



GEMFIRE[®]
ENTERPRISE

Native Client

Release Notes

Version 2.5.0.1

April 2008

Send your comments about this document to 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 © 1997-2008 GemStone Systems, Inc. All rights reserved by GemStone Systems, Inc. Doxygen © 1997-2004 Dimitri van Heesch.

STLport © 1994 Hewlett-Packard Company, © 1996-1999 Silicon Graphics Computer Systems, Inc., © 1997 Moscow Center for SPARC Technology, © 1999-2002 Boris Fomitchev.

ACE, TAO, CIAO, and CoSMIC (henceforth referred to as "DOC software") are copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University. © 1993-2006, all rights reserved.

PATENTS

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

TRADEMARKS

GemStone, GemFire, and the GemStone logo are trademarks or registered trademarks of GemStone Systems, Inc. in the United States and other countries.

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.

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

ACE, TAO, CIAO, and CoSMIC are trademarks of Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University.

Berkeley DB and Sleepycat are trademarks or registered trademarks of Oracle Corporation.

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

GemStone[®] GemFire[®] Enterprise Native Client 2.5.0.1 Release Notes

To contact GemStone Technical Support:

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

Contents

This document includes the following sections:

- ▶ [Product Overview \(page 3\)](#)
- ▶ [Product Changes in 2.5.0.1 \(page 4\)](#)
- ▶ [New Features and Product Changes in 2.5.0.0 \(page 5\)](#)
- ▶ [Known Issues \(page 6\)](#)

For detailed information about configuring, using and administering the GemFire Enterprise native client, refer to the *GemFire Enterprise Native Client Guide* and the programming API documentation included with the product.

Product Overview

The GemFire Enterprise native client provides access for C++ and Microsoft[®] .NET[™] clients to the GemFire Enterprise distributed system. The native client is written entirely in C++, so its initialization process does not involve the creation of a Java virtual machine. The .NET Framework client overlays the C++ native client and provides native operations for the .NET Framework application developer who needs to access the GemFire Enterprise cache server.

Product Changes in 2.5.0.1

Documentation Changes

For the version 2.5.0.1 product documentation, these release notes and the `README.html` files for the product examples have been updated.

Compatibility with GemFire Enterprise

Version 2.5.0.1 of the native client is compatible only with version 5.5.1 of GemFire Enterprise. It is incompatible with all GemFire versions prior to 5.5.1.

New Features and Product Changes in 2.5.0.0

Documentation Changes

For the version 2.5.0.0 product documentation, these release notes, the online API reference documentation, and the *GemFire Enterprise Native Client Guide* have been updated.

Durable Client Messaging

To prevent client data loss if a client's connection to a cache server is severed, the GemFire Enterprise native client now supports durable messaging as an additional layer of protection. When a client configured for durable messaging disconnects, either normally or abnormally, its cache servers continue to queue the client's messages for a configurable length of time. When the client reconnects, it receives its queued messages so its cache is updated.

The native client can be configured for durable messaging through its `gfcpp.properties` configuration file, or through its C++ and .NET API.

Client Authentication and Authorization

The GemFire Enterprise native client can now be configured so it must submit secure credentials before it can connect to a GemFire cache server. The native client can also be configured so it must submit credentials for authorizing cache server operations. The operation cannot be performed if client authorization is denied by the cache server.

GemFire security prevents unauthenticated clients from connecting to cache servers in the distributed system. By requiring the submission of credentials, the distributed system disallows malicious intrusions and blocks unauthorized test systems from inadvertently accessing its cache.

Security is implemented in GemFire as an optional plug-in framework that augments any existing authentication and entitlement management infrastructure. This implementation gives application administrators the flexibility to delegate additional responsibility for system security to external providers. Sample implementations are also provided that demonstrate additional plug-in security capabilities, such as using a LDAP server for authentication, establishing PKCS encryption for authentication, and configuring XML files for defining authorization rules.

Known Issues

To review known issues for the GemFire Enterprise native client, click your product version at http://support.gemstone.com/gfcpp/learning_center/bug_notes/index.htm. You must log in to access the bug notes page. If you don't have a user account, click New User Registration to create one.

The Native Client License File Must be Requested

Before you can operate the native client, you must request its license file from GemStone Customer Support. The license file (`gfCpPLicense.zip`) does not ship with the native client. When you receive the license file, either place it in the top-level directory for the native client product, or set the `license-file` property in `gfcpp.properties` to point to the file location.

GemFire Enterprise Cache Server Tuning Information

See the *GemFire Enterprise System Administrator's Guide* for detailed information about ways to configure and fine-tune the GemFire cache server to work efficiently with the native client.

Unsupported CreateServerRegion and CreateServerSubregion Methods

The `CreateServerRegion` and `CreateServerSubregion` methods are not supported in this release of the native client. An attempt to use these methods results in an `UnsupportedException` error. A future release will add support for these APIs.

Non-Primitive Key Types Require a Java Class Registered with the Cache Server

For non-primitive key types, a Java class defining the type is needed so the cache server can store the value in the cache. The applications accessing the cache may not involve Java at all, but the Java class still needs to be written and registered with the cache server.

Timeout Can Occur While Registering Interest in a Large Number of Keys

Individually registering interest in a large number of keys can result in a timeout condition. Instead, call `registerAllKeys` if you need to register interest in thousands of keys on the server.

Use a Smart Pointer Instead of a Raw Pointer for User Object Functions

If a raw pointer to a user-defined object is used for `Region`, `Cache`, or other API functions for the native client, the pointer is implicitly converted to a smart pointer. This is disallowed by the compiler for serializable keys and values in the `Region` API, but not in other places. A smart pointer takes ownership of an object, and deletes it when no other object references remain. The following code snippet shows the use of a raw pointer to put an object into the cache using the API:

```
CacheableKeyPtr keyPtr = CacheableKey::create(sKey.c_str());
ExampleObject* newObj = new ExampleObject(sValue);
ExampleObjectPtr newObjPtr = newObj;
ExampleCallback* callbackObj = new ExampleCallback();
m_currRegionPtr->put(keyPtr, newObj, callbackObj); // Compiler error: raw
pointer for keys/values is disallowed by the compiler
m_currRegionPtr->put(keyPtr, newObjPtr, callbackObj); // Error: raw
pointer for other arguments is allowed by the compiler but will not work
as expected
```

In the previous code example, the `callbackObj` object will be deleted by the smart pointer when the `put` call returns if the callback does not keep a smart pointer reference.

To prevent the implicit conversion of a raw pointer to a smart pointer, use a `SharedPtr` object when calling the API for a user-defined object. For example:

```
typedef SharedPtr<ExampleCallback> ExampleCallbackPtr;
...
CacheableKeyPtr keyPtr = CacheableKey::create(sKey.c_str());
ExampleObjectPtr newObjPtr = new ExampleObject(sValue);
ExampleCallbackPtr callbackPtr = new ExampleCallback();
m_currRegionPtr->put(keyPtr, newObj, callbackPtr); // Ok
```

