



---

**GEMFIRE**®  
ENTERPRISE

*Release Notes*

Version 5.5

March 2008

Send comments on this manual to [docs@gemstone.com](mailto:docs@gemstone.com)

---

## INTELLECTUAL PROPERTY OWNERSHIP

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

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

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

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

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

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

## COPYRIGHTS

This software product, its documentation, and its user interface Copyright © 1997-2008, GemStone Systems, Inc. All Rights Reserved by GemStone Systems Inc.

Java Software technologies Copyright © 1994-2000 Sun Microsystems, Inc. All rights reserved. Sun Java Runtime Environment Copyright © 2004-2008 Sun Microsystems, Inc. All rights reserved.

Trove Log4J Copyright © 1999 The Apache Software Foundation. All rights reserved. The Trove library is licensed under the Lesser GNU Public License, which is included with the distribution in a file called LICENSE.txt. PrimeFinder and HashFunctions classes in Trove © Copyright 1999 CERN - European Organization for Nuclear Research.

ILOG JViews copyright © 1996-2001 ILOG S.A. All rights reserved.

JavaGroups copyright © 1999-2004 Free Software Foundation, Inc.

GNU Trove copyright © 2001-2004 Eric D. Friedman. The PrimeFinder and HashFunctions classes in Trove are copyright © 1999 CERN - European Organization for Nuclear Research. Copyright © 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

MX4J project (<http://mx4j.sourceforge.net>). Copyright © 2001-2004 by the MX4J contributors. All rights reserved.

Copyright © 2004 Commons Modeler. All rights reserved.

Copyright © 2000 Cees de Groot. All Rights Reserved. Contributions are Copyright © 2000 by their associated contributors.

Copyright © 1993-95 Ioi Kim Lam. Copyright © 1996 Expert Interface Technologies.

Copyright © 1994 Hewlett-Packard Company

Copyright © 1996,97 Silicon Graphics Computer Systems, Inc. Copyright © 1997 Moscow Center for SPARC Technology.

Copyright © 1998-2003 Daniel Veillard. All rights reserved.

Jgroups © 2001, 2002 [www.jgroups.org](http://www.jgroups.org)

Antr © 2005-2007, Terence Parr, Copyright © 2003-2007, Terence Parr All rights reserved.

## PATENTS

GemFire is protected by U.S. patent 6,360,219. Additional patents pending.

## TRADEMARKS

GemStone, GemFire, GemFire Enterprise, and the GemStone logo are trademarks or registered trademarks of GemStone Systems, Inc. in the United States and other countries (trademark application pending for GemFire).

UNIX is a registered trademark of The Open Group in the U. S. and other countries.

Linux is a registered trademark of Linus Torvalds.

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

SUSE is a registered trademark of SUSE AG.

Sun, Sun Microsystems, Solaris, Forte, Java, Java Runtime Edition, JRE, and other Java-related marks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

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

Microsoft, Windows, and Visual C++ are registered trademarks of Microsoft Corporation in the United States and other countries.

ILOG is a registered trademark and JViews is a trademark of ILOG S.A.

Exolab is a registered trademark of ExoLab Group.

IBM and developerWorks are registered trademarks of IBM Corporation.

W3C is a registered trademark of the World Wide Web Consortium.

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

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

---

## Product Overview

GemFire Enterprise version 5.5 is a new release of the distributed caching system from GemStone. GemFire Enterprise is a high-performance, distributed operational data management infrastructure that sits between your clustered application processes and back-end data sources to provide very low-latency, predictable, high-throughput data sharing and event distribution.

GemFire harnesses the memory and disk resources across a network to form a real-time data fabric or grid. By primarily managing data in memory, GemFire enables extremely high-speed data sharing that turns a network of machines into a single logical data management unit - a data fabric.

GemFire is used for managing operational data - Unlike a traditional centralized disk-based database management system used for managing very large quantities of data, GemFire is a real-time data sharing facility specifically optimized for working with operational data needed by real-time applications - it is the “now” data, the fast-moving data shared across many processes and applications. It is a layer of abstraction in the middle tier that collocates frequently-used data with the application and works with back-end databases behind the scenes.

This document includes the following sections:

- ▶ [Product Overview \(page 3\)](#)
- ▶ [Upgrading to Version 5.5 \(page 4\)](#)
- ▶ [New Features and Product Changes in 5.5 \(page 6\)](#)
- ▶ [Product Implementation Note \(page 11\)](#)
- ▶ [Known Issues / Other \(page 12\)](#)
- ▶ [Bugs Fixed \(page 14\)](#)

For detailed information on using GemFire Enterprise to develop applications, please refer to the programming guides and programming API documentation included with the product. For detailed information on administering GemFire Enterprise systems, see the *GemFire Enterprise System Administrator's Guide*. For detailed information on programming with GemFire Enterprise, see the *GemFire Enterprise Developer's Guide*.

To contact GemStone Technical Support:

- ▶ on the web: <http://support.gemstone.com>
- ▶ by e-mail: [support@gemstone.com](mailto:support@gemstone.com)
- ▶ by phone: 800/243-4772 or 503/533-3503

---

## Upgrading to Version 5.5

The migration from GemFire Enterprise 5.1 may require minor changes. It requires a brief downtime of your system—a 5.5 system member cannot connect to a running system with a different version number. These instructions assume you have read these release notes and are familiar with the feature changes described here.

Your applications that were compiled against version 5.1 will continue to work with the new version, but some default behavior has changed, as noted in this document.

Data that was persisted to disk in version 5.1 can be automatically recovered in version 5.5.

*Data persisted to disk in versions prior to 5.1 can not be automatically recovered in version 5.5. If you need to recover persisted data from versions older than 5.1, contact GemStone technical support for assistance.*

If you have a client/server installation, see [Client connectivity issues, Live Server Monitor timing out servers \(page 12\)](#).

Follow these steps to migrate your development and production systems to the new version of GemFire. Of course, you must thoroughly test your development system with the new version before migrating your production system.

1. If you have not installed GemFire Enterprise version 5.5, do so now. For information, see the installation chapter of the *GemFire Enterprise System Administrator's Guide*. GemFire Enterprise 5.5 is installed as a complete product, rather than as a modification to a prior version.
2. Make any changes to your programs required for version 5.5. To begin, carefully review the changes documented in [New Features and Product Changes in 5.5 on page 6](#). In particular, note the following property changes. For some of these, the default setting is changed.

Property	Change described at
region configuration partition-attributes local-max-memory	<a href="#">Improved Bucket Balancing in Partitioned Regions on page 7</a>
server configuration notify-by-subscription	<a href="#">Change to Server's Default notify-by-subscription Setting on page 8</a>
member configuration mcast-flow-control	<a href="#">Change to Member's Default mcast-flow-control Setting on page 8</a>
member configuration member-timeout	<a href="#">Change to Maximum member-timeout Setting on page 8</a>
member configuration conserve-sockets	<a href="#">Change to Implementation of conserve-sockets=false on page 8</a>

You may need to make `cache.xml`, `gemfire.properties`, or API changes for compatibility with the new version or to retain your application behavior.

This release contains updated online GemFire Java documentation. We recommend that you make full use of the online documentation for your modifications.

3. Recompile your Java applications against the `gemfire.jar` in this version of the product.
4. Stop all members of the system running with the prior version.
5. Point all member sessions to the new installation of GemFire Enterprise 5.5. You may wish to use the `bin/setenv` scripts in the new product installation to assist with this. For information, see the

---

installation chapter of the *GemFire Enterprise System Administrator's Guide* and the getting started chapter of the *GemFire Enterprise Developer's Guide*.

6. Restart all system members according to your usual procedures.

---

## New Features and Product Changes in 5.5

This new version of the product includes improved partitioned region performance and bucket balancing, added security features, active continuous querying of server caches by clients, and durable client subscriptions to server regions. This lists includes all product modifications for the 5.5 release.

### Product Change Highlights

This list links to the highlights of the product changes for version 5.5. The lists that follow cover these and all other changes:

- ▶ [Security - Authentication and Authorization Services \(page 9\)](#)
- ▶ [Continuous Querying for Client/Server Installations \(page 7\)](#)
- ▶ [Durable Subscriptions for Client/Server Installations \(page 7\)](#)

### Partitioned Regions

- ▶ [Improved Bucket Balancing in Partitioned Regions \(page 7\)](#)
- ▶ [Improved Primary Balancing for Partition Regions \(page 7\)](#)
- ▶ [Reduced Garbage Generation in Partition Regions \(page 7\)](#)

### Client/Server Installations

- ▶ [Continuous Querying for Client/Server Installations \(page 7\)](#)
- ▶ [Durable Subscriptions for Client/Server Installations \(page 7\)](#)
- ▶ [Change to Server's Default notify-by-subscription Setting \(page 8\)](#)

### Cache and Region

- ▶ [Improved Commit Conflict Resolution for Transactions \(page 8\)](#)
- ▶ [Improvements to the DistributedLockService \(page 8\)](#)
- ▶ [Updated Cache Memory Overhead Figures \(page 8\)](#)

### System Member

- ▶ [Change to Member's Default mcast-flow-control Setting \(page 8\)](#)
- ▶ [Change to Maximum member-timeout Setting \(page 8\)](#)
- ▶ [Change to Implementation of conserve-sockets=false \(page 8\)](#)

### General

- ▶ [Security - Authentication and Authorization Services \(page 9\)](#)
- ▶ [New Class for Handling System Failures \(page 9\)](#)
- ▶ [Cache XML DTD File Sorted Alphabetically \(page 9\)](#)
- ▶ [Removed Unused Script setenv.csh \(page 9\)](#)
- ▶ [Documentation Changes \(page 9\)](#)
- ▶ [Installation Changes \(page 10\)](#)

---

## Improved Bucket Balancing in Partitioned Regions

Bucket balancing is now more consistent between members hosting a partitioned region. The system uses the `Region` configuration setting `partitioned-region local-max-memory` to indicate the overall load balance for data storage between members. For example, if one member sets a partitioned region's `local-max-memory` to twice the amount set by another member, the distributed system uses twice the memory in the first member as in the second for the region's data. This is done regardless of how little memory is used for the partitioned region.

Before this release, the setting was used only to send a warning when the memory used for the region in the member's heap exceeded the limit indicated. The `local-max-memory` attribute is documented in the region attributes listing for `partition-attributes` in the chapter *Configuring the Cache* in the *GemFire Enterprise Developer's Guide*.

## Improved Primary Balancing for Partition Regions

The locations chosen for primary storage of partitioned region entries are better balanced in this release, which allows your listeners to fire more evenly across the system.

## Reduced Garbage Generation in Partition Regions

The amount of garbage generated by partitioned regions is reduced in this release. You may notice better performance due to this.

## Continuous Querying for Client/Server Installations

Client applications can now define queries to be run continuously in their servers' caches. These continuous queries (CQ), execute on new server cache events, sending the client any changes the events cause in the query results. On the client side, user-programmed listeners receive and process the query events. The functionality uses GemFire Enterprise querying, event framework, and highly available client/server messaging mechanisms. When initializing a continuous query, the client can also receive a one-time full query result set. For details, see the *Continuous Querying* chapter in the *GemFire Enterprise Developer's Guide*. See also the Javadocs listings for `QueryService` and for all interfaces and classes beginning with `Cq` in the `com.gemstone.gemfire.cache.query` and `com.gemstone.gemfire.cache.util` packages.

## Durable Subscriptions for Client/Server Installations

Clients can specify interest lists and continuous queries as *durable* in the server. Anytime the clients are disconnected, the servers queue up events for these durable lists and queries. When clients reconnect, the servers replay the saved events. This allows your clients to drop and come back without losing any events. Durable interest list subscriptions are configured at the region level, so the client can pick the specific regions to keep active. Similarly, you can specify which of your continuous queries are durable. A timeout allows the server to drop clients that remain disconnected for too long a time. For more information, see *Configuring Durable Messaging* in the client/server advanced topics chapter of the *GemFire Enterprise Developer's Guide*. See also the Javadocs listings for `durable-client-id` and `durable-client-timeout` in `DistributedSystem`, the `readyForEvents` and `close` methods in `Cache`, the methods for registering and unregistering interest in `BridgeWriter` and `Region`, and the continuous query Javadocs listed above.

---

## Change to Server's Default notify-by-subscription Setting

The default value for the GemFire server property `notify-by-subscription` is changed to `true`. The old value was `false`. For information on distributed client/server configuration, see the client/server chapters in the *GemFire Enterprise Developer's Guide*.

## Improved Commit Conflict Resolution for Transactions

GemFire cache transactions behavior has been changed to better manage commit conflicts. Transactional operations are now enabled with repeatable reads. This allows the transaction management system to verify that a change has not taken place between the time the operation retrieved an object and the time the update to the object is committed to the cache. For details, see the information on cache transactions repeatable reads in the *GemFire Enterprise Developer's Guide*.

## Improvements to the DistributedLockService

The `DistributedLockService` has been improved to be more stable generally and to provide more reliable high availability. Additionally, local destruction of the `DistributedLockService` now causes all threads waiting to acquire locks to immediately throw a `LockServiceDestroyedException`.

The `DistributedLockService` is used for distributed locking in the GemFire Enterprise caching system. Among other things, it provides automated locking for global regions, partitioned regions, and gateways. It is also available for application use in the `com.gemstone.gemfire.distributed` package.

## Updated Cache Memory Overhead Figures

The cache memory overhead estimates have changed due to added functionality and general improvements to the product. If you use these figures to calculate caching requirements, you will want to review the new figures in *Managing Memory* in the *Run-Time Operations* chapter of the *GemFire Enterprise System Administrator's Guide*.

## Change to Member's Default mcast-flow-control Setting

The default value for GemFire property `mcast-flow-control byteAllowance` is changed to 1048576. The old value was 3500000. The `mcast-flow-control` settings are `DistributedSystem` properties that are usually configured in the `gemfire.properties` file. For information on distributed system configuration, see *System Configuration* in the *GemFire Enterprise System Administrator's Guide*.

## Change to Maximum member-timeout Setting

The maximum allowable setting for GemFire property `member-timeout` is changed to 600,000. The old value was 60,000. The `member-timeout` is a `DistributedSystem` setting that is usually configured in the `gemfire.properties` file. For information on distributed system configuration, see *System Configuration* in the *GemFire Enterprise System Administrator's Guide*.

## Change to Implementation of conserve-sockets=false

To improve performance, socket use has been changed for members and threads with `conserve-sockets=false`. In prior versions, each thread that connected to another member used a dedicated sending socket, but shared receiving sockets with other threads. Now each of these threads uses both a

---

dedicated sending socket and a dedicated receiving socket. This change, while improving performance, uses more of the system's available file descriptors for the sockets and may require you to fine-tune your use of the `conserve-sockets` setting. For more information, see *Controlling Socket Use* in the *Monitoring and Tuning Your Applications/Performance Controls* section of *System Configuration* in the *GemFire Enterprise Developer's Guide*

## Security - Authentication and Authorization Services

GemFire Enterprise can now operate in a secure mode to prevent unauthorized clients or members from participating in a secure GemFire system. GemFire security can also prevent clients from performing unauthorized operations. A secure distributed system disallows malicious intrusions, and also blocks unauthorized test systems from inadvertently accessing its cache. GemFire security is implemented as an optional plug-in framework that augments any existing authentication and entitlement management infrastructure. It can be configured through the `gemfire.properties` file, by passing the `Properties` object in the joining member's `DistributedSystem.connect` call, or by specifying security properties on the command line for the member's start script. For details, see the *Security* chapter in the *GemFire Enterprise System Administrator's Guide*.

## New Class for Handling System Failures

GemFire Enterprise now provides a system failure management class `SystemFailure` in the `com.gemstone.gemfire` package. This class helps you manage catastrophic failure in your distributed system, particularly in your JVM. For more information, see the online Javadocs for this class.

## Cache XML DTD File Sorted Alphabetically

For ease of use, the ordering of the elements listings in the cache configuration DTD file `cache5_5.dtd` is alphabetical. The DTD files for older versions of the product are not changed.

## Removed Console from Product

The console has been removed from the product.

## Removed Unused Script `setenv.csh`

The environment configuration script for `csh` and `tcsh` shells has been removed from the product. Also, instructions for setting your environment in `csh` or `tcsh` shells have been removed from the manuals.

## Documentation Changes

This release includes updated Javadocs and product manuals, *GemFire Enterprise System Administrator's Guide* and *GemFire Enterprise Developer's Guide*. In addition to general changes, the *GemFire Enterprise System Administrator's Guide* has been reorganized to make it easier to use and the installation instructions have been moved out according to *Installation Changes*. The documentation is located on the GemStone support web site at <http://www.gemstone.com/docs/5.5.0/product/docs/html> and can be accessed directly on the web site or through the index page in the `GemFire productDir/docs` directory, where `productDir` is the GemFire Enterprise installation directory.

---

## Installation Changes

The installation has been changed for ease of use. The product installation is now provided in a self-extracting .jar file for all supported platforms. In addition, the installation instructions have been moved out of the *GemFire Enterprise System Administrator's Guide* and into a text file `INSTALL.txt`, which accompanies the .jar file.

---

## Product Implementation Note

Gemfire Enterprise provides access to data at in-memory speeds. It also provides reliable notifications, high availability of data and notifications, and highly performant synchronous and asynchronous persistence and overflow to disk. To provide these features, Gemfire servers rely on using system memory to initialize the data in a server, to store and dispatch updates to clients, and to buffer disk updates before they are flushed periodically to disk in asynchronous mode. All of these activities consume transient memory in the system. Initializing a region from another node, providing redundancy for notifications till they are reliably dispatched to clients, and buffering disk updates before they are flushed to disk all consume finite amounts of transient memory, over and above the memory used for data storage.

While Gemfire is very efficient in pooling and reclaiming memory when not in use, the application designers and architects must do capacity planning to ensure that the server never runs out of memory or experiences periods of low performance due to spikes in transient memory usage. The server can be started with more memory using the standard `-Xmx` and `-Xms` memory settings. For information on this, see *System Performance* in the chapter *Monitoring and Tuning Your Distributed System* in the *GemFire Enterprise System Administrator's Guide*. Most of the parameters that dictate the use of transient memory in the system are governed through `java` command-line and GemFire Enterprise parameters that you can tune before your applications go into production. To optimize the performance of your system for your specific configuration and processing needs, your Gemfire applications architects and designers should take into account the memory needs of your applications during the transition to the Gemfire Enterprise Data fabric. You can also engage Gemfire solution architects to help with this process.

---

## Known Issues / Other

This section lists known issues in the GemFire Enterprise product and its supported platforms.

### GemFire Enterprise

#### Client connectivity issues, Live Server Monitor timing out servers

If your client applications experience connectivity problems with the servers and the client logs report read timeouts in the Server Monitor monitoring Live Connections, make these changes to your client and server configurations and restart your clients and servers:

- ▶ In the clients, set the `useLiveServerMonitorThread` parameter to false for all `BridgeClient`, `BridgeLoader`, and `BridgeWriter` configurations.
- ▶ In the servers, set the `BridgeServer maximum-time-between-pings` parameter to 0 (zero).

For more information on these settings, see the *Connection Health Management* section in *Client/Server Advanced Topics* of the *GemFire Enterprise Developer's Guide*.

#### Bug notes webpage

To see the list of known issues in the GemFire Enterprise product, click on your product version at [http://support.gemstone.com/gemfire/learning\\_center/bug\\_notes/index.htm](http://support.gemstone.com/gemfire/learning_center/bug_notes/index.htm). You must log in to reach the bugnotes page. If you do not have a user account, click on New User Registration to create one.

#### Bug in Apache derby

Tests run with Apache derby, release 10.2.1.6-src, uncovered a problem with the database connection pool management. This problem occurs when many threads are contending for the database connections. After a connection has been used once and returned to the pool, any attempt to use it again finds the connection to be nulled out. A bug has been filed with Apache at this location: <http://issues.apache.org/jira/browse/DERBY-2142>.

#### Weblogic Server and UTF Encoding

Some Weblogic servers are unable to process XML files with a UTF encoding specified. To disable the UTF encoding, make sure the `cache.xml` file specifies `<?xml version="1.0"?>` and not `<?xml version="1.0" encoding="UTF-8"?>`.

#### EPoll Based Selectors for Server Thread Pool Management

The server's thread pool management, introduced in this version, uses the selector provided with the JDK. You can use your own selector by supplying it to the VM at the java command line invocation, like this:

```
-Djava.nio.channels.spi.SelectorProvider=<your Selector>
```

GemFire has only been tested with the default selectors provided with the supported JDKs.

The new epoll-based selector available on Linux 2.6 kernels should improve scalability and performance on servers with hundreds or thousands of connections. The specification is `sun.nio.ch.EPollSelectorProvider`. For more information on this, see [http://java.sun.com/j2se/1.5.0/ReleaseNotes.html#150\\_10](http://java.sun.com/j2se/1.5.0/ReleaseNotes.html#150_10).

#### Diagnosing and Correcting Slow Startup with BEA JRockit

If you use the BEA JRockit JVM, you might experience slow startup that you can fix by changing some of the VM's configuration options. The slow startup is especially visible in client/server installations,

---

where slow server startup may cause failures in clients' `registerInterest` operations. For instance, if the bridge server is launched using the BEA JRockit JVM without any hot caching of compiled methods, the first time that a bridge client attempts a `registerInterest` call, the bridge server may respond too slowly.

One possible workaround is to increase the `readTimeout` property for the `BridgeClient`, `BridgeLoader`, and `BridgeWriter` in the client region. The `readTimeout` is the amount of time (in milliseconds) the client side waits for a response from a bridge server. For more information on the `readTimeout` property, see the Client/Server chapters of the *GemFire Enterprise Developer's Guide*.

The BEA website provides a number of recommendations for diagnosing and correcting slow startup problems at [http://edocs.bea.com/jrockit/geninfo/diagnos/slow\\_start.html](http://edocs.bea.com/jrockit/geninfo/diagnos/slow_start.html). These recommendations include supplying an `optfile` using the `-Djrockit.optfile=property` to launch your JVM. For more information, see <http://edocs.bea.com/jrockit/geninfo/diagnos/crash.html#wp1000606>.

## **Acrobat Reader 6.0 links always go to first page**

There is a known problem with Acrobat Reader 6.0 that causes links between pdf files to always go to the first page. If you encounter this problem while viewing the documentation pdfs, you can try these steps. Open Acrobat 6, open the `Edit` menu, select `Preferences`, click on `General`, then uncheck the "Open cross-document links in same window" option.

---

## Bugs Fixed

This section lists the bugs fixed in the GemFire Enterprise product since version 5.1.

### Fixed in GemFire 5.5

#### Possible inconsistency in regions with DataPolicy.REPLICATE

When a new Region with DataPolicy.REPLICATE was created, it was possible that it would miss concurrent updates being applied to other Regions with the same data policy. The window of time that this could have occurred in was miniscule, but it was observed in at least one of the v5.5 regression tests. (#38724)

#### NullPointerException received during transaction commit on servers

If a client had registered interest on a server that did transactional operations (JTA or GemFire), the server thread committing the transaction may have received a NullPointerException with a stack similar to the following:

```
[severe 2008/03/25 16:39:49.471 PDS <Thread-4> nid=0x5f1ba8]
CacheClientProxy[identity(client1(:loner):1:6364ecbb:ClientName1,connection=2);
port=4623; primary=true]: Exception occurred while attempting to add
message to queue
java.lang.NullPointerException at
com.gemstone.gemfire.internal.jta.TransactionImpl.registerSynchronization
(TransactionImpl.java:197 at
com.gemstone.gemfire.internal.cache.LocalRegion.getJTAEnlistedTX(LocalReg
ion.java:5173 at
com.gemstone.gemfire.internal.cache.LocalRegion.put(LocalRegion.java:1098
at
com.gemstone.gemfire.internal.cache.AbstractRegion.put(AbstractRegion.jav
a:188 at
com.gemstone.gemfire.internal.cache.ha.HARegionQueue.put(HARegionQueue.ja
va:386 at
com.gemstone.gemfire.internal.cache.tier.sockets.CacheClientProxy$Message
Dispatcher.enqueueMessage(CacheClientProxy.java:1724 at
com.gemstone.gemfire.internal.cache.tier.sockets.CacheClientProxy.process
Message(CacheClientProxy.java:674 at
com.gemstone.gemfire.internal.cache.tier.sockets.CacheClientNotifier.deliv
er(CacheClientNotifier.java:693 at
com.gemstone.gemfire.internal.cache.tier.sockets.CacheClientNotifier.noti
fyClients(CacheClientNotifier.java:376 at
com.gemstone.gemfire.internal.cache.BridgeServerImpl.notifyClients(Bridge
ServerImpl.java:257 at
com.gemstone.gemfire.internal.cache.LocalRegion.notifyBridgeClients(Local
Region.java:3750 at
com.gemstone.gemfire.internal.cache.LocalRegion.invokePutCallbacks(LocalR
egion.java:3716)
```

If this occurred, the transaction would have been partially applied to the local heap. It would not, however, have been distributed to other server VMs. The workaround was to avoid transactions on servers. (#38709)

#### Newly arrived cache member may confuse existing elder

There was a race condition in how an elder and a new member communicated. If an elder discovered a new cache member before it had processed its first membership view, the new member may have failed

---

to respond to a query from the elder. This would cause the elder, and consequently the entire distributed system, to hang. The workaround was to kill the elder to allow a new elder to be selected. (#38690)

### **Servers and clients may report each other's failures incorrectly**

If a server crashed unexpectedly, a client that it was connected to may have reported the failure in a number of misleading ways, including indicating a corrupted message stream from that process.

Likewise, if a client crashed unexpectedly, a server that it was connected to may have reported the failure in a number of misleading ways, including indicating a corrupted message stream from that process. (#38555)

### **Hang when a disk region entry is destroyed and an operation on that entry is attempted**

After a disk region entry was destroyed, an operation on the same entry would hang. (#38467)

### **ConcurrentModificationException from AckReceiverWindow can corrupt cache member**

There was a race condition in the processing of distributed system traffic that could have caused a `ConcurrentModificationException` during the processing when peers departed the system. The workaround was to stop and restart your cache member. This error was thrown from `AckReceiverWindow`, as noted in the cache server logs. (#38455)

### **PartitionedRegion get unnecessarily serializes/de-serializes byte array values**

A `get` operation on a partitioned region always performed a serialization/de-serialization operation even when the value was a byte array. In the byte array case, the operation caused unnecessary memory garbage (memory allocation/de-allocation) and extra CPU cycles. (#38308)

### **gemfire.jar does not correctly undeploy from an EJB server**

Once an EJB application server was connected to a distributed system, it may not have been able to correctly undeploy `gemfire.jar`. (#38294)

### **Cache hangs on startup due to other member closing its Cache but not its DistributedSystem**

Cache hangs on startup due to other member closing its `Cache` but not its `DistributedSystem`. The workaround was to make sure your members close the `DistributedSystem` when closing the `Cache`. (#38242)

### **Cache member hangs while creating a region**

Under certain high availability conditions, a cache member may have hung while attempting to recover a region from another member that had crashed. In order for this to happen, a cache member needed to be creating a local copy of a region at the same time that another cache member crashed. The workaround was to kill and restart the hung cache member. (#38235)

### **Transactions that include multiple regions cause LRU problems in remote caches**

This problem occurred when a GemFire transaction included many regions, like this:

```
txmgr = cache.getCacheTransactionManager()
txmgr.begin();
region1.put("a", "one");
region2.put("b", "two");
region3.put("c", "three");
txmgr.commit();
```

In addition, two or more of the regions had to have LRU eviction configured in VMs that were remote to the VM where the transaction originated. In this scenario, the LRU mechanism in the remote VMs did

---

not consistently evict the proper number of entries. The problem did not affect eviction in the VM where the transaction originated. The workaround was to only include a single region in a transaction or only have one of the regions configured with LRU behavior. (#38188)

### **Failed initial image creation may throw AssertionError**

If you closed your cache while initializing the data in a distributed region, you may have ended up with a faulty `AssertionError` in the system logs, which you could safely ignore. Now the error does not appear. (#38152)

### **Partitioned Region initialization takes too long**

Partitioned Region initialization was taking over 60 seconds for 45 VMs to create/join a Partitioned Region. In comparison, a replicated region takes ~580ms in the same configuration. (#38041)

### **Partitioned region puts will throw NoClassDefFoundError on remote partitioned region members if the value class is not on the classpath**

A partitioned region put failed with `NoClassDefFoundError` if the value Object's class was not on the classpath of every member that configured data storage for that partitioned region. The only members that should require the class are those that need the value in object form (for example the member that actually does a `get` to read the value or the member with a `CacheListener` that calls `getNewValue`). The workaround was to add the value Object's class to all members that defined the partitioned region. (#38013)

### **conserve-sockets=false may run out of sockets**

It was possible for a member to run out of sockets when using `conserve-sockets=false`. This could have been caused by threads that owned their own sockets having a short lifetime and new threads being created quickly that also owned their own sockets. The workaround was to call `DistributedSystem.releaseThreadsSockets` before a thread's life came to an end. This might have been done from a `finally` block on the thread's run method. (#38011)

### **Improper handling of instances of Java VirtualMachineError**

When a Java virtual machine sends an instance of `VirtualMachineError` to a thread, it has indicated that it has broken the fundamental programming contract and can no longer be trusted.

The most common instance of this is `OutOfMemoryError`, which is sent to one `Thread` somewhere in the JVM. All other `Threads` are effectively suspended at their next attempt to allocate memory until either a) enough memory becomes available, or b) the original thread that was signaled disappears.

In prior versions, GemFire did not properly handle `VirtualMachineErrors`. This improper handling manifested in numerous bugs in the system. GemFire now has a cooperative mechanism by which a cache member can reliably recuse itself from the distributed system when a `VirtualMachineError` occurs. Notice, however, that in order for this to be reliable, your applications must also correctly trap and signal `VirtualMachineError` when they are thrown. See the Javadocs for `SystemFailure` for details on this new API. (#37942)

### **Partition region creation requires a partition-attributes element or a PartitionAttributes setting in the API**

Setting the region `data-policy` to `PARTITION` should cause a region to be created as a partitioned region, but it didn't. The `data-policy` setting was accurately reported, but it did not cause the region to partition its data. For the region to be created as a partitioned region, the region attributes had to have a `partition-attributes` element in the `cache.xml` or a `PartitionAttributes` setting through the API. You did not need to set any non-default partition attributes settings, just use the partition attributes. (#37905)

---

### **Installation Paths with spaces prevent the Native Client from installing correctly**

While installing the Native Client on Microsoft Windows, if a path was specified that contained spaces, for example `C:\Program Files\GemStone\GemFire`, the msi installer that was invoked from `setupWin32_gf51.exe` would fail causing a dialog box to appear detailing the msi command line syntax. After dismissing this dialog the installation would continue and appear to have succeeded. The GemFire Enterprise installation was OK, but the Native Client installation was not: only the `native_client.msi` and a few HTML files were installed for the Native Client. The workaround was to uninstall the product to clean up the system, then reinstall the product into a path without spaces. (#37803)

### **Partitioned Region data storage is skewed**

When quickly loading data into a partitioned region, the number of buckets from one data store to the next may have varied as much as 100%. Due to the seemingly random allocation of buckets, this required that all VMs for the partitioned region have up to two times the required memory for actual storage. Increasing the maximum number of buckets exaggerated the problem. This issue may have caused scaling limitations for storage, network bandwidth and CPU resources. (#37795)

### **Member that is kicked out of the distributed system may not realize it and continue to operate, eventually causing hangs**

If you were using the `gemfire.useFD` or `gemfire.FD_TIMEOUT` system properties to select the alternative GemFire UDP-heartbeat failure detection mechanism, a member could have been forcibly disconnected from the distributed system if it did not respond quickly enough to "are you alive" messages. The `member-timeout` and `gemfire.FD_TIMEOUT` settings control this disconnect timeout. In version 5.1.x of GemFire, the disconnected member did not realize that it had been kicked out of the system and continued to try to operate. Eventually other members may have hung.

We only observed this with the alternate failure detection mechanism and only under significant CPU load. However, setting a short `member-timeout` period may have exacerbated the problem and caused it to happen more easily.

The workaround was to set a reasonably long `member-timeout` period when using `gemfire.useFD`, or set the timeout period with the deprecated `gemfire.FD_TIMEOUT` system property. (#37779)

### **Eviction regions with short life span have unexpected memory and cpu consumption**

`Region.close`, `localDestroyRegion`, and `destroyRegion` on a region with eviction configured did not close the `LRUStatistics` object. If a large number of region destroys were done, this could have caused the statistic sampler to consume an entire CPU and the unclosed statistic object to consume around 100 bytes of memory. The workaround was to avoid giving your LRU regions a short life span. This could have been accomplished by using `clear` instead of doing a `destroy` and `create`. To prevent the CPU consumption, you might have disabled statistic sampling. (#37743)

### **LRU Region eviction may happen early or late**

The LRU limit was not strictly complied with when doing evictions. Evictions might have been done slightly early (causing less space to be used than was specified) or slightly late (causing more space to be used than was specified). The workaround was to set `-Dgemfire.STRIPED_STATS_DISABLED=true` to get the older version of statistics that caused strict compliance to the eviction limit. (#37736)

### **Lock request continues to wait after lock service has been destroyed**

Active lock requests may have continued to wait for a reply from a remote grantor even after the local instance of that `DistributedLockService` had been destroyed. When the threads that were waiting for lock request finally returned they would throw `LockServiceDestroyedException` as expected.

---

These calls to lock request only hung up until the `waitTimeMillis` had been exhausted. The default `waitTimeMillis` for Global regions is 60 seconds, so threads that were waiting for a lock in order to put may have waited up to 60 seconds before throwing `RegionDestroyedException` (internally `LockServiceDestroyedException` rethrown as `RegionDestroyedException` by the Region API).

Disconnecting from the `DistributedSystem` caused the waiting threads to return immediately.

Workarounds were using `waitTimeMillis` to specify how long the lock request would wait and disconnecting from the `DistributedSystem` to cause the waiting threads to return immediately. (#37688)

### **AssertionError after starting and stopping servers or gateway hubs many times without disconnecting from the distributed system**

If you started and stopped a bridge server or gateway hub approximately 32K times without disconnecting from the distributed system, an `AssertionError` would occur. The workaround was to disconnect from the distributed system after disconnecting a bridge server or gateway hub. (#37668)

### **Hang in partitioned region operations when frequently destroying cache or region**

Members that closed their `Cache` without stopping their `DistributedSystem` and members that closed or locally-destroyed a partitioned region may have caused other processes to hang. The hung processes had threads that were `waitingForPrimaryMember` or retrying cache operations. The problem was due to a race condition in internal storage allocation. It was most likely to reproduce with higher redundant-copies settings. There was no workaround but you might have compensated by avoiding using `close` and `localDestroyRegion` on partitioned regions and by avoiding closing the `Cache` without also stopping the `DistributedSystem`. (#37604)

### **Split brain in partitioned regions**

There was a rare race condition that could have occurred in assigning an internal identifier to a partitioned region. The condition caused the system to assign more than one identifier to a single partitioned region, with some processes using one identifier and some using another. Because of this, the processes with one identifier did not recognize operations performed on the `Region` by the processes using the other identifier and vice-versa. (#37549)

### **Member hangs and/or large log files seen following Cache close or stopping of BridgeServers or Gateways**

Stopping bridge servers, gateways, or partitioned regions had to be done by disconnecting from `DistributedSystem`. Specifically, bridge servers, gateways, and partitioned regions attempted to interrupt internal threads that may have been using the `DistributedLockService`. If the timing was just right, the lock service may have caused other members to hang or produce very large log files.

Cache close attempts to stop gateways and bridge servers as well as the partitioned region high availability threads, so closing the cache may have also caused this problem to occur.

Some indications that this had occurred include statements in the log such as:

```
Grantor is still initializing
Grantor creation was aborted but grantor was not destroyed
```

The presence of these in the log meant a thread was interrupted while using the `DistributedLockService` and the member had to be disconnected from the `DistributedSystem`.

The workaround was to use `DistributedSystem.disconnect` to safely stop gateways, bridge servers, and partitioned regions and close the cache without problems in the `DistributedLockService`. The problem only occurred if the member that interrupted the lock service thread remained in the system. (#37380)

---

### **Bridge Clients may become inconsistent in the face of region destruction**

In a client VM, if a region was destroyed and then recreated, there was a race condition that may have caused the client to start losing updates from the server. This was characterized by a log message of `NullPointerException` in the `CacheClientUpdater` thread. The workaround was to avoid region destruction, if at all possible, and, if the message occurred in the logs, to stop and restart your client. (#36932)

### **PartitionedRegion get performance degradation**

Performing a `get` on a partitioned region was three times worse in version 5.1 than in 5.0.1. (#36921)

### **Partitioned Region destruction may indicate incorrect region name**

If a partitioned region was destroyed while operations were active on it, a GemFire system internal name (of an individual bucket) may have been returned to the destroy operation instead of the name of the region itself. (#36905)

### **Hangs during PartitionedRegion entry operations**

This problem could have occurred when one member was performing entry operations, like `put`, in a partitioned region while another member was performing an orderly shutdown of the same region (using either distributed system `disconnect`, `cache close`, `region close`, or `region destroy`). In this situation, it was possible for the member performing the entry operations to hang. (#36798)

### **System member fails to shut down**

Under certain high stress conditions, when a system member tried to close its cache or distributed system, the operation may have failed to complete. This could have caused the entire distributed system to "freeze." The workaround was to kill the failed (hung) process. This allowed the other system members to resume normal operation. (#36754)

### **System member shutdown generates messages**

After a distributed system or a cache started to shut down, you may have seen error messages in the log, usually with stacks. These messages usually indicated that a `ShutdownException` or `ThreadInterruptedException` had occurred. However, other errors, including `ExceptionInInitializerError`, `NullPointerException`, `ConcurrentModificationException`, or `AssertionError` were also possible. If shutdown was underway, the shutdown would not be compromised, even though the error messages were an annoyance. (#36656)

### **Client Region.get() returns null unexpectedly during failover to new server**

In rare circumstances we saw a client cache return a null from a `region get(key)` operation when the cache was failing over to a new server cache. (#36645)

### **A single client disconnect/reconnect from a distributed system may result in multiple pairs of memberLeft/memberJoined events**

When a client disconnected and reconnected, you would expect each bridge client membership listener to receive one `memberLeft` event and one `memberJoined` event. Instead, it received `N memberLeft` events and `N memberJoined` events, where `N` was the number of connections established with the server side by that client. (#35910)

---

**Region attribute mutation may not be received by remote caches**

If a region attribute mutation was done right after a region was created, it was possible for the mutation to not be communicated to remote caches. This could have created situations where distributed messaging was not sent to remote caches. (#34343)