

# BigMemory Go Operations Guide

Innovation Release

Version 4.3.4

April 2017

This document applies to BigMemory Go Version 4.3.4 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2010-2017 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

## Table of Contents

|                                    |          |
|------------------------------------|----------|
| <b>Logging</b> .....               | <b>5</b> |
| SLFJ Logging.....                  | 6        |
| Recommended Logging Levels.....    | 6        |
| <b>Shutting Down Ehcache</b> ..... | <b>7</b> |
| About Shutdown.....                | 8        |
| ServletContextListener.....        | 8        |
| The Shutdown Hook.....             | 8        |
| Dirty Shutdown.....                | 9        |



# 1 Logging

---

- SLFJ Logging ..... 6
- Recommended Logging Levels ..... 6

---

## SLFJ Logging

---

BigMemory Go uses the SLF4J logging facade, so you can plug in your own logging framework. The following information pertains to Ehcache logging. For information about SLF4J in general, refer to the [“SLF4J website”](#).

With SLF4J, users must choose a concrete logging implementation at deploy time. The options include Maven and the download kit.

### Concrete Logging Implementation use in Maven

The maven dependency declarations are reproduced here for convenience. Add *one* of these to your Maven POM.

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-jdk14</artifactId>
  <version>1.5.8</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>1.5.8</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.5.8</version>
</dependency>
```

### Concrete Logging Implementation use in the Download Kit

The slf4j-api jar is in the kit along with the BigMemory Go jars so that, if the app does not already use SLF4J, you have everything you need. Additional concrete logging implementations can be downloaded from [“SLF4J website”](#).

---

## Recommended Logging Levels

---

BigMemory Go seeks to trade off informing production-support developers of important messages and cluttering the log. ERROR messages should not occur in normal production and indicate that action should be taken.

WARN messages generally indicate a configuration change should be made or an unusual event has occurred. DEBUG and TRACE messages are for development use. All DEBUG level statements are surrounded with a guard so that no performance cost is incurred unless the logging level is set. Setting the logging level to DEBUG should provide more information on the source of any problems. Many logging systems enable a logging level change to be made without restarting the application.

---

## 2 Shutting Down Ehcache

---

|                                |   |
|--------------------------------|---|
| ■ About Shutdown .....         | 8 |
| ■ ServletContextListener ..... | 8 |
| ■ The Shutdown Hook .....      | 8 |
| ■ Dirty Shutdown .....         | 9 |

---

## About Shutdown

---

BigMemory Go is shut down through the Ehcache API. Note that Hibernate automatically shuts down its Ehcache CacheManager.

The recommended way to shutdown BigMemory Go is:

- To call `CacheManager.shutdown()`
- In a web app, register the Ehcache ShutdownListener

Though not recommended, you can also register a JVM shutdown hook.

---

## ServletContextListener

---

Ehcache provides a `ServletContextListener` that shuts down the `CacheManager`. Use this to shut down Ehcache automatically, when the web application is shut down. To receive notification events, this class must be configured in the deployment descriptor for the web application. To do so, add the following to `web.xml` in your web application:

```
<listener>
  <listener-class>
    net.sf.ehcache.constructs.web.ShutdownListener</listener-class>
</listener>
```

---

## The Shutdown Hook

---

The `CacheManager` can optionally register a shutdown hook. To do so, set the system property `net.sf.ehcache.enableShutdownHook=true`. This will shut down the `CacheManager` when it detects the Virtual Machine shutting down and it is not already shut down.

Use the shutdown hook when the `CacheManager` is not already being shutdown by a framework you are using, or by your application.

**Note:** Shutdown hooks are inherently problematic. The JVM is shutting down, so sometimes things that can never be null are. Ehcache guards against as many of these as it can, but the shutdown hook should be the last option to use.

The shutdown hook is on `CacheManager`. It simply calls the shutdown method. The sequence of events is:

- Call `dispose` for each registered `CacheManager` event listener.
- Call `dispose` for each `Cache`.

Each `Cache` will:

- Shutdown the `MemoryStore`. The `MemoryStore` will flush to the `DiskStore`.

- Shutdown the DiskStore. If the DiskStore is persistent ("localRestartable"), it will write the entries and index to disk.
- Shutdown each registered CacheEventListener.
- Set the Cache status to shutdown, preventing any further operations on it.
- Set the CacheManager status to shutdown, preventing any further operations on it.

The shutdown hook runs when:

- A program exists normally. For example, when `System.exit()` is called, or when the last non-daemon thread exits.
- The Virtual Machine is terminated, e.g., CTRL-C. This corresponds to `kill -SIGTERM pid` or `kill -15 pid` on Unix systems.

The shutdown hook will not run when:

- The Virtual Machine aborts.
- A SIGKILL signal is sent to the Virtual Machine process on Unix systems, e.g., `kill -SIGKILL pid` or `kill -9 pid`.
- A `TerminateProcess` call is sent to the process on Windows systems.

## Dirty Shutdown

---

If Ehcache is shutdown dirty, all in-memory data will be retained if BigMemory Go is configured for restartability. For more information, see "Configuring Fast Restart" in the *Configuration Guide* for BigMemory Go.