

DNS Caching in Java Virtual Machines

1.0

Wednesday, May 25, 2005



VeriSign Global Registry Services Proprietary Information

This document is the property of VeriSign Global Registry Services, Inc. It may be used by recipient only for the purpose for which it was transmitted and will be returned upon request or when no longer needed by recipient. It may not be copied or communicated without the prior written consent of VeriSign Global Registry Services.

COPYRIGHT NOTIFICATION

Copyright © 2005, VeriSign, Inc. All rights reserved.

VERISIGN GLOBAL REGISTRY SERVICES PROPRIETARY INFORMATION

This document is the property of VeriSign, Inc. Information contained herein may include trade secrets and confidential information belonging to VeriSign. Unauthorized disclosure without the express written consent of VeriSign, Inc. is prohibited. It may be used by recipient only for the purpose for which it was transmitted and will be returned upon request or when no longer needed by recipient. It may not be copied or communicated without the prior written consent of VeriSign, Inc..

DISCLAIMER AND LIMITATION OF LIABILITY

VeriSign, Inc. has made efforts to ensure the accuracy and completeness of the information in this document. However, VeriSign, Inc. makes no warranties of any kind (whether express, implied or statutory) with respect to the information contained herein. VeriSign, Inc. assumes no liability to any party for any loss or damage (whether direct or indirect) caused by any errors, omissions or statements of any kind contained in this document. Further, VeriSign, Inc. assumes no liability arising from the application or use of the product or service described herein and specifically disclaims any representation that the products or services described do not infringe upon any existing or future intellectual property rights. Nothing herein grants the reader any license to make, use, or sell equipment or products constructed in accordance with this document. Finally, all rights and privileges related to any intellectual property right described in this document are vested in the patent, trademark, or service mark owner, and no other person may exercise such rights without express permission, authority, or license secured from the patent, trademark, or service mark owner.

VeriSign Inc. reserves the right to make changes to any information herein without further notice.

NOTICE AND CAUTION

Concerning U.S. Patent or Trademark Rights

The inclusion in this document, the associated on-line file, or the associated software of any information covered by any patent, trademark, or service mark rights will not constitute nor imply a grant of, or authority to exercise, any right or privilege protected by such patent, trademark, or service mark. All such rights and privileges are vested in the patent, trademark, or service mark owner, and no other person may exercise such rights without express permission, authority, or license secured from the patent, trademark, or service mark owner.

This publication was created using Microsoft® Word 2000 for Windows™ by Microsoft Corporation.

Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation.



VeriSign® Global Registry Services

21345 Ridgetop Circle

Dulles, VA 20166-6503

E-mail: info@verisign-grs.com

Internet: <http://www.verisign-grs.com>

Revision History

Author(s)	Date	Version	Description

Contents

1. Executive Summary	7
2. How DNS Caching is done in Java?	8
3. Managing and Configuring JVM DNS Cache Settings	9
4. Managing and Configuring Alternative JVM DNS Cache Settings	11
5. Differences in Java Virtual Machines	13
6. Conclusion	13

1. Executive Summary

The VeriSign Naming and Directory Services (VNDS) offers registry services through the Shared Registration System (SRS) for the various key top level domain names that VeriSign is in the business of managing. The way registrar applications connect to these registry services is typically using a published domain name. It is critical that all registrar applications use this published domain name instead of the ip address, and also respect the time to live (ttl) setting on the ip address associated with the domain name. The reason for this is VeriSign may change the ip address associated with the domain name to move the services from one data center to another or to balance load across multiple data centers.

VeriSign is aware that many of the registrars use Java as their programming platform. The default behavior of Java is to cache DNS lookups for the lifetime of the Java process, which is contrary to standard DNS behavior and will cause unplanned outages to the customer if the best practices in this document are not followed.

The purpose of this document is to outline this issue and a set of best practices in using DNS as a client in Java applications.

2. How DNS Caching is done in Java?

The `java.net.InetAddress` class has a cache to store successful as well as unsuccessful host name resolutions. The positive and negative caching is there to improve performance. The positive caching is also there to guard against DNS spoofing attacks.

By default, the result of positive host name resolutions is cached forever, because there is no general rule to decide when it is safe to remove cache entries. The result of unsuccessful host name resolution is cached for a very short period of time (10 seconds) to improve performance.

Under certain circumstances where it can be determined that DNS spoofing attacks are not possible, a Java security property can be set to a different Time-to-live (TTL) value for positive caching. Likewise, one can configure a different negative caching TTL value when needed.

Once an application has performed network access (i.e. Socket, URLConnection, parsing of xml document with external references, etc), the DNS settings get cached so any subsequent operation will use the old settings even if the real settings have changed. To reset everything, you have to restart the server since the default setting JVM setting is to cache forever.

For connecting to VeriSign systems, we recommend that registrar applications set a ttl value of 0. Since connections are long lived the overhead of this setting should be negligible.

3. Managing and Configuring JVM DNS Cache Settings

There are two properties that can be used to override the default behavior. These two Java security properties control the TTL values used for positive and negative host name resolution caching. They are:

`networkaddress.cache.ttl`

This property indicates that the caching policy for successful name lookups from the name service. The value is specified as an integer to indicate the number of seconds to cache the successful lookup. The default value of this property is -1, which means the successful DNS lookup value will be cached for ever in the JVM. If the value is set as 0, then it means that it will not cache successful DNS lookup up. Any other positive value indicates that successful DNS lookups will be cached for that many seconds.

`networkaddress.cache.negative.ttl`

This property indicates the caching policy for unsuccessful name lookups from the name service. The value is specified as an integer to indicate the number of seconds to cache the unsuccessful lookup. The default value of this property is 10, which means that unsuccessful DNS lookup value will be cached for 10 seconds in the JVM. If the value is set as 0, then it means that it will not cache successful DNS lookup up. Any other positive value indicates that unsuccessful DNS lookups will be cached for that many seconds.

The above mentioned properties are not regular Java properties, but instead security related properties. Hence these properties can be set by one of 2 ways:

1. Edit the `$JAVA_HOME/jre/lib/security/java.security` by changing the value of the `networkaddress.cache.properties` in the file. The advantage of this solution is that it is a non-programmatic solution. The disadvantage is that since JVMs are global resources, used by multiple applications, the setting may not suite all applications.
2. Use `java.security.Security.setProperty("propertyname", "value")` to programmatically set the property . The disadvantage of this solution is that it is a programmatic solution. The advantage is that other applications using the same JVM are not affected.

For example, `java.security.Security.setProperty("networkaddress.cache.ttl" , "0");`

4. Managing and Configuring Alternative JVM DNS Cache Settings

There are two other properties that can be Sun private system property which corresponds to the two properties discussed in the previous settings. They are:

`sun.net.inetaddr.ttl`

This is a sun private system property which corresponds to `networkaddress.cache.ttl`. It takes the same value and has the same meaning, but can be set as a command-line option. However, the preferred way is to use the security property mentioned above.

`sun.net.inetaddr.negative.ttl`

This is a sun private system property which corresponds to `networkaddress.cache.negative.ttl`. It takes the same value and has the same meaning, but can be set as a command-line option. However, the preferred way is to use the security property mentioned above.

These values can be set by adding `-Dsun.net.inetaddr.ttl=value` and `-Dsun.net.inetaddr.negative.ttl=value` on the command line when starting the JVM. It is also important to note that values are effective only if the corresponding `networkaddress.cache.*` properties are not set.

5. Differences in Java Virtual Machines

The DNS settings of Java Virtual machines outlined in this document are per the Java Specification. However, some implementations of Java Virtual Machines may have

alternate interpretations of this specification, and thus it is the responsibility of the registrars to test the DNS caching and the effects of these settings on their applications.

6. Conclusion

It is critical that VNDS customers using Java understand the default behavior of DNS caching and configure their Java based servers to use a ttl of 0.