

A large, light gray diamond shape is centered on the page, composed of several smaller triangles. The word "GEMFIRE" is written in a bold, black, sans-serif font across the middle of this diamond, with a small orange triangle above the letter 'I'. A thin green horizontal line is positioned below "GEMFIRE". Below "GEMFIRE", the word "ENTERPRISE" is written in a smaller, black, sans-serif font.

**GEMFIRE**<sup>®</sup>  
ENTERPRISE

*Release Notes*

Version 5.7

September 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.

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.

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.

Commons Modeler Copyright © 2004 Commons Modeler. All rights reserved.

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

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)

Antlr © 2005, 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.

Exolab is a registered trademark of ExoLab Group.

IBM, AIX, 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 5.7 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 5.7 \(page 4\)](#)
- ▶ [New Features and Product Changes in 5.7 \(page 13\)](#)
- ▶ [Product Implementation Note \(page 18\)](#)
- ▶ [Known Issues / Other \(page 19\)](#)
- ▶ [Bugs Fixed \(page 21\)](#)

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 information on administering GemFire Enterprise systems, see the *GemFire Enterprise System Administrator's Guide*. See the *GemFire Enterprise Developer's Guide* for information on programming with GemFire Enterprise.

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 5.7

The migration from GemFire Enterprise 5.5 may require minor changes. It requires a brief downtime of your system because a 5.7 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.

### Note for Data Persisted to Disk

Data persisted to disk in versions prior to 5.7 cannot be automatically recovered in version 5.7. Cache data is serialized for persistence to disk, so the compatibility of persisted data across Gemfire versions depends on the serialization format that is used. To accommodate more user types, in 5.7 we expanded the object type identifier in the serialized form from a `byte` to an `int`. This change gives you greater flexibility in the types of objects you can persist to disk, but requires additional steps for the upgrade to 5.7. The 5.7 version expects an `int` identifier and does not automatically read data persisted in pre-5.7 versions, with the `byte` identifier.

If you need to recover persisted data from prior versions, please contact GemStone technical support for assistance.

### Upgrade Steps

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 5.7, do so now. For information, see the installation chapter of the *GemFire Enterprise System Administrator's Guide*. GemFire Enterprise 5.7 is installed as a complete product, rather than as a modification to a prior version. The Java JRE runtime environment is not bundled with GemFire Enterprise 5.7, so if you previously used the bundled JRE you will need to install and configure an appropriate JDK or JRE to comply with GemFire requirements and your unique system needs.
2. Make any changes to your programs required for version 5.7. To begin, see [Backward Compatibility of the Bridge\\* Client Interfaces on page 4](#) and [Upgrading to the New Client/Server Functionality on page 5](#). Then review the changes documented in [New Features and Product Changes in 5.7 on page 13](#). You may need to make `cache.xml`, `gemfire.properties`, or API changes for compatibility with the new version or to retain your application behavior. It is recommended that you make full use of the updated online Java API 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 5.7. For information, see the installation chapter of the *GemFire Enterprise System Administrator's Guide* and the programming chapter of the *GemFire Enterprise Developer's Guide*.
6. Restart all system members according to your usual procedures.

## Backward Compatibility of the Bridge\* Client Interfaces

The `Bridge*` client interfaces have been deprecated and replaced with `Pool` and related configuration settings. If you continue to use the `Bridge*` client interfaces, you will no longer be able to change them at runtime using the `AttributesMutator`. You can still change other plug-ins to the `CacheWriter` and `CacheLoader`.

---

## Upgrading to the New Client/Server Functionality

This section is for developers who already have GemFire client/server installations. It outlines the changes you need to make to switch to the functionality described in *Server Scalability and Ease of Use* on page 14. For complete information on implementing dynamic server discovery, balancing, and logical grouping, see the online Java documentation and the product manuals. These topics cover the changes:

- ▶ [Changes to Distributed System Configuration \(page 5\)](#)
- ▶ [Overview of Changes to the cache.xml and API \(page 5\)](#)
- ▶ [Server Configuration \(page 6\)](#)
- ▶ [Client Configuration: Replacement of BridgeClient, BridgeLoader, BridgeWriter \(page 6\)](#)
- ▶ [Example Configuration Changes \(page 9\)](#)
- ▶ [Special Configuration Requirements \(page 11\)](#)

### Changes to Distributed System Configuration

To use the dynamic server discovery and balancing, you need to run server locators in your server distributed system. If you are already using locators for peer discovery, they will automatically run as server locators in the new version. If you are not using locators, you need to add them to your server system. For information, see *Configuring Member Discovery and Communication in the GemFire Enterprise System Administrator's Guide*. Your servers and locators must have the same locator location information.

You then list your locators in your client `Pool` configuration so the clients can find the locators. Your clients will no longer be configured with explicit server location information.

### Overview of Changes to the cache.xml and API

This section describes the changes to the API and XML for the new dynamic server management.

These are the changes to the packages used for client/server:

- ▶ **com.gemstone.gemfire.cache**—`Bridge*` client interfaces are not plugged into `CacheLoader` and `CacheWriter` anymore. Pools, with expanded bridge client functionality, are defined at the cache level and assigned to regions in the region attributes. Change made to region attributes to add pool name.
- ▶ **com.gemstone.gemfire.cache.client**—New package with `Pool` and related interfaces and classes.
- ▶ **com.gemstone.gemfire.cache.server**—New package with server load probe interfaces and classes.
- ▶ **com.gemstone.gemfire.cache.util**—`BridgeServer` has been changed to `CacheServer`. `CacheServer` is expanded to support dynamic discovery, logical grouping, and load probe. Preexisting functionality is essentially unchanged for pool use (but there are additions for the overflow to disk of the server-to-client event queues). `Bridge*` client interfaces deprecated in favor of `Pool`. `BridgeMembership` interfaces are unchanged.

The following sections details the changes to the `cache.xml`. The API has corresponding changes. For more information, see the online Java documentation and the manuals.

---

## Server Configuration

For preexisting functionality, the configuration is basically unchanged but there are new configuration options to support dynamic discovery and load balancing. This table lists the changes and new functionality.

**Table 1.1 Changes to Server Configuration**

Old bridge-server	New cache-server	Description (for new properties)
	<group>	List of server groups this cache server belongs to. All servers belong to a global group.
	<client-subscription>	Queue disk overflow settings. This is not required for pool configuration. For information on this setting, see this property in the Client/Server Basics chapter of the <i>GemFire Enterprise Developer's Guide</i> .
	<custom-load-probe>	Application plug-in for providing server load information to the locators. Used to replace the default GemFire load probe.
	bind-address	The ip address or host name that this server will listen on.
	hostname-for-clients	String representing the ip address or host name that server locators will tell clients that this server is listening on.
	load-poll-interval	How often the server's load probe should run.
DEFAULT_THREAD_LIMIT	DEFAULT_MAX_THREADS	No change except the name. The old name works but is deprecated.
DEFAULT_CONNECTION_LIMIT	DEFAULT_MAX_CONNECTIONS	No change except the name. The old name works but is deprecated.

## Client Configuration: Replacement of BridgeClient, BridgeLoader, BridgeWriter

The Bridge\* interfaces are deprecated - use Pool and related configurations.

*You can still use these Bridge\* client interfaces, but you can no longer modify them after region creation. The AttributesMutator no longer allows you to add or remove Bridge\* interfaces in the plug-ins for CacheLoader and CacheWriter.*

With the new Pool configuration, you define named pools at the cache level and then assign the pools to your client regions by setting the new region attribute, pool-name.

The `pool` configuration is supported by the `cache5_7.dtd`. The `Bridge*` client interfaces were application plug-ins to the `CacheLoader` and `CacheWriter` and had no DTD support.

This is a side-by-side of the old `Bridge*` configuration properties and their replacements in the `pool` configuration, followed by a listing of the new configuration options in the `pool`. For more information on all properties, see the online Java documentation.

**Table 1.2 Migration of Property Settings from Bridge\* Client to Client Pool.**

Bridge*	<pool>	Description
<code>allowableServerTimeouts</code>		Removed. The job of monitoring server health has been moved to the locators.
<code>allowableServerTimeoutPeriod</code>		Removed. The job of monitoring server health has been moved to the locators.
<code>clientAckInterval</code>	<code>subscription-ack-interval</code>	No change.
<code>connectionsPerServer</code>	<code>min-connections</code>	Minimum number of pool connections to keep open. The new configuration does not specify number of connections <i>per server</i> . Instead, the client just specifies the connection count and the pool connects to the best available servers. See also the new property <code>max-connections</code> (page 8).
<code>endpoints</code>	<code>&lt;locator&gt;</code>	Old: list of servers to connect to. New: list of locators to connect to for dynamic server information. You can still use the server list if you want to by adding <code>&lt;server&gt;</code> elements instead of <code>&lt;locator&gt;</code> elements. With this, you do not have dynamic server discovery, balancing, or grouping.
<code>establishCallbackConnection</code>	<code>subscription-enabled</code>	Requests event streaming from server.
<code>LBpolicy</code>	<code>thread-local-connection</code>	The new configuration only chooses whether to have connections stick to threads until released. In the old version, you chose between sticky and non-sticky, and between round-robin and random server selection. The new functionality uses load information from the servers for the load balancing portion of this.
<code>messageTrackingTimeout</code>	<code>subscription-message-tracking-timeout</code>	No change.
<code>readTimeout</code>	<code>read-timeout</code>	No change.
<code>redundancyLevel</code>	<code>subscription-redundancy</code>	No change.
<code>retryAttempts</code>	<code>retry-attempts</code>	No change.

Bridge*	<pool>	Description
retryInterval	ping-interval	This has a dual purpose: 1. How often to communicate with the server to show the client is alive. 2. How long to wait after failing to create a connection before trying again.
socketBufferSize	socket-buffer-size	No change.
useLiveServerMonitorThread		Removed. The job of monitoring server health has been moved to the locators.

This table lists the new pool configuration properties. The defaults should work for your upgrade.

**Table 1.3 New Configuration in Client Pool**

<pool>	Description
free-connection-timeout	How long a thread should wait for a free pool connection. Threads have to wait when all open connections are in use and the max-connections has been reached, so the pool can't open any more connections.
idle-timeout	How long a connection can remain open and unused before being closed. The connection is not closed if it would take the total open connections below the min-connections limit.
load-conditioning-interval	The amount of time, in milliseconds, a pool connection can remain open before being eligible for silent replacement to a less-loaded server.
max-connections	Maximum number of pool connections to open at one time. See also <a href="#">min-connections (page 7)</a> .
name	Pool name - required for assigning this pool to the client regions.
server-group	Optional logical server group name to use.
statistic-interval	Interval at which to send client statistics to the server for monitoring.

**Table 1.4 New Configuration in Client Region**

<region-attributes>	Description
pool-name	This makes the region a client region and assigns it to the named pool. The name must correspond to a name setting in an existing pool.

---

## Example Configuration Changes

This section lists old and new configuration files for a simple client/server installation.

### Server and Locator Startup

For both, the server uses a locator for peer discovery.

This is the `gemfire.properties` file for locator and server:

```
mcast-port=0
locators=localhost[41111]
```

If you were already using locators, you do not need to change anything to use them as server locators in the new system. If you need to add locators, see *Configuring Member Discovery and Communication in the GemFire Enterprise System Administrator's Guide*. The old locator and new are started with the same command:

```
gemfire start-locator -port=41111
```

By default, the new locator acts as a server locator as well as a peer locator. If you wanted to explicitly set these options, you would enter this command:

```
gemfire start-locator -port=41111 -server=true -peer=true
```

### Server Cache Configuration

For many installations, the server cache configuration does not need to change. This is the old configuration file from our examples.

#### Example 1.1 Old Server cache.xml

---

```
<cache>
  <cache-server port="40404" notify-by-subscription="true"/>
  <region name="root">
    <region-attributes scope="distributed-ack">
    </region-attributes>
  <region name="my_region">
    <region-attributes scope="distributed-ack" data-policy="replicate">
      <cache-loader>
        <class-name>cacheRunner.StringLoader</class-name>
      </cache-loader>
      <cache-listener>
        <class-name>cacheRunner.LoggingCacheListener</class-name>
      </cache-listener>
    </region-attributes>
  </region>
</region>
</cache>
```

---

There are two cases where the server configuration may need additions. These are listed after the basic client configuration changes, under *Hostname For Clients* on page 11 and *Server Grouping* on page 11.

---

## Client Cache Configuration

Your client configuration will change. This is the old client `cache.xml`, which lists the specific server to connect to and uses a `BridgeClient` in the region attributes cache loader for server access. The area that will be changed is in **bold**:

### Example 1.2 Old Client `cache.xml`

---

```
<cache>
  <region-attributes id="clientAttributes" scope="local">
    <cache-loader>
      <class-name>com.gemstone.gemfire.cache.util.BridgeClient</class-name>
      <parameter name="endpoints">
        <string>server=lucy:40404</string>
      </parameter>
      <parameter name="establishCallbackConnection">
        <string>>true</string>
      </parameter>
    </cache-loader>
  </region-attributes>
  <region name="root">
    <region-attributes scope="distributed-no-ack" statistics-enabled="true">
    </region-attributes>
    <region name="my_region">
      <region-attributes refid="clientAttributes" statistics-enabled="true"/>
    </region>
  </region>
</cache>
```

---

The new client `cache.xml` shown in the following example has no server-specific information. Instead, it names the locator to connect to for server information. This configuration provides dynamic server discovery and load balancing. Another server can be added to the server distributed system and this client would use it for a balanced share of its client/server operations.

### Example 1.3 New Client `cache.xml`

---

```
<cache>
  <pool name="client" subscription-enabled="true">
    <locator host="localhost" port="41111"/>
  </pool>
  <region-attributes id="clientAttributes" pool-name="client" scope="local"/>
  <region name="root">
    <region-attributes scope="distributed-no-ack" statistics-enabled="true">
    </region-attributes>
    <region name="my_region">
      <region-attributes refid="clientAttributes" statistics-enabled="true"/>
    </region>
  </region>
</cache>
```

---

---

## Special Configuration Requirements

This section lists specific configuration changes that your installation may require to continue operating as it did with older versions of the product.

### Hostname For Clients

In the old system, you specifically gave the client the servers' hostnames and ports. In the new system, when the server connects to the locator it tells the locator the host and port where it is listening for client connections. If the host the server uses by default is one that the client can't translate into an IP address, the client will have no route to the server's host and won't be able to find the server. For this situation, you must supply the server with an alternate hostname for client use. The server hostname that you were using in the `Bridge*` configuration `endpoints` list should work. This is how the `<cache-server>` line in [Example 1.1 on page 9](#) would look with this specification:

```
<cache-server port="40404" notify-by-subscription="true"
hostname-for-clients="lucy"/>
```

### Server Grouping

Some client configurations manually group servers. In the old configuration, one set of endpoints was assigned to the `Bridge*` configuration for some regions, and another set of endpoints assigned to the `Bridge*` configuration for other regions. This is how the old client configuration would have looked:

#### Example 1.4 Old Client `cache.xml` With Server Grouping

---

```
<region-attributes id="clientAttributes1" scope="local">
  <cache-loader>
    <class-name>com.gemstone.gemfire.cache.util.BridgeClient</class-name>
    <parameter name="endpoints">
      <string>server=lucy:40404</string>
      <string>server=ethel:40404</string>
    </parameter>
    <parameter name="establishCallbackConnection">
      <string>true</string>
    </parameter>
  </cache-loader>
</region-attributes>
<region-attributes id="clientAttributes2" scope="local">
  <cache-loader>
    <class-name>com.gemstone.gemfire.cache.util.BridgeClient</class-name>
    <parameter name="endpoints">
      <string>server=ricky:40404</string>
      <string>server=fred:40404</string>
    </parameter>
    <parameter name="establishCallbackConnection">
      <string>true</string>
    </parameter>
  </cache-loader>
</region-attributes>
. . .
<region name="my_region1">
  <region-attributes refid="clientAttributes1" statistics-enabled="true"/>
</region>
<region name="my_region2">
  <region-attributes refid="clientAttributes2" statistics-enabled="true"/>
</region>
```

---

---

In the new configuration, you do the grouping in the server and client configurations so the clients do not need to know exactly which servers belong to which groups. The servers identify themselves as belonging to a group with this `<cache-server>` configuration change:

**Example 1.5 New Server `<bridge-server>` Specification With One Server Group**

---

```
<cache-server port="40404" notify-by-subscription="true">
  <group>group1</group>
</cache-server>
```

---

A server can belong to multiple groups by just adding more `<group>` elements. All servers belong to a global, no-name server group, so if your clients do not specify a group then they have access to all servers in the system.

The client then specifies the group to use in its pool configuration:

**Example 1.6 New Client `<pool>` Specification With Server Grouping**

---

```
<pool name="client1" subscription-enabled="true" server-group="group1">
  <locator host="localhost" port="41111"/>
</pool>
<pool name="client2" subscription-enabled="true" server-group="group2">
  <locator host="localhost" port="41111"/>
</pool>
<region-attributes id="clientAttributes1" pool-name="client1" scope="local"/>
<region-attributes id="clientAttributes2" pool-name="client2" scope="local"/>
```

---

---

## New Features and Product Changes in 5.7

This version of the product includes new dynamic server discovery, load balancing, and grouping functionality. The product also provides new capabilities for handling network partitioning situations (split brain), and support for backward compatibility. Other new features include a new GFMOn system monitoring tool, disk overflow for client queues on the server, and queue conflation on a per-client basis. Complete descriptions of the new 5.7 product features are included in the following list.

### Product Change Highlights

This list links to the new 5.7 product highlights. The lists that follow cover these and all other changes:

- ▶ [Server Scalability and Ease of Use \(page 14\)](#)
- ▶ [Network Partitioning Detection and Recovery \(page 14\)](#)
- ▶ [Support for Backward Compatibility \(page 14\)](#)
- ▶ [New GFMOn System Monitoring Tool \(page 15\)](#)

### Client/Server Installations

- ▶ [Server Scalability and Ease of Use \(page 14\)](#)
- ▶ [Overflow to Disk for Client Subscription Queues \(page 15\)](#)
- ▶ [Client-Level Configuration of Subscription Queue Conflation \(page 15\)](#)

### Cache and Region

- ▶ [Custom Expiration for Region Entries \(page 15\)](#)
- ▶ [Change to DataSerializable \(page 15\)](#)
- ▶ [Bulk Operations \(page 15\)](#)

### System Member

- ▶ [Network Partitioning Detection and Recovery \(page 14\)](#)
- ▶ [E-Mail Notification for JMX Agent \(page 16\)](#)

### General

- ▶ [Support for Backward Compatibility \(page 14\)](#)
- ▶ [Change in Default Locator Behavior \(page 16\)](#)
- ▶ [Support for AIX Operating System \(page 15\)](#)
- ▶ [Common Installer for All Platforms \(page 15\)](#)
- ▶ [Key Size Added for AES and Blowfish Authentication Algorithms \(page 16\)](#)
- ▶ [Compatibility with IPv6 Addresses \(page 16\)](#)
- ▶ [Runtime Java JRE Now User-Specified \(page 17\)](#)
- ▶ [VSD Performance Monitor Availability \(page 17\)](#)
- ▶ [Documentation Changes \(page 17\)](#)

---

## Server Scalability and Ease of Use

GemFire now offers dynamic server discovery, load balancing, and server grouping to improve scalability and ease of use in client/server installations. The new functionality uses GemFire locators, running in the server distributed system and acting as *server locators* to clients. These are the primary changes and new features:

- ▶ **Server name change**—Bridge Server has been changed to Cache Server.
- ▶ **Dynamic server discovery**—The GemFire server locator utility dynamically tracks server processes and directs clients to new servers, giving clients indirection from explicit server information. Clients only need to know how to connect to the locator services. They do not need to know where servers are running or how many servers are available at any time.
- ▶ **Server groups**—You can assign your servers to logical groups that your clients can refer to in their connection configurations. For example, you might use groups to manually partition your data, with one group of servers hosting one set of data and another hosting another set. Or, you might use a group to direct all database-centric traffic to the subset of servers that are directly connected to a backend database. Servers can belong to multiple groups. Your clients only need to specify the group to use and are isolated from having to know which servers belong to which groups.
- ▶ **Server load balancing**—The GemFire server locator tracks current load information for all servers, directing new client connections to the servers with the least load. GemFire provides a default load probe for your servers, which you can replace with your own customized plug-in.
- ▶ **Server connection conditioning**—Client connections can be configured to transparently time out and be replaced with new connections, which allows overall server use to be rebalanced after new servers are started. This helps speed conditioning in situations like when you scale your server group, and following recovery from server crashes and other downtimes.
- ▶ **New client connection use configuration**—You can now cap the number of connections your client can open, and you can configure connections to time out when left unused for a period of time.

## Network Partitioning Detection and Recovery

You can now configure a set of cache servers to keep them from entering into a network partition configuration in the presence of a network failure or when members lose the ability to communicate to each other. When a network or communication failure occurs, there is a possibility that the system could split into multiple network partitions, which, if not monitored, could result in operations in the different partitions causing data inconsistencies or a forced disconnect.

## Support for Backward Compatibility

GemFire Enterprise 5.7 is now backward compatible-ready, with compatibility extending over two major versions of the product. You will be able to mix version 5.7 clients with version 6.0 or 7.0 cache servers, and successfully connect more recent client versions with older cache server versions.

For loosely-coupled distributed systems connected using GemFire gateways, a version 5.7 server in one distributed system operates as expected while communicating over a gateway to another distributed system populated with servers up to version 7.0. Conversely, a newer server will function normally over a gateway communicating with servers older by up to two major versions.

If a connecting server or client is more recent than two major versions, a message describes the appropriate corrective action.

---

## New GFMon System Monitoring Tool

The new GFMon tool continuously monitors a distributed system, acquiring detailed information about the state of the deployed GemFire system and analyzing the results. GFMon is not distributed with GemFire, so contact GemStone Technical Support to acquire GFMon.

## Support for AIX Operating System

GemFire now supports the IBM AIX open standards-based UNIX operating system. See the *Product Installation* chapter in the *GemFire Enterprise System Administrator's Guide* for details about AIX.

## Common Installer for All Platforms

GemFire now installs using a common installer for Windows, Linux, Solaris, and AIX.

## Overflow to Disk for Client Subscription Queues

You can now configure the server to overflow its client subscription queues to disk if they reach capacity in memory. For more information, see *Limiting Subscription Queue Size* in the *Client Server Advanced Topics* chapter of the *GemFire Enterprise Developer's Guide*.

## Client-Level Configuration of Subscription Queue Conflation

You can now override the server queue conflation configuration on a per-client basis by setting the `conflate-events` property in the client's `gemfire.properties`. The three valid settings are `server`, which uses the server settings, `true`, which conflates everything sent to this client, and `false`, which does not conflate anything sent to this client. For more information, see *System Properties in the gemfire.properties File* in the *GemFire Enterprise System Administrator's Guide*.

## Custom Expiration for Region Entries

You can now configure custom expiration on individual region entries, based on entry content, on a per-entry basis. For more information, see *CustomExpiry*, under *Application plug-ins*, in the *Configuring the Cache* chapter in the *GemFire Enterprise Developer's Guide*.

## Change to DataSerializable

For custom data objects, you can use the GemFire `com.gemstone.gemfire.DataSerializable` interface to help create your own data serializers. The interface may also be used to replace standard data serialization for better performance. In addition, the `DataSerializable` interface has been modified. The constructor's second arg has been changed from a `byte` to an `int`. Refer to the Javadocs and the *GemFire Enterprise Developer's Guide* for more information.

## Bulk Operations

The new `Region.putAll` method can batch up multiple `put` operations into the cache in a single bulk distributed operation without needing to handle multiple round trips for message distribution.

---

## Change in Default Locator Behavior

The default locator behavior has changed to include the new server locator functionality used for dynamic server discovery, described in [Server Scalability and Ease of Use on page 14](#). For current information on locators, see *Configuring Member Discovery and Communication* in the *GemFire Enterprise System Administrator's Guide*.

If you do not want your locator to act as a server locator, you now must explicitly exclude the behavior. At the command line, do this by adding `-server=false` to your locator startup command. The API for locator startup has a similar new server locator specification.

## Key Size Added for AES and Blowfish Authentication Algorithms

GemFire now provides the ability to specify the key size in the `security-client-dhalgo` property for the AES and Blowfish algorithms to encrypt credentials during client-server security authentication.

## Compatibility with IPv6 Addresses

GemFire now works with IPv6 addresses, but may require a change to configuration settings for use on IPv6 systems. If you have locators specifications in `gemfire.properties` files formatted like this:

```
locators=host-name1:bind-address1[port1],host-name2:bind-address2[port2]
```

you need to change the colon separator ':' between the host-name and bind-address to an "at" sign separator '@', like this:

```
locators=host-name1@bind-address1[port1],host-name2@bind-address2[port2]
```

Colons can't be used to separate host names and IPv6 addresses because an IPv6 address may contain a variable number of colons and hexadecimal numbers.

For example, for compatibility with IPv6, change this `gemfire.properties` specification

```
locators=lucy:10.80.10.80[34455],ricky:10.80.10.80[34455]
```

to this

```
locators=lucy@10.80.10.80[34455],ricky@10.80.10.80[34455]
```

For more information on locator specifications, see *Configuring Member Discovery and Communication* in the *GemFire Enterprise System Administrator's Guide*.

## E-Mail Notification for JMX Agent

The Java Management Extensions (JMX) Agent can now be configured to provide e-mail notification to a list of recipients for stat alerts set in GFMon and for membership change events. For more information, see *Using JMX to Administer GemFire* in the *GemFire Enterprise System Administrator's Guide*.

---

## Runtime Java JRE Now User-Specified

Starting with GemFire Enterprise 5.7, the bundled Java JRE runtime environment is not provided with the product. Now you can choose the optimum JRE version for your unique system requirements. An appropriate JDK or JRE can be downloaded from the Sun Developer Network at <http://java.sun.com>. After installing the JRE you will need to configure your GemFire environment to use it. See the *GemFire Enterprise System Administrator's Guide* for more information.

## VSD Performance Monitor Availability

The Visual Statistics Display (VSD) performance monitoring tool is no longer installed with GemFire. Contact GemStone Technical Support for instructions about acquiring VSD.

## Documentation Changes

This release includes updated Javadocs and product manuals, *GemFire Enterprise System Administrator's Guide* and *GemFire Enterprise Developer's Guide*. The documentation is located on the GemStone support web site at <http://www.gemstone.com/docs/5.7.0/product/docs> 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.

---

## 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 until 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 Member Performance* in the chapter *Monitoring and Tuning the 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*.

#### NIO selector implementation on certain VMs

When a Cache server does not respond to clients even though the VM server is using CPU, client operations will hang and eventually fail after they have timed out if they are not able to connect to another Cache server.

You must set a system property to enable the workaround. You only need to enable the workaround if you:

- ▶ are running on linux
- ▶ have set the CacheServer's max-thread to attribute a value greater than 0
- ▶ are using one of these JDK versions: 1.6.0\_5, 1.6.0\_7

Versions 1.5.0\_15, 1.6.0\_3, 1.6.0\_10 do not require the workaround.

To enable the workaround, set this system property:

```
-DCacheServer.NIO_SELECTOR_WORKAROUND=true
```

#### Bug notes web page

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.

### Linux Requirements for Multicast Settings

On Linux systems, when using multicasting, you must specify different mcast addresses and different mcast ports in order to keep different distributed systems separated. Systems with different addresses but the same port specification may find each other and communicate.

Due to the way Linux interprets RFC 1112, multicast sockets using the same port will receive datagrams from each other even if using different multicast addresses. See these links for more information:

```
https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=231899  
http://bugs.sun.com/bugdatabase/view_bug.do;:YfiG?bug_id=4701650  
http://www.uwsg.iu.edu/hypermail/linux/net/0211.1/0003.html
```

Make sure to select different multicast ports for different distributed systems to keep them isolated from one another.

---

## 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 cache 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 cache 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 cache 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 5.5.

### Fixed in GemFire 5.7

#### **GemFire Installation failed when run with java.util.zip.ZipException with spaces in the directory name**

The GemFire installer did not run correctly with spaces left in the name of the installer directory. The failure came with a stack trace similar to the following:

```
The system cannot find the path specified Exception in thread "main"  
java.util.zip.ZipException: The system cannot find the path specified at  
java.util.zip.ZipFile.open(Native Method) at  
java.util.zip.ZipFile.<init>(ZipFile.java:203) at  
java.util.zip.ZipFile.<init>(ZipFile.java:234) at  
ZipSelfExtractor.extract(ZipSelfExtractor.java:99) at  
ZipSelfExtractor.main(ZipSelfExtractor.java:34)
```

The workaround was to move the GemFire installer jar into a directory without spaces and then rerun it (#38782)

#### **Partitioned Regions did not fire RegionMembershipListener events**

If a `RegionMembershipListener` was added to a partitioned region, the following methods did not fire for the listener:

```
initialMembers afterRemoteRegionCreate afterRemoteRegionDeparture  
afterRemoteRegionCrash
```

There was no workaround for this. (#38719)

#### **SSL communications hung if conserve-sockets=false**

The product hung if SSL communications were configured and the GemFire `conserve-sockets` property was set to false. This was due to a small flaw in the cache's handshake interchange that took place immediately after a SSL socket was connected.

There was no workaround for this. (#38531)

#### **Recycling the DistributedSystem connection while creating a DistributedLockService or Cache resulted in a corrupted DistributedLockService or Cache**

Internal references to the `DistributedSystem` are provided to the `Cache` and `DistributedLockService`. If the `Cache` or a `DistributedLockService` was being created at the same time that another thread disconnected and reconnected to the `DistributedSystem`, it was possible for those features to retain a reference to the previous `DistributedSystem` connection, which could have resulted in unexpected `ShutdownExceptions` or other problems.

The workaround for this was to, first, make sure all other application threads were no longer actively using the GemFire APIs, then disconnect and reconnect to the `DistributedSystem`. (#38356)

#### **Distributed lock requests may have hung after failover if deserialization of lock information failed with BufferUnderflowException**

This affected `DistributedLockService`, partitioned regions, global regions, and gateway hubs. `DistributedLockService` lock requests may have hung waiting for lock information if another member had failed to deserialize lock information while assuming grantor duties. A severe level log statement similar to the following would appear in one of the member logs:

---

```
... [severe 2007/05/27 06:35:24.203 PDT gemfire2_biscuit_337
<P2P message reader for biscuit(336):42137/47098-0x151> nid=0x9aa16bb0]
Error deserializing message java.io.IOException:Could not create an
instance of
com.gemstone.gemfire.distributed.internal.locks.DLockRecoverGrantor
Processor$DLockRecoverGrantorReplyMessage at
com.gemstone.gemfire.DataSerializer.readObject(DataSerializer.java:34 86)
at
com.gemstone.gemfire.internal.tcp.Connection.processNIOBuffer
(Connect ion.java:2553)at
com.gemstone.gemfire.internal.tcp.Connection.runNioReader(Connection.
java:1203)at
com.gemstone.gemfire.internal.tcp.Connection.run(Connection.java:1139)
at java.lang.Thread.run(Thread.java:595) Caused by: java.io.IOException:
Could not create an instance of com.gemstone.gem
fire.distributed.internal.locks.DLockToken at
com.gemstone.gemfire.DataSerializer.readObject(DataSerializer.java:34 86)
at com.gemstone.gemfire.DataSerializer.readObjectArray(DataSerializer.ja
va:2469)at
com.gemstone.gemfire.distributed.internal.locks.DLockRecoverGrantorPr
ocessor$DLockRecoverGrantorReplyMessage.fromData(DLockRecoverGrantorProce
ssor.ja va:308) at
com.gemstone.gemfire.DataSerializer.readObject(DataSerializer.java:34 76)
... 4 more Caused by: java.nio.BufferUnderflowException at
java.nio.Buffer.nextGetIndex(Buffer.java:398) at
java.nio.DirectByteBuffer.get(DirectByteBuffer.java:205) at
com.gemstone.gemfire.internal.tcp.ByteBufferInputStream.read(ByteBuff
erInputStream.java:34) at
java.io.DataInputStream.readInt(DataInputStream.java:353) at
com.gemstone.gemfire.distributed.internal.locks.DLockToken.fromData(D
LockToken.java:486) at
com.gemstone.gemfire.DataSerializer.readObject(DataSerializer.java:34 76)
... 7 more
```

The workaround for this was to disconnect and reconnect the member with the severe log. As soon as other members detected the departure of this member, they would recover and locking would resume. (#36888)

### **Failure while getting the initial image of a persistent disk region left partial data on disk**

This bug affected persistent replicas. If a VM crashed while it is getting the initial image of a persistent replica region, the partial persisted disk files would remain on disk. If the VM was then restarted and there were no cached replicas from which to initialize the region, it was automatically initialized from this partial data set on disk.

The workaround for this was if you detected a crash of a VM with persistent replica regions, you had to decide how to restart the VM based both on the state of available data and on your application needs. If there was a cached replica from which to initialize the region, that may have been your primary source of up-to-date data. If there was no in-memory replica, your latest backup of the persistent region, stored in the backup directory, GFBAK-<backup#>, may have been your most complete source. Or you may have needed to choose to initialize from the partial data set that was left on disk. (#35745)

### **LockNotHeldException: Attempted to unlock in Partitioned Region Lock Service but this thread did not own the lock**

Partitioned region management used the DLock system to coordinate distributed resource creation. If there were failures in the distributed system that required the DLock system to recover, you may have subsequently received `LockNotHeldExceptions` on the partitioned regions operations, get (if a

---

CacheLoader is defined), put, or create. This was not a bug in the DLock system, but a problem with how a partitioned region managed locks.

The workaround for this was to catch the exception and retry the operation. (#35710)

### **DataSerializer did not work with long (>64k) strings**

The `DataSerializer` interface did not handle strings larger than 64 Kilobytes.

The workaround for this was to inhibit `DataSerialization` of the string by placing the string inside a holder that the `DataSerializer` didn't know how to handle. The example code used was as follows:

```
public class holder implements Serializable { public Object content;
public holder(Object content) { this.content = content; } }
region.put("aStringHolder", new holder(bigString)); ... holder
stringHolder = (holder)region.get("aStringHolder"); if (stringHolder !=
null) { System.out.println("found " + stringHolder.content); }
```

The `DataSerializer` then created an `ObjectOutputStream` to serialize the holder and its content (the string). (#35406)

### **In Pure Java mode, cacheserver start could start multiple concurrent cache servers**

When you start a cacheserver process, GemFire checks that no cacheserver is already running in the same working directory. This ensures a distinct status file for each cacheserver. Currently, this check is done using native code. If you run in pure Java mode (without the native library), the check is not performed and you are allowed to start multiple cache servers in the same directory. (For information on pure Java mode, see *Programming with GemFire Enterprise on page 45 of the GemFire Enterprise Developer's Guide*.)

The workaround for this was, if you were using cacheservers and running the product in pure Java mode, you must have taken care not to start more than one cacheserver in the same working directory. (#35244)

