GemStone®

# GemStone/S 64 Bit™ Installation Guide

for the Mac OS X on i386
(Development only)

**Version 3.3.1**

June 2016

## INTELLECTUAL PROPERTY OWNERSHIP

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

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 GemTalk Systems.

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

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

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

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

## COPYRIGHTS



## PATENTS

GemStone software is 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

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

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

**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.

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

**Microsoft**, **MS**, **Windows, Windows 7**, **Windows 2008**, and **Windows 8** are registered trademarks of Microsoft Corporation in the United States and other countries.

**Linux** is a registered trademark of Linus Torvalds and others.

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

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

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

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

**Apple**, **Mac**, **Mac OS**, and **Macintosh** 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. GemTalk Systems cannot attest to the accuracy of all trademark information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

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

# Preface

## About This Documentation

This document explains how to install GemStone/S 64 Bit™ version 3.3.1, and how to upgrade from previous GemStone/S 64 Bit versions.

For information regarding new and modified features in GemStone/S 64 Bit v3.3.1, please refer to the *GemStone/S 64 Bit Release Notes* for version 3.3.1.

These documents are also available on the GemStone customer website, as described below.

## Terminology Conventions

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

## Technical Support

### Support Website

**gemtalksystems.com**

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

- **Documentation** for the current and for previous released versions of all GemTalk products, in PDF form.
- **Product download** for the current and selected recent versions of GemTalk software.

- **Bugnotes**, identifying performance issues or error conditions that you may encounter when using a GemTalk product.
- **TechTips**, providing information and instructions that are not in the documentation.
- **Compatibility matrices**, listing supported platforms for GemTalk product versions.

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

## Help Requests

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

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

**Website: techsupport.gemtalksystems.com**

**Email: techsupport@gemtalksystems.com**

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

When submitting a request, please include the following information:

- Your name and company name.
- The versions of GemStone/S 64 Bit and of all related GemTalk 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 GemTalk holidays.

## 24x7 Emergency Technical Support

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

# Training and Consulting

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

# Table of Contents

## *Chapter 1. Installing GemStone/S 64 Bit Version 3.3.1*



## *Chapter 2. Upgrading from GemStone/S 64 Bit 3.3*

## *Chapter 3. Upgrading from GemStone/S 64 Bit 3.x versions*



## *Chapter 4. Converting from GemStone/S 64 Bit 2.4.x versions*



## *Chapter 5. Upgrading GsDevKit Applications*



## *Chapter 6. Configuring GBS for GemStone/S 64 Bit*

Chapter

# 1 Installing GemStone/S 64 Bit Version 3.3.1

This chapter describes the procedure for installing GemStone/S 64 Bit version 3.3.1 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 17 provides references to procedures and related information in the *System Administration Guide.*

**GemStone/S 64 Bit on Mac OS X/Darwin is supported for development use only, not for use in production.**

## Review the Installation Procedure

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

# Check the System Requirements

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

## Platform

- Intel Core 2 Duo or later

## RAM and Swap space

- While small installations can run on systems with only a few GB of physical RAM, increasing RAM is important for GemStone performance.
- Adequate free disk space on the machine's boot partition beyond other system needs.

## 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

- OS X 10.9.5 (Mavericks), with Darwin 13.4.0 kernel

## C/C++ Compiler

- Apple LLVM version 6.0 (clang-600.0.56)

  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

- GNU gdb 6.3.50-20050815 (Apple version gdb-1469) (Wed May 5 04:36:56 UTC 2010)

  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. You must be logged in as root, or as a user with administrative privileges, in order to use a debugger or to generate stack traces from GemStone processes.

# Configure the Operating System

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

### 1. Shared Memory

The maximum shared memory segment should be set to a value larger than your desired Shared Page Cache size, and not more than 75% of your real memory size. In addition, for most system you will need to increase the system-wide limit on shared memory segments.

Note that `kern.sysv.shmmax` is configured in bytes, but `kern.sysv.shmall` is configured in pages with a base page size of 4096.

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

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

To set shared memory, you would include the following text in the `/etc/sysctl.conf` file, creating this file if it does not exist. The setting of the shared memory size is read from this file during the boot process.

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

### 2. Number of Processes

The default limits on the number of processes are adequate for up to about 250 GemStone users, for configurations establishing security via captive account. To allow more processes for the GemStone administrative userid, increase the settings for `kern.maxprocperuid` and **`kern.maxproc`** in `/etc/sysctl.conf`.

### 3. PAM

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

PAM authentication definitions are files under the directory `/etc/pam.conf/`.

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

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

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

```
auth                required            pam_opendirectory.so
```

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

```
auth                required            pam_opendirectory.so
```

You can allow the "other" authentication stack to be used for GemStone authentication by editing the file `/etc/pamd.conf/other` so it includes the following:

```
auth                required            pam_opendirectory.so
```

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

### 4. Configuring GemStone' Shared Page Cache

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

For more general information about these and other configuration options, see Appendix A of the *System Administration Guide*.

### 5. 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.

### 6. 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.

# 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/rdisk*`
: Optional raw partitions for repository extents and transaction logs.

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

*/InstallDir/*`GemStone64Bit3.3.1-i386.Darwin`
: 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.

1. See the *System Administration Guide* for more information.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.:
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 -k
   ```

   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 `GemStone64Bit3.3.1-i386.Darwin.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 GemStone64Bit3.3.1-i386.Darwin.zip
   ```

8. The *InstallDir* now contains a GemStone directory with a name similar to `GemStone64Bit3.3.1-i386.Darwin`.

   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, 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 and are not appropriate for the new installation, you must specifically unset each one. For example, depending on your shell:

```
% unsetenv GEMSTONE GEMSTONE_SYS_CONF
$ unset GEMSTONE GEMSTONE_SYS_CONF
```

c. Set the environment variable GEMSTONE to the *full pathname* (starting with a slash) of your new GemStone installation directory. For example, depending on your shell:

```
% setenv GEMSTONE InstallDir/GemStone64Bit3.3.1-i386.Darwin
$ GEMSTONE=InstallDir/GemStone64Bit3.3.1-i386.Darwin
$ export GEMSTONE
```

## Set the GemStone Key File

To run GemStone, you must have a key file for the correct version of GemStone/S 64 Bit and for the appropriate platform. The keyfile must be located where GemStone can find it on startup:

- `$GEMSTONE/sys/gemstone.key`
- `$GEMSTONE/sys/community.starter.key`
- The KEYFILE configuration parameter in theconfiguration file used by the stone, such as `$GEMSTONE/data/system.conf`

### Licensed Customer key file

You may use a keyfile from version 3.3 with your version 3.3.1 installation. If you are upgrading from 3.2.x or earlier, or you have questions about your keyfile or license limits, email keyfiles@gemtalksystems.com, or contact GemTalk Technical Support.

### Community key file

The GemStone distribution includes a community key file, `community.starter.key`., with product and system limits per the Community and Web Edition License. See https://gemtalksystems.com/licensing for details on the license terms.

If you do not install a customer keyfile, this starter keyfile will be used instead.

### Installing a keyfile

If you specify the location and name of the keyfile using the KEYFILE configuration parameter, edit the configuration file that will be used by the v3.3.1 stone to include the location and name of the keyfile.

To set the keyfile in the default location:

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

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

2. Copy the keyfile into this directory, using the name `gemstone.key`.

   `cp` *mykeyfile.key* `$GEMSTONE/sys/gemstone.key`

3. Change the file and directory permissions so that they are not writable:

```
% chmod 555 $GEMSTONE/sys/gemstone.key
% chmod 555 $GEMSTONE/sys
```

## Verify TCP/IP

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

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

   `% /sbin/ping -c 1` *hostname*

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

## Define the NetLDI Service

The NetLDI service, by default `gs64ldi`, should be defined in your system services database. A NetLDI is required for certain kinds of local and remote sessions to log into GemStone, and if it cannot be resolved by name, you must refer to it by port number. For clients on remote machines, the same NetLDI service name and port number must be defined on the remote machines as well as the main host.

If you are upgrading from a previous version, you may need to keep the NetLDI for that version running. In this case, select a distinct name and port for the NetLDI for GemStone/S 64 Bit 3.3.1.

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

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

   If you are using a local copy of the system services database, **/etc/services**, then check in this file for a definition for `gs64ldi`.

   ```
   % grep gs64ldi /etc/services
   ```

   If you are using NIS or LDAP, consult your UNIX system administrator for assistance.

   If `gs64ldi` is defined, skip the rest of this procedure and continue with the installation at "Run the Installation Script" on page 14.

   If it is not defined, continue performing this procedure.

2. Add an entry similar to the following to the system services database:

   ```
   gs64ldi 50377/tcp #GemStone/S 64 Bit 3.3.1
   ```

   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).

3. If several machines will be running GemStone, have the UNIX system administrator update the system services database for each machine. This includes Windows client machines as well as UNIX nodes. 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 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*.)

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.*

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 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.

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* 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 3.3.1 has been installed, you should notify each GemStone user of the installation and explain how to use the `gemsetup` files.

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 3.3.1 directory. For example, depending on your shell:

```
% setenv GEMSTONE InstallDir/GemStone64Bit3.3.1-i386.Darwin
$ GEMSTONE=InstallDir/GemStone64Bit3.3.1-i386.Darwin
$ export GEMSTONE
```

2. Invoke the script `gemsetup`. For example, depending on your shell:

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

# 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 3.3.1 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*. This chapter describes in detail the many options for configuring your 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.*
- Chapter 3 of the *System Administration Guide* has additional information about setting up distributed configurations.
- To start and stop the GemStone object server, refer to instructions in Chapter 4 of the *System Administration Guide*.

Chapter

# 2 Upgrading from GemStone/S 64 Bit 3.3

This chapter describes how to upgrade an existing GemStone/S 64 Bit 3.3 installation to GemStone/S 64 Bit version 3.3.1.

For upgrading from GemStone/S 64 Bit versions 3.1.x and 3.2.x, which require extra steps such as recompilation, see Chapter 3, starting on page 25

For upgrading from GemStone/S 64 Bit 2.x versions, which require conversion, see Chapter 4 on page 33.

If you are using GemBuilder for Smalltalk (GBS), you also need to upgrade the client libraries that are used by GBS. You may also need to upgrade your version of GBS; versions of GBS earlier than 8.1 or 5.4.3 are not compatible with v3.3.1. See Chapter 6 for supported versions of GBS for use with GemStone/S 64 Bit 3.3.1, and instructions on installing updated client libraries.

### Note on ICU library version

The upgrade from 3.2.x to version 3.3 introduced a new version of the ICU libraries, which created potential problems with Indexes and SortedCollections that rely on the ICU libraries for collation; see the bugnote for bug 46056. If your 3.1.x or 3.2.x application contained SortedCollections or Indexes that involve ICU collation prior to the ugprade to 3.3, and you have not already performed the resort and rebuild, respectively, you should do this during the upgrade process to avoid the risk of errors.

# Upgrade Procedure

The following list summarizes the steps necessary to perform the upgrade to GemStone/S 64 Bit version 3.3.1.



*NOTE*

*The following instructions use the version number 3.3 to refer to the version you are upgrading from, and version number 3.3.1 indicate the target version you are upgrading to.*

## Prior to Upgrade in existing application

### 1. Check for use of deprecated methods

Verify that your application does not invoke any methods that were deprecated in previous releases, by enabling error or logging on deprecation in your existing repository. Deprecated methods are subject to removal in major releases; finding them before upgrading allows the deprecation messages to provide replacement instructions.

For details on finding deprecated methods, refer to the *Programming Guide for GemStone/S 64 Bit* v3.3.

### 2. File out modifications to GemStone classes

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

You will need to carefully compare these changes with GemStone/S 64 Bit 3.3.1 kernel methods, and refer to the *Release Notes* for version 3.3.1 to determine whether your changes are still necessary or appropriate.

*CAUTION*

*Any changes that you have made to the GemStone/S 64 Bit kernel classes will be lost during upgrade; you MUST file these out in order to preserve the changes in version 3.3.1.*

## Prepare for Upgrade

### 1. Install and configure GemStone/S 64 Bit 3.3.1

Install GemStone/S 64 Bit 3.3.1 to a new installation directory, separate from the installation directory for version 3.3, as described in Chapter 1, starting on page 7.

Configure GemStone/S 64 Bit 3.3.1 the way you expect to use it — that is, with the appropriate extent locations and sizes.

If you copy the configuration files from your previous version to the version 3.3.1, be sure to review any changes in configuration parameters to determin if changes are needed.

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

### 2. Reset SystemUser password

Log in to the version 3.3 system as a user with OtherPassword privilege, such as DataCurator, 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, and resets the password for DataCurator and GcUser.

### 3. Stop user activity

Log in to the version 3.3 system as a user with SessionAccess and SystemControl privileges, such as DataCurator, and halt all user activity on the repository.

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

### 4. Shut down the repository

You may now shut down the Stone. At the UNIX command line:

```
% stopstone stone33
```

where *stone33* is the name of the version 3.3 stone on this machine. The repository must be cleanly shut down to avoid needing recovery when it is restarted with the new version's executables.

### 5. Set up the version 3.3.1 environment.

Set the environment variables required for the upgrade.

C shell:

```
% setenv GEMSTONE InstallDir331
% set path = ($GEMSTONE/bin $path)
% setenv upgradeLogDir tempDir
```

Bourne or Korn shell:

```
$ GEMSTONE=InstallDir331
$ export GEMSTONE
$ export PATH=$GEMSTONE/bin:$PATH
$ upgradeLogDir=tempDir
$ export upgradeLogDir
```

where *InstallDir331* is the GemStone/S 64 Bit version 3.3.1 installation and *tempDir* is a temporary directory for which you have write permission.

*NOTE*

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

### 6. Copy extent files

Copy your version 3.3 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 3.3 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 3.3.1 installation. For example:

```
% cp InstallDir33/data/extent0.dbf 331location
% cp InstallDir33/data/extent1.dbf 331location
% cp InstallDir33/data/extent2.dbf 331location
```

where *331location* is the location specified by DBF_EXTENT_NAMES in the configuration file that will be used in version 3.3.1.

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 3.3.1. Transaction logs from earlier versions are not compatible with version 3.3.1. If the transaction log directories will be reused for version 3.3.1, any transaction logs should be deleted or copied elsewhere.

## Perform the Upgrade

### 1. Start the Stone

Start the 3.3.1 Stone on the 3.3 extents you just copied:

```
% startstone stoneName331
```

### 2. Upgrade image

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

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

```
upgradeImage [-h] [-c <cacheSize>] [-s <stoneName>]
```

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

For example,

```
% upgradeImage -s stoneName331
```

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

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

The script should complete with the message:

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

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

### 3. Restore System Account passwords

Log in to GemStone/S 64 Bit version 3.3.1 as DataCurator or SystemUser, and change the password for SystemUser, DataCurator, and GcUser to a secure password, such as the passwords used for these accounts in v3.3. For example:

```
topaz 1> printit
(AllUsers userWithId: 'SystemUser') password: '33Password'.
(AllUsers userWithId: 'GcUser') password: '33Password'.
(AllUsers userWithId: 'DataCurator') password: '33Password'.
System commitTransaction
%
```

where *33Password* is the account password used in version 3.3.

## GsDevKit Upgrade

If you are using the Open-source Development Kit for GemStone/S 64 Bit (GsDevKit, previously referred to as Seaside or GLASS), you will need to perform another step to upgrade your GsDevKit image. This step upgrades the GsDevKit base code, and you will also need to reload your application code.

For details, see Chapter 5, starting on page 45.

When you have completed the GsDevKit upgrade, continue with the upgrade process and perform the following steps.

## 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 of these products into your repository at this time.

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

### 2. File in Kernel class changes

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 3.3.1 to see whether your changes are still necessary or appropriate. Carefully review the *Release Notes* for each intervening version, as well as examining code in the image.

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

## Backup repository and configure GBS

### 1. Make backup

At this point, you should create a full backup of the upgraded repository.

### 2. Configure GBS

If you are using GBS clients, ensure you are running a supported version of GBS and client Smalltalk. You must use GBS version 8.1 or later for VW, or GBS 5.4.3 or later for VA, to connect to a GemStone/S 64 Bit v3.3.1 repository.

Configure GBS to use the version 3.3.1 client libraries. Depending on the GBS version you are upgrading from, the required libraries, library naming conventions, and the process GBS uses to identify the correct library to load may have changed.

See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details. If your GBS clients run on a different platform than your GemStone server, refer to the *Installation Guide* for that platform.

Chapter

# 3 Upgrading from GemStone/S 64 Bit 3.x versions

This chapter describes how to upgrade an existing GemStone/S 64 Bit 3.2.x, 3.1.x, or 3.0.x installation to GemStone/S 64 Bit version 3.3.1. If you are upgrading from version 3.3, which does not require recompile, see Chapter 2 on page 19. For upgrading from GemStone/S 64 Bit 2.x versions, which require conversion, see Chapter 4 on page 33.

If you are using GemBuilder for Smalltalk (GBS), you also need to upgrade the client libraries that are used by GBS. You may also need to upgrade your version of GBS; versions of GBS earlier than 8.1 or 5.4.3 are not compatible with v3.3.1. See Chapter 6 for supported versions of GBS for use with GemStone/S 64 Bit 3.3.1, and instructions on installing updated client libraries.

## Keyfiles

New keyfiles are required with GemStone/S 64 Bit version 3.3.1; keyfiles for GemStone/S 64 Bit 3.2.x or earlier will not work with v3.3.1. To acquire an updated keyfile for version 3.3.1, email keyfiles@gemtalksystems.com, or contact GemStone Technical Support, preferably providing your existing keyfile.

The keyfiles for v3.2 and later also manage access to GemConnect and GemBuilder for Java. If you are using these add-on products, you must use a keyfile with the appropriate permissions. Applications using GemConnect or GemBuilder for Java also require updated versions of these products, depending on which version you are upgrading from. These products need to be reinstalled following the upgrade process.

# Upgrade Procedure

The following list summarizes the steps necessary to perform the upgrade to GemStone/S 64 Bit version 3.3.1.



*NOTE*

*The following instructions use the version number 3.2.15 to refer to the version you are upgrading from, and version number 3.3.1 indicate the target version you are upgrading to. The process is the same when upgrading from any 3.x repository, and upgrading to any 3.2.x versions for which this Installation Guide applies.*

## Prior to Upgrade in existing application

### 1. Check for use of deprecated methods

Verify that your application does not invoke any methods that were deprecated in previous releases, by enabling error or logging on deprecation in your existing repository. Deprecated methods are subject to removal in major releases; finding them before upgrading allows the deprecation messages to provide replacement instructions.

For details on finding deprecated methods, refer to the *Programming Guide for GemStone/S 64 Bit* v3.3.

### 2. File out your application code

If you do not already have source code for your application stored externally to the GemStone repository in a code management system, it is recommended to file out all application source code. Filein of application code is used to recompile all methods. You may also write code to manually recompile methods in all classes; see "Recompile application code" on page 30 for details.

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

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

### 3. File out modifications to GemStone classes

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

You will need to carefully compare these changes with GemStone/S 64 Bit 3.3.1 kernel methods, and refer to the *Release Notes* for version 3.3 and all release notes after the

version you are upgrading from, to determine whether your changes are still necessary or appropriate. For a listing of Release Notes, see GemStone/S 64 Bit Release History.

*CAUTION*

*Any changes that you have made to the GemStone/S 64 Bit kernel classes will be lost during upgrade; you MUST file these out in order to preserve the changes in version 3.3.1.*

## Prepare for Upgrade

### 1. Install and configure GemStone/S 64 Bit 3.3.1

Install GemStone/S 64 Bit 3.3.1 to a new installation directory, separate from the installation directory for version 3.2.15, as described in Chapter 1, starting on page 7.

Configure GemStone/S 64 Bit 3.3.1 the way you expect to use it — that is, with the appropriate extent locations and sizes.

If you copy the configuration files from your previous version to the version 3.3.1, be sure to review any changes in configuration parameters to determin if changes are needed. Applications upgrading from v3.2.x will need to set the value for STN_GEM_LIBICU_VERSION, as described in the next section.

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

### 2. If upgrading from v3.2, set STN_GEM_LIBICU_VERSION to legacy version

Repositories that are upgrading from 3.2.x should edit the system.conf file used by the v3.3.1 stone to include the following:

```
STN_GEM_LIBICU_VERSION = "51.2";
```

Do not perform this step if you are upgrading from version 3.0.x or 3.1.x.

### 3. Reset SystemUser password

Log in to the version 3.2.15 system as a user with OtherPassword privilege, such as DataCurator, 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, and resets the password for DataCurator and GcUser.

### 4. Stop user activity

Log in to the version 3.2.15 system as a user with SessionAccess and SystemControl privileges, such as DataCurator, and halt all user activity on the repository.

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

### 5. Shut down the repository

You may now shut down the Stone. At the UNIX command line:

```
% stopstone stone3215
```

where *stone3215* is the name of the version 3.2.15 stone on this machine. The repository must be cleanly shut down to avoid needing recovery when it is restarted with the new version's executables.

### 6. Set environment variables for version 3.3.1

Set the environment variables required for the upgrade.

C shell:

```
% setenv GEMSTONE InstallDir331
% set path = ($GEMSTONE/bin $path)
% setenv upgradeLogDir tempDir
```

Bourne or Korn shell:

```
$ GEMSTONE=InstallDir331
$ export GEMSTONE
$ export PATH=$GEMSTONE/bin:$PATH
$ upgradeLogDir=tempDir
$ export upgradeLogDir
```

where *InstallDir331* is the GemStone/S 64 Bit version 3.3.1 installation and *tempDir* is a temporary directory for which you have write permission.

*NOTE*

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

### 7. Copy extent files

Copy your version 3.2.15 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 3.2.15 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 3.3.1 installation. For example:

```
% cp InstallDir3215/data/extent0.dbf 331location
% cp InstallDir3215/data/extent1.dbf 331location
% cp InstallDir3215/data/extent2.dbf 331location
```

where *331location* is the location specified by `DBF_EXTENT_NAMES` in the configuration file that will be used in version 3.3.1.

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 3.3.1. Transaction logs from earlier versions are not compatible with version 3.3.1. If the transaction log directories will be reused for version 3.3.1, any transaction logs should be deleted or copied elsewhere.

## Perform the Upgrade

### 1. Start the Stone

Start the 3.3.1 Stone on the 3.2.15 extents you just copied:

```
% startstone stoneName331
```

### 2. Upgrade image

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

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

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

For example,

```
% upgradeImage -s stoneName331
```

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

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

The script should complete with the message:

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

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

### 3. Restore System Account passwords

Log in to GemStone/S 64 Bit version 3.3.1 as DataCurator or SystemUser, and change the password for SystemUser, DataCurator, and GcUser to a secure password, such as the passwords used for these accounts in v3.2.15. For example:

```
topaz 1> printit
(AllUsers userWithId: 'SystemUser') password: '3215Password'.
(AllUsers userWithId: 'GcUser') password: '3215Password'.
(AllUsers userWithId: 'DataCurator') password: '3215Password'.
System commitTransaction
%
```

where *3215Password* is the account password used in version 3.2.15.

## GsDevKit Upgrade

If you are using the Open-source Development Kit for GemStone/S 64 Bit (GsDevKit, previously referred to as Seaside or GLASS), you will need to perform another step to upgrade your GsDevKit image. This step upgrades the GsDevKit base code, and you will also need to reload your application code.

For details, see Chapter 5, starting on page 45.

When you have completed the GsDevKit upgrade, continue with the upgrade process and perform the following steps.

## 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 of these products into your repository at this time.

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

### 2. File in Kernel class changes

If you have modified any kernel class methods of the previous version or if you have added methods to kernel classes, compare your changes with the changes in version 3.3.1 to see whether your changes are still necessary or appropriate. Carefully review the *Release Notes* for each intervening version, as well as examining code in the image.

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

### 3. Recompile application code

The upgrade process requires all executable methods to be recompiled. You should have fileins available, from Step 2. on page 26. Alternatively, you may iterate classes in your image and recompile each one. If you are reusing scripts that managed recompile for the 2.x conversion, verify that the configuration parameter `GemConvertArrayBuilder` is not set.

GsDevKit environments will not need to perform this step.

a. Recompile by filein

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

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

b. Recompile within image

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

*Class* `recompileAllMethods`

You should ensure that you do not miss any classes. If you have instance of older versions of your classes, you will need to recompile methods on these older versions. For example:

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

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

### 4. Recompile source code in persistent blocks

The compiled code in persistent blocks also requires recompile before it can be executed. All persistent executable blocks will need to be recompiled as part of upgrading.

c. Application-specific stored blocks

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

d. SortBlocks in SortedCollections

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

`postconv [-c <`*`numCacheWamerGems`*`>][-h][-s <`*`stoneName`*`>] [-r] [-n <`*`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` *stoneName331*

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

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

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

### 5. If upgrading from 3.1.x, resort for ICU-based collation

This step is only required if you are upgrading from 3.1.x, which used ICU version 48.1.1 for collation, and only if your 3.1.x application has SortedCollections that

include unicode strings, or indexes on unicode strings, or if your application is in Unicode comparison mode.

If this is the case, you must resort all instances of SortedCollection that involve unicode strings (or legacy strings in Unicode Comparison Mode), and drop and rebuild indexes that involve instances of unicode strings (or legacy strings in Unicode Comparison Mode), to ensure these are sorted according the ICU collation now in use in the image.

### 6. Regenerate cached instances of PetitParser classes

Instance of PetitParser classes (classes with names that begin with PP) are not automatically converted to the new class versions. If you are upgrading from v3.2x and using PetitParser classes directly, and you have persistent instances, you should regenerate these instances.

## Backup repository and configure GBS

### 1. Make backup

At this point, you should create a full backup of the upgraded repository.

### 2. Configure GBS

If you are using GBS clients, ensure you are running a supported version of GBS and client Smalltalk. You must use GBS version 8.1 or later for VW, or GBS 5.4.3 or later for VA, to connect to a GemStone/S 64 Bit v3.3.1 repository.

Configure GBS to use the version 3.3.1 client libraries. Depending on the GBS version you are upgrading from, the required libraries, library naming conventions, and the process GBS uses to identify the correct library to load may have changed.

See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details. If your GBS clients run on a different platform than your GemStone server, refer to the *Installation Guide* for that platform.

Chapter

# 4 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 3.3.1.

GemStone/S 64 Bit version 3.3.1 supports upgrade from GemStone/S 64 Bit versions 2.4.x and later. To upgrade from 3.x versions, see Chapter 2 on page 19 for version 3.3 or Chapter 3 on page 25 for versions 3.0.x, 3.1.x, or 3.2.x.

If you are upgrading from an earlier 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. You should carefully read the *GemStone/S 64 Bit Release Notes* for v3.2, which contain important information on the changes.

If you are using GemBuilder for Smalltalk (GBS), you also need to upgrade the client libraries that are used by GBS. You may also need to upgrade your version of GBS; versions of GBS earlier than 7.6.1 are not compatible with v3.3.1. See Chapter 6 for supported versions of GBS for use with GemStone/S 64 Bit 3.3.1, and instructions on installing updated client libraries.

## Keyfiles

New keyfiles are required with GemStone/S 64 Bit version 3.3.1. Keyfiles for GemStone/S 64 Bit 2.x will not work with v3.3.1. To acquire an updated keyfile, email keyfiles@gemtalksystems.com, or contact GemStone Technical Support, preferably providing your existing keyfile.

The keyfiles for v3.2 and later also manage access to GemConnect and GemBuilder for Java. If you are using these add-on products, you must use a keyfile with the appropriate permissions. Applications using GemConnect or GemBuilder for Java also require updated

versions of these products for compatibility. These products 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 3.3.1 installation, prior to performing the full upgrade. This will allow you to test your code changes in the 3.3.1 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.

# Upgrade Procedure

The following list summarizes the steps necessary to perform the upgrade to GemStone/S 64 Bit version 3.3.1.



*NOTE*

*The following instructions use the version number 2.4.8 to represent any 2.x version from which upgrade/conversion is supported, and 3.3.1 to represent a 3.3.1 or later upgrade destination version for which this Install Guide applies. The procedure is the same regardless of which versions you have.*

## Prior to Upgrade in existing application

### 1. File out your application code

If you do not already have source code for your application stored externally to the GemStone repository in a code management system, it is recommended to file out all application source code. Filein of application code is used to recompile all methods. You may also write code to manually recompile methods in all classes; see "Recompile application code" on page 39 for details.

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

GemStone supports multiple versions of the same class, but tools operate on the most recent version of the classes. If you have instance of older versions of your applications

classes that have not been migrated to the latest version, these class versions will not be upgraded by filein. We recommend that you migrate all instances to the most recent version of your application classes.

### 2. File out modifications to GemStone classes

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 3.3.1 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, 3.2, and 3.3. See the Release Notes for version 3.0 for details of the major changes, and read these carefully before filing 2.x code modifications into your version 3.3.1 repository. You should also review the changes documented in all subsequent release notes. For a listing of Release Notes, see GemStone/S 64 Bit Release History.

### 3. Update references to ScaledDecimal, if necessary

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

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

### 4. Verify returned value from disallowUsedPasswords is Boolean

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

```
AllUsers disallowUsedPasswords
```

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

```
AllUsers disallowUsedPasswords: true.
System commitTransaction.
```

### 5. Remove indexes on instances of reimplemented server classes

If you have indexes that include instance of any of the following classes, these indexes will need to be removed and rebuilt in version 3.3.1 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.

### 6. Prepare class comments for 3.x changes, if necessary

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 and configure GemStone/S 64 Bit 3.3.1

Install GemStone/S 64 Bit 3.3.1 to a new installation directory, separate from the installation directory for version 2.4.8, as described in Chapter 1 of this Installation Guide.

Configure GemStone/S 64 Bit 3.3.1 the way you expect to use it — that is, with the appropriate extent locations and sizes.

If you copy the configuration files from your previous version to the version 3.3.1, be sure to review any changes in configuration parameters to determin if changes are needed.

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

### 2. Reset SystemUser password

Log in to the version 2.4.8 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.

### 3. Stop users and shut down repository

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

Log in to Topaz as DataCurator, and execute:

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

You may now shut down the Stone:

```
% stopstone stone248
```

where *stone248* is the name of the version 2.4.8 stone on this machine. It is strongly recommended, but not required, that the repository be cleanly shut down before it is restarted under version 3.3.1. If the repository is not cleanly shut down, you must restart using the -N option.

### 4. Set up the version 3.3.1 environment.

Set the environment variables required for the upgrade.

C shell:

% **setenv GEMSTONE** *InstallDir331*
% **set path = ($GEMSTONE/bin $path)**
% **setenv upgradeLogDir** *tempDir*

Bourne or Korn shell:

$ **export GEMSTONE=***InstallDir331*
$ **export PATH=$GEMSTONE/bin:$PATH**
$ **export upgradeLogDir=***tempDir*

where *InstallDir331* is the GemStone/S 64 Bit version 3.3.1 installation and *tempDir* is a temporary directory for which you have write permission.

### 5. Copy extent files

Copy your version 2.4.8 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 2.4.8 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 3.3.1 installation. For example:

% **cp** *InstallDir248***/data/extent0.dbf** *331location*
% **cp** *InstallDir248***/data/extent1.dbf** *331location*
% **cp** *InstallDir248***/data/extent2.dbf** *331location*

where *331location* is the location specified by `DBF_EXTENT_NAMES` in the configuration file that will be used in version 3.3.1.

If version 3.3.1 will run on a platform with a different byte ordering than the version 2.4.8 repository, or if you are using raw partitions, you will need to use copydbf.

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 3.3.1. If the transaction log directories will be reused for version 3.3.1, any transaction logs should be deleted or copied elsewhere.

## Perform the Upgrade

### 1. Convert using startstone

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

% **startstone** -C *stoneName331*

### 2. Upgrade image

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

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

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

For example,

`% upgradeImage -s stoneName331`

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

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

The script should complete with the message:

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

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

### 3. Restore System Account passwords

Log in to GemStone/S 64 Bit version 3.3.1 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 2.4.8 value:

```
topaz 1> printit
(AllUsers userWithId: 'SystemUser') password: '248Password'.
(AllUsers userWithId: 'GcUser') password: '248Password'.
(AllUsers userWithId: 'DataCurator') password: '248Password'.
System commitTransaction
%
```

where *248Password* is the account password used in GemStone/S 64 Bit version 2.4.8.

## GsDevKit Upgrade

If you are using the Open-source Development Kit for GemStone/S 64 Bit( GsDevKit, previously referred to as Seaside or GLASS), you will need to perform another step to upgrade your GsDevKit image. This step upgrades GsDevKit base code, and you will also need to reload your application code.

For details, see Chapter 5, starting on page 45.

When you have completed the GsDevKit upgrade, continue with the upgrade process and perform the following steps.

## 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 3.3.1 requires new versions of these products. Contact GemStone Technical Support for more information.

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

### 2. File in Kernel class changes

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 3.3.1 to see whether your changes are still necessary or appropriate. Carefully review the *Release Notes* for each intervening version, as well as examining code in the image.

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

### 3. Recompile application code

The upgrade process requires all executable methods to be recompiled, to use the new bytecodes and classes introduced in version 3.0. You should have fileins available, from Step 1. on page 34. Alternatively, you may iterate classes in your image and recompile each one.

GsDevKit environments will not need to perform this step.

a. Recompile by filein

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

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

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

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

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

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

b. Recompile within image

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

*Class* `recompileAllMethods`

You should ensure that you do not miss any classes. If you have instance of older versions of your classes, you will need to recompile methods on these older versions. For example:

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

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

## 4. Convert persistent blocks

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

c. Application-specific stored blocks

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

d. SortBlocks in SortedCollections

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

`postconv [-c <`*`numCacheWamerGems`*`>][-h][-s <`*`stoneName`*`>] [-r] [-n <`*`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 stoneName331
```

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

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

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

## 5. Restore and upgrade Class comments

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

As part of upgrade, 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 GsDevKit handles class comments in a way that is consistent with 3.3.1 behavior, GsDevKit environments do not need to perform this step.

First, restore Class descriptions that were saved from your version 2.4.8 repository (as per Step 6. on page 36). Descriptions are restored using the description: method.

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

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

For example,

```
% upgradeComments -s stoneName331
```

The script should complete with the message:

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

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

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

## 6. Convert instances of reimplemented server classes

Optionally, you may wish to convert instances of reimplemented classes. Instances of Float, SmallFloat, LargePositiveInteger, LargeNegativeInteger, DoubleByteString, DoubleByteSymbol or QuadByteString in your version 2.4.8 repository are not

converted during conversion to 3.x. The instances 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. Convert FixedPoint instances

If for Step 3. on page 35, you decided to continue to use the old ScaledDecimal class, now named FixedPoint, do not perform this step.

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

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

Verify that errorcount is 0, and commit.

### 8. 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 5. on page 35 for more details on this requirement.

### 9. Upgrade user account passwords

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

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

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

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

See image comments for more details.

## Backup repository and configure GBS

### 1. Make backup

At this point, you should create a full backup of the upgraded repository.

### 2. Configure GBS

If you are using GBS clients, ensure you have upgraded to a supported version of GBS and client Smalltalk. You must use GBS version 8.1 or later for VW, or GBS 5.4.3 or later for VA, to connect to a GemStone/S 64 Bit v3.3.1 repository.

Configure GBS to use the version 3.3.1 client libraries. Note that the required libraries, library naming conventions, and the process GBS uses to identify the correct library to load have changed.

See Chapter 6, 'Configuring GBS for GemStone/S 64 Bit' for details. If your GBS clients run on a different platform than your GemStone server, refer to the *Installation Guide* for that platform.

Chapter

# 5 Upgrading GsDevKit Applications

This chapter describes the additional upgrade step that applies when upgrading an application that is using the Open-source Development Kit for GemStone/S 64 Bit (GsDevKit, previously referred to as Seaside or GLASS) to GemStone/S 64 Bit version 3.3.1.

If you are using the most recent version, from github.com/GsDevKit/GsDevKit_home, then you may use the upgrade scripts provided there to perform the entire upgrade, rather than using the instructions in this Installation Guide.

The complete process for upgrading GemStone is described in the relevant chapter of this Installation Guide; Chapter 4 for version 2.4.x and Chapter 2 for upgrade from 3.x. You will need to follow the steps in that chapter, which will note the point at which the GsDevKit upgrade takes place.

Due to the differences when GsDevKit is added to the GemStone/S 64 Bit environment, not all steps of the upgrade process apply. The upgrade instructions note the differences.

While the upgrade process below should work in most cases, some upgrade scenarios may require individual attention. If you encounter issues or have questions about the GsDevKit upgrade, send email to `beta@seaside.gemstone.com`.

## Upgrade Procedure

After you have completed the GemStone/S 64 Bit image upgrade, as described in Chapter 2 or Chapter 4, you can upgrade your GsDevKit application.



After the GsDevKit upgrade is complete, you should return and complete the remaining upgrade steps as described in Chapter 2 or Chapter 4.

## Configure the GsDevKit Upgrade

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 v2.x to v3.0 changes were extensive, and impacted most application code.

Before upgrading your GsDevKit application, you will need to configure your environment. 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.9.1' . #('default') .
BootstrapRepositoryDirectory } }
```

`#BootstrapExistingConfigurationList`
A collection of configurations in your repository, which 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 upgrade configuration script

The following script is an example of an GsDevKit upgrade customization.

**Example 5.1 Example GsDevKit upgrade setup script**

```
set user DataCurator pass swordfish
login
run
UserGlobals
   at: #BootstrapRepositoryDirectory
   put: GsPackageLibrary getMonticelloRepositoryDirectory.
true
%
run
UserGlobals
   at: #BootstrapApplicationLoadSpecs
   ifAbsent: [
      UserGlobals
         at: #BootstrapApplicationLoadSpecs
         put: {
            { 'ConfigurationOfGLASS' . '1.0-beta.9.1' .
            #('default') . BootstrapRepositoryDirectory } .
         }.
      ].
true
%
commit
logout
```

## Perform the Upgrade

The GsDevKit 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, as described above.

`upgradeSeasideImage [-c <tempObjCacheSize>] [-s <stoneName>] [-u <gemstoneUser>] [-p <password>]`
`-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.**
`-u <gemstoneUser>` specifies the GemStone user name, if seaside was installed as a user other than DataCurator. If not used, defaults to DataCurator.
`-p <password>` the password of the GemStone user that installed seaside. If not used, defaults to swordfish.

For example,

```
% $GEMSTONE/seaside/bin/upgradeSeasideImage -s stoneName331
```

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

The script should complete with the message:

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

## Load your Application Code

After upgrade has successfully completed, load your application code.

## Complete the Upgrade Process

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

Chapter

# 6 Configuring GBS for GemStone/S 64 Bit

The GemStone/S 64 Bit v3.3.1 server requires a compatible version of GBS; versions of GBS earlier than 8.1 cannot be used with GemStone/S 64 Bit v3.3.1 or later.

Versions of GemBuilder for Smalltalk (GBS) that are compatible with GemStone/S 64 Bit v3.3.1 do not support Smalltalk client applications running on Solaris/x86, AIX, or Macintosh.

GemStone/S 64 Bit v3.3.1 supports client Smalltalk/GBS applications running with VisualWorks on Solaris/SPARC, Linux, and Windows, and with VA Smalltalk running on Windows. For a table of all supported GBS and client Smalltalk platforms, see the *GemStone/S 64 Bit Release Notes* for v3.3.1.

For instructions for updating GBS clients that are running on Solaris/SPARC or Linux, see the *GemStone/S 64 Bit Installation Guide* for that platform; for GBS clients running on Windows, see the *GemStone/S 64 Bit Windows Client Installation Guide*.