Case Nos. 2017-1118, -1202

In the

United States Court of Appeals

for the

Federal Circuit

ORACLE AMERICA, INC.,

Plaintiff – Appellant

v.

GOOGLE INC.,

Defendant – Cross-Appellant

On Appeal from the United States District Court for the Northern District of California, Case No. 3:10-cv-03561-WHA · Honorable William H. Alsup, United States District Judge

BRIEF OF *AMICUS CURIAE* MOZILLA URGING AFFIRMANCE OF THE JUDGMENT

MARCIA HOFMANN
ZEITGEIST LAW PC
25 Taylor Street
San Francisco, California 94102
(415) 830-6664 Telephone
marcia@zeitgeist.law

Counsel for Amicus Curiae Mozilla

May 30, 2017

COUNSEL PRESS · (800) 3-APPEAL PRINTED ON RECYCLED PAPER

CERTIFICATE OF INTEREST

Counsel for *amicus curiae* certifies the following:

1. The full name of the *amicus curiae* represented by me is:

   Mozilla Corporation

2. The name of the real party in interest (if the party named in the caption is not the real party in interest) is:

   None.

3. All parent corporations and any publicly held companies that own 10 percent or more of the stock of the *amicus curiae* represented by me are:

   Mozilla Corporation is wholly owned by the Mozilla Foundation, a non-profit organization. No publicly held corporation owns 10% or more of Mozilla Corporation's stock.

4. The names of all law firms and the partners and associates that appeared for the *amicus curiae* now represented by me in the district court or are expected to appear in this court are:

   Mozilla did not appear in the district court.

   Mozilla is represented before this Court by Marcia Hofmann of Zeitgeist Law PC.

Dated: May 30, 2017

/s/ Marcia Hofmann
Marcia Hofmann

TABLE OF CONTENTS

TABLE OF AUTHORITIES

CASES

CONSTITUTIONS

STATUTES

OTHER AUTHORITIES

STATEMENT OF INTEREST OF *AMICUS CURIAE*[1]

Amicus curiae Mozilla Corporation is a technology company that believes the jury's finding of fair use in this case recognizes the value of independent reimplementation, which is vital to software development.

Mozilla is a global, mission-driven organization that works with a community of software developers around the globe to create open-source software such as the Firefox browser. Firefox is among the most popular browsers in the world. Several hundred million users rely on it to discover, experience, and connect to the Internet on computers, tablets, and mobile phones.

Mozilla's mission is guided by a set of principles recognizing that, among other things, free and open software promotes the development of the Internet as a global public resource, and that the effectiveness of that resource depends on interoperability. Mozilla is also the custodian of an open-source/free-software license called the Mozilla Public License.[2] The current version of that license explicitly recognizes fair use and equivalent rights under copyright laws.

---

[1] All parties have consented to the filing of this brief. It was not written in whole or part by counsel for any party. No person or entity other than undersigned counsel or *amicus* has made a monetary contribution to the preparation or submission of this brief.

[2] At https://www.mozilla.org/en-US/MPL. All of the websites cited in this brief were last visited on May 30, 2017.

INTRODUCTION

This case is about the freedom to build technology. Like any developer, when Google developed a new product—the Android platform—it wanted to ensure that other developers could begin writing software without having to learn a new programming language. To achieve this goal, Google used the declaring code and structure, sequence and organization of 37 of 166 Java API packages. Appx51938. Google implemented those packages with its own code to create the Android smartphone environment. Appx51098.

Although this case is about Google's use of Oracle's APIs, this Court's decision will affect developers around the world who are working every day to create new and innovative technologies to solve real problems. Developers rely on declaring code in the API development process to enable their software programs to interact with other software. This use results in independent implementations that are efficient and easy for other software developers to use.

The incorporation of declaring code in independent implementations aligns with the fundamental purpose of copyright, which is to promote creative works. Oracle argues that Google's use of declaring code in Android cannot be a fair use as a matter of law. Oracle Br. 25-55. If accepted by this Court, Oracle's proposed rule will have far-reaching implications for competition and the speed at which technologies are created. Technology will be less inclusive, slower to develop, and more expensive.

There must be latitude for such uses of code to be fair so that the software industry can continue to flourish.

ARGUMENT

A finding that re-using declaring code in an independent implementation is a fair use is consistent with the fundamental purpose of copyright law: to "promote the progress of Science and useful Arts." U.S. Const. art. I, § 8, cl. 8. Copyright law protects original expression while giving others the latitude to build upon earlier works to create new ones. This breathing room is created primarily through the fair use doctrine, which requires the courts "to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster." *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 577 (1994) (quoting *Stewart v. Abend*, 495 U.S. 207, 236 (1990)).

The Copyright Act lists four fair use factors for courts to consider: "(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the use upon the potential market for or value of the copyrighted work." 17 U.S.C. § 107(1)-(4). As this Court has recognized, these factors are "nonexclusive," and other considerations may be taken

into account as well. *Oracle Am., Inc. v. Google Inc.*, 750 F.3d 1339, 1373 (Fed. Cir. 2014) (quoting *Harper & Row, Publishers, Inc. v. Nation Enters.*, 471 U.S. 539, 549 (1985)).

I. Incorporating Declaring Code Into an Independent Implementation Can Weigh in Favor of Fair Use Under the First Factor.

The first fair use factor is "the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes." 17 U.S.C. § 107(1). The use of declaring code as part of an independent implementation can favor a finding of fair use under this factor because it can be transformative. And when an implementation is put under an open-source license, that decision has important public benefits that may mitigate an otherwise commercial purpose.

A. Incorporating Code into New and Different Software is Transformative When it Uses the Underlying Code in a New Context or Produces a New Creation.

The keystone of the first factor in a fair use analysis is the purpose and character of the use. 17 U.S.C. § 107(1); *Campbell,* 510 U.S. at 578-79; *see also* Pierre N. Leval, *Toward a Fair Use Standard*, 103 Harv. L. Rev. 1105, 1111 (1990). A use is more likely to be considered fair when it "changes" the underlying copyrighted work or uses it "in a different context" so that the work is "transformed into a new creation." *Oracle*, 750 F.3d at 1374 (quoting *Perfect 10, Inc. v. Amazon.com, Inc.*, 508 F.3d 1146, 1165 (9th Cir. 2007)).

The relevant question in a fair use analysis of the use of declaring code in a new implementation should not be whether the use transforms declaring code *per se*, but whether the implementation as a whole—*including* the declaring code—is a new and different work. A programmer reusing an existing piece of code can radically depart from the underlying purpose of the original work by implementing the package differently while retaining the basic declaring code to ensure the work remains compatible with others. These departures can be transformative, "add[ing] something new, with a further purpose or different character" to the declaring code. *Campbell,* 510 U.S. at 579. A finding that Google's use could not be fair as a matter of law would potentially imperil many transformative uses of declaring code.

An implementation may be transformative because it puts the code in a new and different context. For example, TweetNaCL is a re-implementation that performs many common security functions and is written to be as compact as possible: small enough to include in applications that could not fit a full security suite. TweetNaCL: a Crypto Library in 100 Tweets, https://tweetnacl.cr.yp.to; *see also* Matthew Green, *TweetNaCL*, A Few Thoughts on Cryptographic Engineering (July 20, 2013).[3] Applications that use this library integrate its functionality, giving it new context and purpose.

An implementation can change the affordances of a language. For example, Facebook has produced HHVM, a virtual machine that runs programs written in the

[3] At https://blog.cryptographyengineering.com/2013/07/20/tweetnacl.

language PHP. HHVM, http://hhvm.com. Before HHVM, PHP was usually interpreted into the language C++ or compiled as a static binary. HHVM uses the building blocks of the PHP language, similar to declaring code in an API, but runs the language in a different way. The result creates vast gains in speed and efficiency over executing PHP on the standard interpreter. *See, e.g.*, Erin Green, *Under the Hood: Box's HHVM Migration*, Facebook Code (July 14, 2015).[4] HHVM thus transforms PHP by giving it new characteristics and performance capabilities.

An implementation might solve for a particular weakness inherent in other code performing similar functions. For example, one group of programmers built a version of an important piece of security software in a new language that minimized the potential for certain security problems. Enguerrand Decorne et al., *OCaml Inside: a Drop-in Replacement for Libtls*, Ocaml Users and Developers Workshop 2016 at 1 (Aug. 8, 2016).[5] The programmers used the original software's declaring code to make the more secure implementation compatible with software that had been designed to use the more vulnerable implementation. The new implementation could replace the original version, immediately reducing the potential for security flaws.

An implementation may be developed to work within the constraints of embedded systems, allowing for expansion into entirely new and sometimes

[4] At https://code.facebook.com/posts/1607907626123431/under-the-hood-box-s-hhvm-migration.
[5] At https://www.cl.cam.ac.uk/~jdy22/papers/ocaml-inside-a-drop-in-replacement-for-libtls.pdf.

unexpected applications. For example, MicroPython is an implementation of the Python programming language for small circuit boards. MicroPython, https://micropython.org. Programmers who use it can build projects on connected devices such as a detector that can tell whether a door is open or closed. Tony DiCola, *MicroPython Basics: What is MicroPython*, Adafruit (Oct. 21, 2016).[6] MicroPython transforms the underlying code by implementing parts of the Python language in a different system, giving it new context and possibilities for development.

Likewise, toybox is a "from scratch" implementation of common Linux command-line utilities in a small, single executable that makes it more suited for minimal environments such as the Android platform. Toybox, https://landley.net/toybox. It was created to allow Android devices to run these tools, which made it possible to use Android as a development environment. *Make Android Self-Hosting (musl, toybox, qcc)*, Aboriginal Linux.[7] The toybox implementation makes it possible for Linux tools to function in a different context, on a new system while maintaining the same interface and behavior that had become standard for those tools. Toybox brought such value to the Android platform that it was merged into official Android versions beginning in 2015. *FAQ*, Toybox.[8]

The conclusion that these can all be forms of transformative uses is consistent with binding Ninth Circuit precedent, which has found that that producing a new

[6] At https://learn.adafruit.com/micropython-basics-what-is-micropython/overview.
[7] At https://landley.net/aboriginal/about.html#selfhost.
[8] At https://landley.net/toybox/faq.html.

platform that creates new opportunities for consumers can be a fair use, even if there are similarities in function and output. *Sony Computer Entm't v. Connectix Corp.*, 203 F.3d 596, 606-07 (9th Cir. 2000); *Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1522 (1992). They all create powerful new opportunities for existing copyrighted works, ensuring the advancement of software development and building on what came before. A finding by the Federal Circuit that Google's acts in this case cannot be fair use as a matter of law would cast a legal shadow over these many forms of re-implementation.

B. Commerciality May Have Little Weight in a Fair Use Analysis for an Implementation that Includes Declaring Code When the New Work is Put Under an Open-Source License.

A consideration under the first fair use factor is whether "a use is of a commercial nature or is for nonprofit educational purposes." 17 U.S.C. § 107(1). The commerciality of a use is "not conclusive," *Sony Corp. of Am. v. Universal City Studios, Inc.*, 464 U.S. 417, 448 (1984), and should not be elevated to a "hard presumptive significance," *Campbell,* 510 U.S. at 585. And as the Ninth Circuit has noted, a court may take into account "the public benefit resulting from a particular use notwithstanding the fact that the alleged infringer may gain commercially." *Sega*, 977 F.2d at 1523; *see also Hustler Magazine, Inc. v. Moral Majority, Inc.*, 796 F.2d 1148, 1152-53 (9th Cir. 1986).

Companies often produce software for private commercial gain. But when software is put under an open-source license, as Google has done in this case, that

decision has important public benefits that are also relevant to the fair use analysis. *Sega*, 977 F.2d at 1523; *Hustler*, 796 F.2d at 1152-3. Code offered to others under an open-source license will run on new platforms, be used for new purposes, tackle new problems, and create new solutions ranging from commercial to non-profit and educational. For example, a group of medical researchers re-implemented an open-source medical records platform to develop a new system for a study on tuberculosis epidemiology. Hamish SF Fraser et al., *Adaptation of a Web-Based, Open Source Electronic Medical Record System Platform to Support a Large Study of Tuberculosis Epidemiology*, 12 BMC Med. Informatics and Decision Making 125 (2012).[9] They were able to incorporate the platform into their academic research to advance the state of medical knowledge, a non-commercial use with significant social benefits.

Open-source licensing also helps the larger ecosystem to thrive by giving developers the ability to quickly and efficiently build new software to run on an existing platform, which makes innovation easier and in turn will give consumers more choice.

Because the public benefits that flow from open-source licensing are substantial, the decision to put an implementation under an open-source license should offset any commercial nature of the developer's use of declaring code in the implementation.

[9] At https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/1472-6947-12-125.

II. The Functional Nature of Declaring Code in an Implementation Can Support a Finding of Fair Use Under the Second and Third Fair Use Factors.

The second fair use factor considers "the nature of the copyrighted work," 17 U.S.C. § 107(2), and the third fair use factor weighs "the amount and substantiality of the portion used in relation to the copyrighted work as a whole," *id.* § 107(3). In an implementation that includes declaring code, these factors can both weigh in favor of a fair use finding, particularly due to the functional nature of the code at issue.

For purposes of the second fair use factor, the nature of declaring code is highly functional: it "identifies [a] prewritten function." *Oracle*, 750 F.3d at 1349. As this Court has recognized, even if API packages are protected by copyright, their functional aspects "may be relevant to a fair use analysis." *Id.* at 1376-77. The more functional the code, the more likely its use is fair.

In fact, a common use of declaring code is to automatically generate the documentation for software. There are software applications whose sole purpose is to parse source code files and associated comments—including any declaring code—to visualize and index the overall structure and components of software. Paul W. McBurney and Collin McMillan, *Automatic Documentation Generation via Source Code Summarization of Method Context*, Proceedings of 22nd International Conference on Program Comprehension at 279 (2014).[10]

[10] At https://www3.nd.edu/~cmc/papers/mcburney_icpc_2014.pdf.

As for the third fair use factor, an implementation does not have to incorporate much declaring code to achieve compatibility. Implementations might only use the code that defines the basic inputs and the outputs of the implementation, which may be quite minimal.

But even if an implementation uses a substantial amount of declaring code, that use may be fair when it is necessary to gain access to the functional elements embodied in the declaring code. *Connectix*, 203 F.3d at 603-04; *Sega*, 977 F.2d at 1527-28. A programming language specification may have many different implementations that emphasize different features. *See* Ryan Wilcox, *The Many Interpreters and Runtimes of the Ruby Programming Language*, Toptal.[11] All of these implementations must share declaring code with the language to conform to the specification, *i.e.*, in order to be that programming language. Thus, the functional nature of declaring code used in an independent implementation may support a finding of fair use under the second and third factors.

III. The Use of Declaring Code in an Independent Implementation Can Weigh in Favor of a Fair Use Under the Fourth Factor.

The fourth factor of the fair use analysis considers the effect of the use on the potential market for the original work. 17 U.S.C. § 107(4). The relevant question is whether the use would adversely affect the market for the work by "diminishing potential sales, interfering with marketability, or usurping the market[.]" *Sega*, 977 F.2d

[11] At https://www.toptal.com/ruby/the-many-shades-of-the-ruby-programming-language.

at 1523. Implementations can actually expand the market for software that they incorporate. And if the implementation is put under an open-source license, those terms can increase the value of the underlying work in other ways, as well.

A. The Use of Declaring Code in an Independent Implementation Can Expand the Market for the Original Software.

An implementation that includes declaring code can create new market opportunities for the underlying work by spurring innovation, increasing demand for the underlying technology, and creating opportunities to attract more users.

For example, when Philips Lighting introduced the Hue connected lightbulb, it published an API so that developers could build add-on innovation. One developer re-implemented the API as a library rather than a web service. *Node Hue API*, Github, https://github.com/peter-murray/node-hue-api. Users can integrate that library with Node-RED, a tool that makes programming more intuitive. *node-red-contrib-node-hue,* Node-RED, https://flows.nodered.org/node/node-red-contrib-node-hue. These integrations make the API accessible to a wider range of users who wish to program Hue lights, expanding the market and increasing user demand for the product.

Likewise, Mozilla has adopted Google Chrome's extensions API—which enables developers to build extensions that add new features to web browsers—for its own Firefox browser. *WebExtensions*, Mozilla Developer Network.[12] Mozilla's choice to support WebExtensions allows developers to build one extension and, after a few

[12] At https://developer.mozilla.org/en-US/Add-ons/WebExtensions.

tweaks, deploy the extension in a number of different browsers, such as Google's Chrome browser, Mozilla's Firefox browser, and Microsoft's Edge browser. This increases the number of potential extensions available to all users, allowing them to easily enhance and add new functionality to their chosen browser. Google benefits when others create new implementations of its extensions API (as Mozilla has), because Chrome users gain access to many new features that allow them to customize the browser to fit their needs and preferences.

Developers and users may choose a new implementation because it has capabilities that a different implementation does not, but keeping the same declaring code helps preserve the market for the original. For instance, Preact.js, an alternative to Facebook's React library, uses a nearly identical API but requires less memory and has higher performance. *See* Preact, https://preactjs.com. Preact is designed so that developers can add a compatibility layer and use React components in a Preact application. *Preact-Compat*, Github, https://github.com/developit/preact-compat. Preact creates a whole new environment where developers can use React components, increasing the functionality of React and expanding the market for React as a library.

Developers may have to switch to a new implementation when a flaw is discovered in the original implementation. The ability to re-implement with the same API may keep them from abandoning the platform entirely. For example, JNASmartCardio is a re-implementation of an Oracle Java library for interacting with smartcards such as chip and pin credit cards. *JNASmartCardio*, Github,

https://github.com/jnasmartcardio/jnasmartcardio. JNASmartCardio included a fix for a flaw in the official implementation that had made smartcard readers running Oracle's Java 7 for Mac unable to function correctly. *JDK-7195480: javax.smartcardio does not detect cards on Mac OS X*, Oracle Java Bug Database.[13] Until the flaw was fixed—a process that took more than a year and a half—the person who discovered the problem suggested using Apple's version of Java instead of the original implementation. *Id.* But users of JNASmartCardio were able to fix the flaw in their projects more easily than in the versions of the code bundled with the standard Java environment. The availability of JNASmartCardio may have kept some of those developers from choosing Apple's version of Java over Oracle's or abandoning Java altogether.

And an implementation may create new opportunities to use software that otherwise is not reachable with existing code. For example, Mozilla purposefully created its Geolocate API to use the same interface as the Google Maps Geolocation API endpoint, while adding some additional calls to allow for more features. *Compare Service API: Geolocate*, Mozilla,[14] *with Google Maps Geolocation API*, Google Maps APIs.[15] As a result, developers do not have to change their API calls when making a request for location information from Mozilla Location Services, and Mozilla preserves the flexibility to develop new features that the Google API does not support. The

[13] At http://bugs.java.com/bugdatabase/view_bug.do?bug_id=7195480.
[14] At https://mozilla.github.io/ichnaea/api/geolocate.html.
[15] At https://developers.google.com/maps/documentation/geolocation/intro.

implementation creates new demand and helps to solidify the viability of the technology, benefiting Mozilla, Google, and others on the Internet.

Thus, an implementation that includes declaring code may create a range of new opportunities for the underlying software, rather than adversely affect the market for the work.

B. Independent Implementations are Likely to Enhance the Value of Underlying Software When They Are Put Under and Open-Source License.

Implementations are particularly likely to encourage new innovation when placed under an open-source license, which may grow the market for both works by encouraging the development of new technologies that utilize the code in both implementations. One of the most fundamental aspects of an open-source license is that any new work released under the license is available for anyone to use—*including the developer of the underlying software*.

As this Court has found, the value of software distributed under open-source licenses is manifold. *Jacobsen v. Katzer*, 535 F.3d 1373, 1378-79 (Fed. Cir. 2008); *see also Planetary Motion, Inc. v. Techsplosion, Inc.,* 261 F.3d 1188, 1200 (11th Cir. 2001). The benefits are not limited to licensing royalties. They also include reputational enhancement, product improvements, and other economic benefits. *Jacobsen,* 535 F.3d at 1378-9; *see also* Steven Melendez, *How Facebook's Massive Open-Source Push Delivers*

Better Code and Better Engineers, Fast Company (Jan. 26, 2015);[16] *Open Source (Almost) Everything*, Tom Preston-Werner (Nov. 22, 2011) (co-founder of Github explaining that open sourcing code is "great advertising," helps to attract and retain talent, and reduces duplication of coding effort, among other things).[17] Developers may put an implementation under an open-source license to reap these benefits, which do not "diminish[] potential sales, interfer[e] with marketability, or usurp[] the market" for an underlying work incorporated into the implementation. *Sega*, 977 F.2d at 1523. To the contrary, the license can help build adoption and expand the market for the reimplementation and the underlying work.

Sun itself recognized the value for itself *and* other developers when it released Java under an open-source license. Sun put its Java platform implementations under a version of the GNU General Public License to grow the reach of Java and the number of developers familiar with it so that Sun could license complementary technologies. Appx50499. Sun also benefited in indirect ways. For example, a group of reimplementers created the GNU Classpath open-source version of the Java API without Sun's permission. Appx50990-50992. Sun's discussions with the reimplementers ultimately helped Sun to improve the quality of its own API. Appx50992.

[16] At https://www.fastcompany.com/3038842/how-facebooks-massive-open-source-push-delivers-better-code-and-better-engineers.
[17] At http://tom.preston-werner.com/2011/11/22/open-source-everything.html.

Sun recognized that software developed by others would also benefit from the decision to put Java under an open-source license. As James Gosling, the "father of Java technology," said at that time, other open-source implementations of the Java programming language would "certainly be able to mine our source for stuff to incorporate into their projects." Robert Eckstein, *James Gosling on Open Sourcing Sun's Java Platform Implementation*s*, Part 2*, Oracle Technology Network (Nov. 2006).[18] Indeed, Java is widely popular today in part because of Android's success and the innovation built on top of the Android platform.

Other companies realize the inherent value of releasing their code under an open-source license. Developers ranging from Microsoft to Adobe to IBM host open-source projects to which developers can contribute, improving not just the code but the larger ecosystem, as well.[19] Netflix, for example, open sources a number of its own internal projects to improve the software and the market. *Open Source at Netflix*, Netflix Technology Blog (July 13, 2012);[20] *see also* Netflix Open Source Software Center, https://netflix.github.io. In one recent case, Netflix created and open sourced a software suite to test implementations of TLS, an Internet protocol that protects communications through end-to-end encryption. *BetterTLS*, Netflix Technology Blog (Apr. 10, 2017).[21] Netflix discovered that many browsers had implemented TLS in a way that left users exposed to a certain security issue. By open sourcing the test suite,

[18] At https://www.oracle.com/technetwork/articles/java/gosling-os2-qa-136546.html.

Netflix made it possible for other vendors to detect the issue quickly and easily. In fact, both Google and Oracle improved their TLS implementations as a result. *Id.*

Thus, independent implementations may not only expand the market for the software that they incorporate, but can also increase the value of the underlying work in other ways. Implementations put under an open-source license are particularly likely to encourage new innovation and benefit the developer of the original work. These realities weigh in favor of a fair use finding under the fourth factor.

In sum, the use of declaring code in an independent implementation can weigh in favor of fair use under every factor. A finding that such a use cannot be a fair use as a matter of law would be at odds with the fundamental purpose of copyright law: to promote progress. Mozilla urges this Court to reject that position and ensure the law allows flexibility for fair uses of software, in turn fostering innovation.

---

[19] *See* https://github.com/Microsoft; https://github.com/adobe; https://github.com/ibm.
[20] At https://medium.com/netflix-techblog/open-source-at-netflix-c2c4e036e144.
[21] At https://medium.com/netflix-techblog/bettertls-c9915cd255c0.

CONCLUSION

Mozilla respectfully asks this Court to uphold the jury verdict in Google's favor and affirm the district court's judgment.

Dated: May 30, 2017

Respectfully submitted,

/s/ Marcia Hofmann
Marcia Hofmann
Zeitgeist Law PC
25 Taylor Street
San Francisco, CA 94102
Telephone: (415) 830-6664
marcia@zeitgeist.law

Counsel for Amicus Curiae

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITATION, TYPEFACE REQUIREMENTS, AND TYPE STYLE REQUIREMENTS

I certify that the foregoing brief complies with the type-volume limitation of Federal Rules of Appellate Procedure 29(a)(5) and 32(a)(7), as well as Federal Circuit Rule 28.1. This brief contains 4,130 words, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(f).

This brief's type size and typeface comply with Federal Rule of Appellate Procedure 32(a)(5) and (6) and Federal Circuit Rule 28.1. It was written in Garamond proportionally spaced typeface with 14-point font.

Dated: May 30, 2017

/s/ Marcia Hofmann
Marcia Hofmann

CERTIFICATE OF SERVICE

I hereby certify that on May 30, 2017, I electronically filed the foregoing *amicus curiae* brief with the Clerk of the Court for the United States Court of Appeals for the Federal Circuit by using the appellate CM/ECF system.

I certify that all participants in the case are registered CM/ECF users and that service will be accomplished by the appellate CM/ECF system.

/s/ Marcia Hofmann
Marcia Hofmann