do this, find all instances of FixedPoint and execute code similar to the following:

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

   Verify that errorcount is 0, and commit.

6. Convert reimplemented classes

   Optionally, you may wish to convert instances of reimplemented classes. Instances of Float, SmallFloat, LargePositiveInteger, LargeNegativeInteger, DoubleByteString, DoubleByteSymbol or QuadByteString in your version 2.2.6 repository are not converted during conversion to 3.x. The instances auto-migrated to the new class each time they are faulted into the VM.

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

7. Rebuild all indexes.

8. If you have modified any kernel class methods of the previous version or if you have added methods to kernel classes, carefully compare your changes with the changes in version 3.1 to see whether your changes are still necessary or appropriate. Carefully review the Release Notes for v3.0 and v3.1, as well as examining code in the image. If you changes are still applicable, file in the changes and commit.

Backup repository and configure GBS

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

2. If you are using GBS clients, ensure you have upgraded to a supported version. You must use GBS version 7.5 or later to connect to a GemStone/S 64 Bit v3.1 repository.

   Configure GBS to use the version 3.1 client libraries. Note that the required libraries, library naming conventions, and the process GBS uses to identify the correct library to load have changed in version 7.5. Read the updated instructions carefully.

3. See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details.

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 3.1.

NOTE

GemStone/S 64 Bit version 3.1 requires the use of GBS version 7.5 and VisualWorks Smalltalk v7.9. Earlier GBS versions cannot log into GemStone/S 64 Bit 3.1 due to changes including shared library requirements and naming.

See "VisualWorks Versions and Platforms" on page 50 for the versions of VisualWorks that may be used with GemStone/S 64 Bit v3.1.

In addition to using the appropriate version of GBS, you must use GemStone/S 64 Bit 3.1 client libraries to be able to log in to the GemStone/S 64 Bit 3.1 server. The required client libraries have changed and the library naming conventions have changed in v3.1, and GBS version 7.5 follows new rules for library loading. Please review the details in the following sections for the new process.

Applications using GBS version 7.5 can log in to either GemStone/S or GemStone/S 64 Bit repositories (but not simultaneously), provided that the appropriate client libraries for each are used.

On UNIX and Linux platforms, GemStone/S 64 Bit provides both 64-bit libraries and 32-bit libraries. With 64-bit VisualWorks installations, it is possible to login either RPC or linked. 32-bit VisualWorks processes cannot load 64-bit libraries; you must use 32-bit libraries with 32-bit VisualWorks installations. With 32-bit VisualWorks installations, only RPC logins are possible.

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

VisualWorks Versions and Platforms

The following table summarizes GemBuilder for Smalltalk, client Smalltalk and client operating system support for GemStone/S 64 Bit v3.1.

GemStone/S 64 Bit v3.1 requires GBS version 7.5, in order to log in. Older versions of GBS will not be able to log into v.3.1.

GBS version 7.5

The following VisualWorks and OS platforms and versions are supported with GemStone/S 64 Bit 3.1 and GemBuilder for Smalltalk version 7.5.

Note that GBS v7.5 requires VisualWorks Smalltalk version 7.9. GBS v7.5 cannot be loaded into older versions of VisualWorks.

VW 7.9 32-bit	VW 7.9 64-bit
▸ Windows 2008, Windows 7, and Windows XP ▸ Solaris 10 on SPARC ▸ SuSE Linux ES 10 and 11 RedHat Linux ES 5.5 and 6.1	▸ Solaris 10 on SPARC ▸ SuSE Linux ES 10 and 11 Red Hat Linux ES 5.5 and 6.1

For more details, see the *GemBuilder for Smalltalk Installation Guide* for v7.5.

GemStone/S 64 Bit 3.1 Windows Client Installation

To allow GemStone/S 64 Bit clients to connect from 32-bit GBS applications and to run topaz 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 ability to run gslist to get information about remote UNIX servers
- The 32-bit client libraries to be used by GBS for RPC logins from Windows.

If you do not require topaz or gslist on Windows, you may wish to extract only the client libraries from the Windows installation, and move these files to an appropriate directory on the client. See "Copying GemStone/S 64 Bit 3.1 Libraries" on page 52.

To install the GemStone/S 64 Bit Windows Client product, use the following procedure.

1. The GemStone/S 64 Bit client installation for Windows is provided as a zipped archive with a name similar to `GemBuilderC3.1.0-x86.Windows_NT.zip`. Move this distribution file to the directory location in which this will be installed, *InstallDir*.
2. Create a directory named `GemBuilderC3.1.0-x86.Windows_NT` (or another name of your choice), into which the client will be installed, and make this directory the working directory. For example:

   InstallDir> **mkdir GemBuilderC3.1.0-x86.Windows_NT**
   InstallDir> **cd GemBuilderC3.1.0-x86.Windows_NT**
3. Unzip the distribution file using unzip. For example,

   InstallDir`\GemBuilderC3.1.0-x86.Windows_NT>` **unzip GemBuilderC3.1.0-x86.Windows_NT.zip**

   In addition to several subdirectories, the unzipped distribution also contains several text files: `PACKING.txt`, which lists all of the GemStone files, and `version.txt`, which identifies the product and release, and files containing licensing information.
4. Add the installation directory

   InstallDir`\GemBuilderC3.1.0-x86.Windows_NT\bin`

   to the machine search path, %PATH%.

Copying GemStone/S 64 Bit 3.1 Libraries

If your GBS clients are on the same machine as an installation of the GemStone/S 64 Bit v3.1 server (either the server subset available on Windows, or a full installation on Solaris or Linux), the client libraries can be accessed from that location, and you do not need to follow the steps in this section.

If GBS clients will be running on a platform other than that of your GemStone/S 64 Bit v3.1 server installation, you will need to download the server for that platform to access the libraries to load into the client. The Installation Instructions for Windows are in the previous section. For Solaris or Linux, see the appropriate Installation Guide for that platform.

For remote linked GBS logins (which are not available from Windows), you will normally need access to a larger part of the server product. This section applies primarily to GBS clients that will login RPC from machines that are not running Stone, gem, or caches.

Copying libraries on Windows

If you have installed the full GemStone/S 64 Bit Windows Client product, and updated your machine search path, you do not need to follow these steps.

However, if you do not need the server components other than the shared libraries, you can simplify by only copying the required shared libraries.

There are several options when copying the shared libraries:

- You may copy these libraries to the working directory of your VisualWorks image or to a directory under %VISUALWORKS%\bin.
- You may copy these libraries to an existing directory on the machine search path %PATH%.
- You may create a new directory, copy the libraries to this location, and add this directory to the machine search path.
- You may copy these shared libraries to another location, not on the path, and specify the absolute path and filename of the library using `libraryName:` or from the GBS dialog.

Note that in GBS version 7.5 and later, leaving `libraryName` empty will not detect client libraries in directories on the %PATH%. See the options for setting `libraryName` listed on page 54 for details on the rules for finding the client libraries in version 7.5 and later.

The following shared libraries files should be copied from the GemStone/S 64 Bit Windows client installation directory, *InstallDir*`\GemBuilderC3.1.0-x86.Windows_NT\bin`, to your selected shared library location:

`libgcirpc-3.1.0-32.dll`
`libssl-3.1.0-32.dll`
`msvcr90.dll`

While only **`libgcirpc-3.1.0-32.dll`** is specifically loaded using `libraryName:`, the other libraries must be available, usually in the same directory.

Copying libraries on Solaris or Linux

If your GBS clients have access to GemStone/S 64 Bit v3.1 server installation, you do not need to follow the steps in this section. If GBS clients are on a separate machine, but do not need access to other GemStone/S 64 Bit server functionality, you can simplify your client GBS installation by only copying the required shared libraries.

There are several options when copying the shared libraries:

- You may copy these libraries to the working directory of your VisualWorks image or to a directory under $VISUALWORKS/bin.
- You may copy these libraries to an existing directory on the machine search path $LD_LIBRARY_PATH.
- You may create a new directory, copy the libraries to this location, and add this directory to the path.
- You may copy these shared libraries to another location, not on the path, and specify the absolute path and filename of the library using `libraryName:` or from the GBS dialog.

The following shared libraries files should be copied from the GemStone/S 64 Bit client installation directory, to your selected shared library location:

for 32-bit VisualWorks clients

```
lib32/libgcirpc-3.1.0-32.so
lib32/libssl-3.1.0-32.so
```

for 64-bit VisualWorks clients:

```
lib/libgcirpc-3.1.0-64.so
lib/libssl-3.1.0-64.so
```

Installing GemStone/S 64 Bit 3.1 Libraries onto GBS Clients

This section describes the procedure for configuring GBS to recognize the server shared libraries.

1. Quit and restart the Smalltalk VM (if you have not already done so), and ensure that you have positioned the set of required v3.1 client libraries in your intended location. Note that while a specific named library is loaded, the other required library file/s must be available; normally they are in the same directory.
2. Specify the correct library file to load using the GBS configuration parameter `libraryName`. You can set this parameter using the GBS settings dialog; select the tab titled “Server Communication” and the field libraryName:.

   You may also set the `libraryName` by executing an expression of the form:
   `GbsConfiguration current libraryName:` *libNameString*

   The `libraryName` may be set to a full path and filename, just the filename, or to an empty string. These options are listed below.

   If the system fails to load the library it finds, the resulting error dialog allows you to navigate to and select the specific library file.

There are a number of ways to specify the library file that should be loaded.

- Set `libraryName` to the full path and file name of the client library file. For example, for clients on Windows:

  ```
  GbsConfiguration current libraryName:
  'InstallDir\GemBuilderC3.1.0-x86.Windows_NT\bin\libgcirpc-3.1.0-32.dll'
  ```

  or for 64-bit GBS clients on Solaris that will log in linked as well as RPC:

  ```
  GbsConfiguration current libraryName:
  'InstallDir/GemStone64Bit3.1.0-sparc.Solaris/lib/libgcilnk-3.1.0-64.so'
  ```

  This does not require that the directory containing the libraries be on the %PATH% or $LD_LIBRARY_PATH.

- Set `libraryName` to the file name of the client library file. For example,

  ```
  GbsConfiguration current libraryName:
     'libgcirpc-3.1.0-32.dll'
  ```

  This requires that the libraries be in a directory on the %PATH% or $LD_LIBRARY_PATH; or on Linux, in the standard Linux library directories.

  On Windows, it may also be in the current working directory or in the VisualWorks executable directory.

- Leave `libraryName` set to the empty string. This is the default, equivalent to:

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

  This requires that the client libraries be in the current working directory, or in the bin directory or subdirectory of your VisualWorks image's VISUALWORKS directory.

  Be sure to verify that no other versions of GemStone shared libraries are in these locations.

- Use the file dialog to select the library name and path.

  If the library name specified by `libraryName` cannot be loaded, the resulting notification provides the option of selecting a file using a file dialog. The load error may be due to an incorrect `libraryName`, copying the wrong library, or an additional required library failing to load.

  Navigate to the location of the shared library and select the file. The dialog provides filename filtering, so only valid library names will appear.

  Selecting a file will set `libraryName` to the full path selected.

3. Save your image to preserve the new `libraryName` setting. To ensure the correct libraries are loaded on image startup, exit and restart your image.