October 28, 2004

David J. Byer
Testa, Hurwitz & Thibeault, LLP
125 High Street
Boston, MA 02110-2704

Dear Mr. Byer:

We have received and reviewed your letter of October 31, 2003 on behalf of your client, JBoss Group, LLC ("JBoss"). The Apache Software Foundation ("ASF") takes the matter of your letter seriously and we appreciate your having brought these allegations to our attention. As explained more fully below, we believe that none of the alleged similarities between the Apache Geronimo ("Geronimo") code and the code contained in the JBoss application server constitute a violation of your client's intellectual property rights.

In your October 31 letter there are three named exhibits, A, B, and C, and a fourth alleged similarity that for the purposes of this discussion will be referred to as assertion D.
Our conclusions with respect to each of the exhibits presented in your letter are as follows.

- In Exhibit A, you attached XLevel files from both JBoss and Geronimo and stated that the two files "demonstrate[d] on their face a very high degree of similarity." Both the JBoss and Geronimo code files are derived from the same code taken from an ASF-licensed Apache codebase, the Apache Logging project (formerly Apache Jakarta Log4j). It appears that information required by the ASF license was deleted by your clients. After you confirm our findings, you should advise your clients to ensure that the required attributions and notices for Apache licensed code are reinstated.

- In Exhibit B, you attached PatternParser files. This similarity arises from the same cause as the similarity apparent in Exhibit A. Both files were derived or based on the same code from the ASF-licensed Apache codebase, the Apache Logging project.

- In Exhibit C, you attached InvocationType files and stated that their code "appears to be nearly identical." This similarity is due to the same author licensing code to JBoss under the LGPL and to the ASF under the ASL. Such dual licensing of code is a legitimate and common mechanism in the open source community. We also verified that changes subsequently made by JBoss to the code it received under the LGPL were not incorporated within Geronimo.

1901 Munsey Drive, Forest Hill, MD 21050-2747 fax: +1.410.803.2258

- In Assertion D, you contend that the use of "JBoss-specific payloads" such as "AsIs," "Transient," and "Marshalled" in the Geronimo Invocation file indicates that the Geronimo project must contain JBoss source code. An earlier version of Geronimo did contain an interface that utilized the modifiers "AsIs," "Transient," and "Marshal" in processing the movement of data. However, the modifiers were used in an expression that differed significantly from the JBoss usage of "AsIs," "Transient," and "Payload" in its Invocation implementation. Thereafter, the "AsIs" and "Marshal" modifiers were removed from the Geronimo code, and only the modifier "Transient" was retained for the purpose of allowing the user of an Invocation implementation to declare that data placed into that implementation is transient. This is also a far different expression than that in the referenced JBoss code. The only similarity between the files is one commonly used defined term.

We have thoroughly investigated the claims by JBoss and have not found any infringement of JBoss's intellectual property. Because the ASF is a collaborative, consensus-based community that believes in full disclosure of its activities, our review was undertaken publicly by the ASF and all discussion and results are available via an archive of the Geronimo development list. For your convenience, you may find an archive of the messages at this URL:

http://nagoya.apache.org/eyebrowse/SummarizeList?listName=geronimo-dev@incubator.apache.org.

Further, the summary of the findings is available in our CVS, are included below as Attachment 1, and publicly accessible at this URL:

http://cvs.apache.org/viewcvs.cgi/incubator-geronimo/docs_nopublish/JBoss_20031031.html

It is important to us that this issue is resolved and that the resolution is clear and unambiguous. We respectfully request that you and your client review our response. We are confident that your review will confirm our findings. Therefore, we request that when your review is completed you confirm to us in writing that your concerns have been satisfactorily resolved. Of course, if you have any remaining issues, please do not hesitate to bring them to our attention.

We will continue to monitor Geronimo as we do all of our projects to ensure quality of the code contained therein. As an organization dedicated to open source software development, we are continuously working to ensure that the code that we provide through our many projects is free of IP encumbrance or questions of provenance. Please do not hesitate to contact us with any questions about this matter, or any further concerns your client may have.

Very truly yours,

Geir Magnusson Jr.
Director, The Apache Software Foundation

Attachment 1

The following is the summary of the investigation of the JBoss claims written by developers of the Geronimo community. It can be found in the Apache public CVS at

http://cvs.apache.org/viewcvs.cgi/incubator-geronimo/docs_nopublish/JBoss_20031031.html

This version has editorial insertions for clarification and addition to referenced attached materials.

Summary of Investigation

This document summarizes the results of the investigation into the allegations of similarity between JBoss code and Geronimo code. The original allegations are detailed in the letter dated October 31, 2003.

This document attempts to describe and characterize the technical issues surrounding the allegations. This is not a formal response by the Apache Software Foundation.

In the letter there are three named exhibits, A, B and C, and a fourth similarity that for the purposes of this discussion we will refer to as Assertion D.

Specifically, the exhibits:

- Exhibit A : The source file org.apache.geronimo.core.log.XLevel has a "very high degree of similarity" to org.jboss.logging.XLevel, and suggests that the Geronimo file is "derived from the JBoss file".
- Exhibit B : The source file org.apache.geronimo.core.log.PatternParser "appears to be nearly identical" to org.jboss.logging.layout.PatternParserX.
- Exhibit C : The source file org.apache.geronimo.common.InvocationType is "nearly identical" to org.jboss.invocation.InvocationType.
- "Assertion D", following Exhibit C : The source files org.jboss.invocation.Invocation and org.apache.geronimo.common.Invocation are similar. Further, the architectural concepts of "AsIs", "Transient" and "Marshalled" are present because of copying of the JBoss code, and that these concepts are central to the architecture of both JBoss and Geronimo.

Notes

1. The source code for Apache Geronimo is accessible via CVS at http://cvs.apache.org/view.cvs/incubator-geronimo/ and all Geronimo code references are relative to this root.
2. The source code for JBoss is accessible via CVS at http://cvs.sourceforge.net/viewcvs.py/jboss/jboss/src/main/ and all JBoss code references are relative to this root.
3. The Apache Software License is a business-friendly license that allows others to take our

software and use it as they please, as long as they respect the terms of our license. This will be important for Exhibits A and B - the license clearly states:

> *1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.*

Summary for Exhibit A
Exhibit A is concerned with similarity between org.apache.geronimo.core.log.XLevel and org.jboss.logging.XLevel rev 1.3.2.1.

This issue has been exhaustively researched [and documented – a copy of which is attached as Attachment 2] (http://www.qos.ch/logging/jboss.html) by the founder of the log4j project, Ceki Gülcü. His summary:

"In summary, the source code incriminated by Exhibit A existed at the Apache Software Foundation at least 6 months before it made its first appearance at JBoss. Thus, in relation to this exhibit, I cannot see how JBoss LLC has any valid claims against the ASF for this particular code. In fact, it appears that by neglecting to attribute the code and follow the Apache License for this example, the violation of copyright would be the exact reverse."

Ceki has traced back the history of the XLevel class, and thus we claim that the XLevel code originated at the ASF as part of the log4j project. Thus, we believe that Exhibit A is invalid due to the fact that the original source of the code in question is copyrighted by the ASF.

Summary for Exhibit B
Exhibit B in concerned with similarity between org.apache.geronimo.core.log.PatternParser and org.jboss.logging.layout.PatternParserEx.

According to the same research for Exhibit A by Ceki Gülcü :

"The PatternParserEx class, cited in Mr. David J. Byer's letter to the ASF, very closely follows the pattern established MyPatternParser and AppServerPatternParser classes. The earliest record of this class in JBoss source code repository dates to September 15th, 2002. Unless JBoss LLC claims that PatternParserEx predates MyPatternParser or AppServerPatternParser classes found in log4j, it looks like the JBoss LLC removed the existing Apache copyright when it based PatternParserEx class on modified versions of MyPatternParser and AppServerPatternParser. This is prohibited by the first clause of the Apache Software License.

We believe that the claims in Exhibit B are invalid owing to the fact that the code in question was based on code in the Apache log4j codebase.

Summary for Exhibit C
Exhibit C is concerned with the similarity between org.apache.geronimo.common.InvocationType and org.jboss.invocation.InvocationType. The following summary is based on discussion from the geronimo-dev list.

The initial version of org.jboss.invocation.InvocationType in the JBoss CVS repository can be found at the following location:

http://cvs.sourceforge.net/viewcvs.py/jboss/jboss/src/main/org/jboss/invocation/InvocationType.java?rev=1.1&view=markup

This contains a very similar code excerpt to that cited in the letter from JBoss Group LLC:

```
public static final InvocationType REMOTE =
    new InvocationType("REMOTE");
public static final InvocationType LOCAL =
    new InvocationType("LOCAL");
public static final InvocationType HOME =
    new InvocationType("HOME");
public static final InvocationType LOCALHOME =
    new InvocationType("LOCALHOME");
```

This code was contributed to the JBoss project by dsundstrom (Dain Sundstrom) on 7/14/2002. As the original copyright holder, Dain would be free to contribute this code to Geronimo as well.

The JBoss version of this code was modified exclusively by dsundstrom up to and including the 1.3 revision (dated 10/30/2002) in the JBoss CVS, where the code had evolved to the following:

```
public static final InvocationType REMOTE =
    new InvocationType("REMOTE", false, false);
public static final InvocationType LOCAL =
    new InvocationType("LOCAL", false, true);
public static final InvocationType HOME =
    new InvocationType("HOME", true, false);
public static final InvocationType LOCALHOME =
    new InvocationType("LOCALHOME", true, true);
```

Dain had the right to contribute to Geronimo the code up to (but not including) the changes made by Scott or anyone else.

Dain's last rev is at:

http://cvs.sourceforge.net/viewcvs.py/*checkout*/jboss/jboss/src/main/org/jboss/invocation/InvocationType.java?content-type=text%2Fplain&rev=1.3

The latest rev in JBoss is v 1.5, so we need to figure out what happened between v1.3, which Dain had the right to contribute to the ASF.

JBoss 1.3:
committed by Dain, for which he has the rights to contribute and re-license here in Geronimo under the ASL

JBoss v1.3 -> v1.4:
a) Switched from using a static int to a final static int as the max value for the InvocationType array. We see the same in the Geronimo code. This is a common, accepted practice for creating constants in Java, and would be suggested by any code inspector (like Idea or Eclipse). In my opinion, Dain should have his hand slapped for not doing it this way in the first place.
b) Switched to a statically allocated array [] rather than an ArrayList for holding the invocationType objects. This change is in the Geronimo code.
c) Changed the signature of the InvocationType from the Dain way (new InvocationType("LOCAL", false, true)) that is in Geronimo now, to a different way (new InvocationType("HOME", 2)). Clearly the Geronimo code didn't copy this change.
d) Removed two booleans isLocal and isHome, present in the Dain-contributed 1.3. These variables persist in the Geronimo version as local and home. This change was not adopted by Dain for Geronimo.

JBoss v1.4 -> v1.5:
Added a new invocation type 'SERVICE_ENDPOINT', which is not present in the Geronimo code. I conclude that any changes between v1.4 and 1.5 of the JBoss code were not co-opted into the Geronimo codebase.

Here's my conclusion. I would appreciate commentary:

Conclusion
==========

Dain contributed the same code to Geronimo that he contributed to JBoss.

The only difference between his JBoss contributions, for which he has complete rights to contribute and relicense elsewhere, is that he changed the Geronimo implementation to use an array rather than an ArrayList to hold the InvocationType objects, and a static final int 'constant' to keep the size of that array rather than a static int field.

Summary of Assertion D
Assertion D is concerned with the similarity between org.apache.geronimo.common.Invocation and org.jboss.invocation.Invocation. Further, the claim is that both files contain "AsIs," "Transient," and "Marshalled," which are believed to be JBoss-specific payloads, and thus could only be there via copying. Further, the claim is that Invocation file is central to the JBoss architecture, and thus copying could have great impact throughout Geronimo.

The file org.apache.geronimo.core.service.Invocation currently is a riff on java.util.Map, namely:

```
public interface Invocation {

    Object get(InvocationKey key);
    void put(InvocationKey key, Object value);
}
```

Clearly this can't be what the lawyers are talking about. However, it used to be, when the code first placed into Geronimo, slightly different:

http://cvs.apache.org/viewcvs/incubator-geronimo/modules/core/src/java/org/apache/geronimo/common/Attic/Invocation.java?rev=1.1&content-type=text/vnd.viewcvs-markup

```
public interface Invocation {
    Object getMarshal(Object key);

    void putMarshal(Object key, Object value);

    Object getAsIs(Object key);

    void putAsIs(Object key, Object value);

    Object getTransient(Object key);

    void putTransient(Object key, Object value);
}
```

This is still an interface, but dealing with the three notions of "Marshal," "AsIs," and "Transient." This is what the JBoss Group LLCs lawyers are referring to. Now, looking at the Invocation class in JBoss, and looking at the version in their CVS at the time of the import into Geronimo, 1.10.2.6, there is an implementation of the same notion in a method - following snipped out for brevity :

```
/**
    * Advanced store
    * Here you can pass a TYPE that indicates where to put the value.
    * TRANSIENT: the value is put in a map that WON'T be passed
    * AS_IS: no need to marshall the value when passed (use for all JDK
    *    java types)
    * PAYLOAD: we need to marshall the value as its type is application specific
    */
   public void setValue(Object key, Object value, PayloadKey type)
   {
      if(type == PayloadKey.TRANSIENT)
      {
         transient_payload.put(key,value);
      }
      else if(type == PayloadKey.AS_IS)
      {
         as_is_payload.put(key,value);
      }
      else if(type == PayloadKey.PAYLOAD)
      {
         payload.put(key,value);
      }
      else
      {
        throw new IllegalArgumentException("Unknown PayloadKey: " + type);
      }
   }
```

It appears that this is a storage class for moving bits through their invocation/interceptor mechanism, and that they are doing what appears to be an early optimization by having the caller define via the keytype if a) the data doesn't need to be marshalled as it's staying put on this side of the wire (TRANSIENT), b) the data doesn't need to have any special care and feeding as it's a JDK data type (ASIS), or c) it will need to be marshalled (PAYLOAD).

So it's clear to me that the code originally in Geronimo (and now in the Attic) implemented this *idea* in their interface, moving it outside of the Invocation implementation and into the interface. This is an implementation of the idea.

Now, I guess we have to come back to the code as it exists today in the o.a.g.core.service package. Repeating for Invocation.java, it is now much simpler - the Geronimo developers got rid of the "Asis," "Marshal," and "Transient" "modifiers" on the methods and reduced it to

```
public interface Invocation {

    Object get(InvocationKey key);
    void put(InvocationKey key, Object value);
}
```

where

```
public interface InvocationKey {

    boolean isTransient();
}
```

So now the idea of declaring something as not going over the wire (my assumption) is taken care of in the key itself into this map, letting (I assume again) the endpoint doing the serialization decide if the element in the map needs to go based on the isTransient() method, and how marshalled based on the class.

Summary so far: the original code had an expression of the idea of "Marshall," "AsIs" and "Transient." Most of the idea was dropped. All that remains of the idea is letting the caller declare the data as transient.

While the JBoss lawyers assert that "the Invocation file is central to the architecture of both JBoss and Geronimo," we believe that this claim is invalid because if this notion of AsIs, Transient and Marshalled was "central to the architecture," it couldn't be dropped to the degree that the Geronimo developers did. IOW, the notions of AsIs and Marshalled are NOT central to the architecture at all - they don't exist anymore. So to summarize:

1) The original code in Geronimo (w/ getAsIs()) is not a copy of JBoss code - it's a different implementation of an idea in the JBoss implementation.

2) The current code has thrown out all but the idea that the user of an Invocation implementation declare that data placed into that implementation is transient.

Attachment 2

The following is the content of http://www.qos.ch/logging/jboss.html, the documentation of the investigation of the Jboss allegations related to Exhibits A and B. This work was done by Ceki Gülkü, the founder and a developer of the Apache Log4J project.

Comments on JBoss' claims against the ASF
by Ceki Gülcü, November 12th 2003

The text below is my personal opinion on the allegation leveled by JBoss LLC against the Apache Software Foundation. I am not an officer of the ASF. The opinons expressed here represent my personal views on the matter.

On October 31st, 2003, an attorney representing JBoss LLC has sent a letter to the Apache Software foundation.

As the founder of the log4j project, I have been involved with log4j since 1999. Given that certain parts of JBoss claims against the Geronimo project of the Apache Software Foundation (ASF) are related to log4j, I have been asked to have a closer look at the charges leveled against the ASF.

It is very clear to *me* that the source code that the JBoss group claims as its own, is actually based on source code developed within the log4j project. Thus, in my opinion, the ASF has intellectual property rights on the source code mentioned in Exhibits A and B of the said letter.

Regarding Exhibit A: XLevel

`XLevel` stands for `eXtended Level`. The `XLevel` class, which I wrote, is intended as an example of how to extend existing log4j levels. I did not want to name the class `ExtendedLevel` because that seemed too long.

The capability to customize core log4j classes, in particular the `Priority` class, has existed for a long time.

Version 1.0.4 of log4j, a copy of which is readily obtainable was released on January 12th, 2001. This release contained the following file

```
jakarta-log4j-
1.0.4/org/apache/log4j/xml/examples/XPriority.java
```

On September 2nd 2001, the `Level` class replaced `Priority` class. The `Priority` class was kept around for compatibility reasons. It still exists today (as of 2003-11-12). As such, the `XPriority` class was appropriately renamed as `XLevel`. This was done on September 2nd, 2001, as attested by the records of the log4j source code repository.

I now list further evidence demonstrating that the code base in dispute originates from the log4j project.

org.apache.log4j.xml.test.TPriority

earliest version dated **December 14, 2000**

org.apache.log4j.xml.examples.XPriority
earliest version dated December 14, 2000

org.apache.log4j.xml.examples.XLevel
earliest version dated September 2, 2001

Please note that these classes are part of a whole. They integrate with various other classes, such as `XCategory` and `XLogger`. They are part of test suites testing their functionality. Only the original author would go to the pain of writing test suites to check the correctness of his code.

Please note that the TPriority class which stands for TestPriority is a cut-and-paste copy of XPriority. The same code was duplicated in org.apache.log4j.xml.test because I did not want to have test cases depend on code intended as examples.

In contrast, and as far as I can tell, the earliest mention of a subclass of `Priority` or `Level` in JBoss code dates to June 20th 2001 as attested by the JBoss source code repository.

org.jboss.logging.log4j.TracePriority
earliest version dated June 20, 2001

org.jboss.logging.XPriority
earliest version dated Feb 24, 2002

org.jboss.logging.XLevel
earliest version dated May 23, 2002

In summary, the source code incriminated by Exhibit A existed at the Apache Software Foundation at least 6 months before it made its first appearance at JBoss. Thus, in relation to this exhibit, I cannot see how JBoss LLC has any valid claims against the ASF for this particular code. ***In fact, it appears that by neglecting to attribute the code and follow the Apache License for this example, the violation of copyright would be the exact reverse.***

Regarding Exhibit B: PatternParser

The case for Exhibit B is similar. Customizations of `PatternLayout` via `PatternParser` extensions has been available for several years in log4j code.

The initial contribution allowing the extension of `PatternLayout` behavior was made by Anders Kristensen on 2000-08-27. Anders was granted committer access for this contribution. This extension was documented by Paul Glezen on January 2001. The description of the AppServerPatternParser.java class in the same document is particularly relevant. As further evidence, the file AppServerPatternParser.java is part of log4j version 1.1.3 released on June 19, 2001. Moreover, records of the log4j source code repository also attest for the existence of `PatternParser` extension on **December 14th, 2000**.

The `PatternParserEx` class, cited in Mr. David J. Byer's letter to the ASF, very closely follows the pattern established `MyPatternParser` and `AppServerPatternParser` classes. The earliest record of this class in Jboss source code repository dates to September 15th, 2002. Unless JBoss LLC claims that `PatternParserEx` predates

`MyPatternParser` or `AppServerPatternParser` classes found in log4j, it looks like the JBoss LLC removed the `existing` Apache copyright when it based `PatternParserEx` class on modified versions of `PatternParserEx` and `AppServerPatternParser`. This is prohibited by the first clause of the Apache Software License.

I hope you find the above convincing. When someone writes as much code as some of the JBoss developers, it is possible to forget the origin of the code. Notwithstanding the great respect and admiration I have for the JBoss project and its developers, I cannot let the name of the ASF to be falsely tarnished, especially if the attacking party is basing its accusation against the ASF on software which originated at ... the ASF. Talk about irony.

If any JBoss code is, in fact, in Geronimo, the ASF will without doubt remove it. I equally hope that the JBoss project will also take the time to correctly attribute those sections of JBoss software derived from ASF code.