MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
DANIEL P. MUINO (Bar No. 209624)
dmuino@mofo.com
755 Page Mill Road, Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street, Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900, Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway, Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.

Plaintiff,

v.

GOOGLE INC.

Defendant.

Case No. CV 10-03561 WHA

ORACLE'S OBJECTIONS TO GOOGLE'S PROPOSED CLAIM CONSTRUCTIONS

Dept.: Courtroom 8, 19th Floor
Judge: Honorable William H. Alsup

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

CASES

OTHER AUTHORITIES

INTRODUCTION

Oracle does not believe additional claim construction briefing is necessary at this stage. Google has proposed three terms for construction. Two have their plain and ordinary meaning, and need no construction. Google's proposals are unduly narrow and run afoul of the specifications. The third is "computer-readable medium" from the '476 patent, which the parties fully briefed in March of this year. For convenience, Oracle presents the salient portions of its argument here.

ARGUMENT

I. OBTAIN A REPRESENTATION OF AT LEAST ONE CLASS FROM A SOURCE DEFINITION PROVIDED AS OBJECT-ORIENTED PROGRAM CODE ('720 PATENT)

Claim	Term or Phrase	Oracle Proposed Construction	Google Proposed Construction
'720 patent, Claims 1 and 10	obtain[ing] a representation of at least one class from a source definition provided as object oriented program code	No construction necessary. The phrase has the ordinary meaning that its constituent words give it.	load at least one class definition by compiling object oriented source code

There is no need to construe the phrase "obtain a representation of at least one class from a source definition provided as object-oriented program code," which has the ordinary meaning that its constituent words give it. There is no "better" way to rephrase the phrase that will help the jury weigh the testimony from the experts about whether Android performs the obtaining step or whether the prior art does or does not disclose it.

Yet, Google seeks a specialized construction for the phrase—to have it mean "load at least one class definition by compiling object oriented source code"—to support a non-infringement argument. But that is not what the phrase means.

Indeed, the term "source code" employed in Google's proposed construction does not appear anywhere in the '720 patent. The '720 patent instead uses the word "source" to mean "a point of origin or procurement," which is its plain and ordinary meaning. WEBSTER'S NINTH NEW COLLEGIATE DICTIONARY 1127 (1986) (Declaration of Marc Peters in Support of Oracle's

Objections to Google's Proposed Claim Constructions ("Peters Decl.") Ex. A); *see also* MICROSOFT PRESS COMPUTER DICTIONARY 443 (3d ed. 1997) (definition of "source": "In information processing, a disk, file, document, or other collection of information from which data is taken or moved.") (Peters Decl Ex. A). The '720 patent also uses the phrase "source definition," which is where the class preloader obtains the stored representations of one or more classes. The '720 patent does not use the word "source" to refer to source code. Although the "source definition provided as object-oriented program code" could be source code, and could be written in a high-level programming language, the preferred embodiments of source definitions in the '720 specification are object code: binary forms of class definitions. It would be error to construe the claim to exclude these embodiments, and Google's proposal should be rejected. *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1583-84 (Fed. Cir. 1996) (construction that excludes the preferred embodiment is "rarely, if ever, correct").

A. Google's proposed narrowing construction excludes the preferred embodiments disclosed by the '720 patent

The '720 patent discloses many examples of a "source definition provided as object-oriented program code" that are in binary or object code form. These disclosures use the word "source" to mean "a point of origin or procurement," not "source code." "Source" refers to the sources of class definitions shown in Figures 1 and 2. Figure 1 shows that class libraries containing class representations may be stored on local (client) and remote (server) file systems (class libraries 17 and class libraries 20) as well as individual class files stored in local file systems (classes 16). Each of the classes and class libraries are source definitions—for example, "[e]ach client 13 is operatively coupled to a storage device 15 and maintains a set of classes 16 and class libraries 17, which respectively ***define*** code modules that specify data structures and sets of methods that operate on the data, and shareable collections of the modules." '720, 4:32-37 (emphasis added.)

Class libraries are collections of already-compiled classes; they are not source code repositories from which code is compiled on the fly. This is shown in more detail in Figure 2, which, like Figure 1, shows a client system having classes 36 and class libraries 37 in its storage

35. The figure shows that the system also includes System App Class Loader 40, Bootstrap Class Loader 39, and Applications Launched 38, each of which is a compiled executable program. ’720, 5:48-50 (“Upon initialization, the master JVM process 33 reads an ***executable process image*** from the storage device 35 and performs bootstrapping operations.”) (emphasis added). Notably, the figure uses the same graphic icon (a box with a wavy bottom) for the class loaders and launched applications as it does for classes 36 and class libraries 37, indicating that the classes and libraries are executable binaries as well.

The claims confirm this understanding. Claim 14, which depends from asserted Claim 10, requires “maintaining the source definition as a class file on at least one of a local and remote file system.” Claim 5 has similar language. “Class file” is a term that refers to a compiled binary file satisfying the Java Virtual Machine Specification. It is object code, not source code. When a source definition is maintained as a class file, the class representation it contains is already in binary form, and the representation is obtained by loading it, not by compiling source code. Claims 5 and 14 are original claims that were filed with the initial application (Claim 14 was originally numbered Claim 16), and as such are part of the specification and provide the written description of the invention under Section 112. ’720 Original Application at 16, 18 (attached as Peters Decl Ex. B); *N. Telecom, Inc. v. Datapoint Corp.*, 908 F.2d 931, 938 (Fed. Cir. 1990) (original claims are part of the patent specification); *In re Benno*, 768 F.2d 1340, 1346 (Fed. Cir. 1985) (same). Class representations cannot be obtained from class files by compiling source code, because there is no source code in a class file. Google’s proposal is directly contrary to the specification and the claim language; it cannot be correct.

The specification further confirms that the claimed source definition can contain binary representations of classes. The ’720 specification states: “Class loading requires identifying a ***binary form of a class type*** as identified by specific name, as further described below with reference to FIG. 10. Depending upon whether the class was previously loaded or referenced, class loading can include ***retrieving a binary representation from source*** and constructing a class object to represent the class in memory.” ’720, 6:49-54 (emphasis added).

Figure 10 of the '720 patent also demonstrates that the source definitions are in binary form, not source code. It shows a flow diagram showing process steps for preloading a class. Step 157 is "load the bytes for class from source associated with class loader." This step in the flow diagram corresponds to the phrase "obtain a representation of at least one class from a source definition provided as object oriented program code." Step 157 is described in the specification as follows: "the master JVM process 33 attempts to ***load the bytes for the class from the source*** associated with the applicable bootstrap class loader 39 and system application class loader 40 (block 157)." '720, 9:48-51 (emphasis added). The word "bytes" in the diagram and the written description indicates that the source definition is in object code form—Java bytecode, to be specific—not source code form. The "source definition" is where the bytes that define a class or classes are stored, such as the class files or class libraries illustrated in Figures 1 and 2.

Further intrinsic evidence confirming that the class file embodiments are object code, not source code, may be found in the specification of U.S. Patent No. 7,213,240, which is incorporated into the '720 patent by reference. '720, 3:1-6 ("The classes and interfaces are identified through profiling by ranking a set of classes according to a predetermined criteria, such as described in commonly-assigned U.S. patent application Ser. No. 09/970,661, filed Oct. 5, 2001, pending, the disclosure of which is incorporated by reference."). The '240 specification, which is part of the '720 specification, distinguishes between class files and source code: "Profiling tool 420 runs on ***class files or Java source code*** to create an ordered list of methods based on predetermined criteria." '240, 9:44-46 (emphasis added) (Peters Decl. Ex. C). The '240 specification further explains that a virtual machine obtains program code in the form of class files (not source code) from sources such as servers:

> Host 102 and servers 104a-104n may ***supply devices 106a-106n with programs written in a platform-independent language, such as Java***. For example, a software developer may create one or more Java programs and compile them into class files that contain bytecodes executable by a virtual machine, such as a Java virtual machine. When a device, such as device 106a, wishes to execute a Java program, it may issue a request to a server, such as server 104a, that contains the program. In response, server 104a ***transmits the corresponding class files*** to device 106a via an appropriate communication channel, such as network 108

> (which may comprise a wired or wireless communication network, including the Internet). Device 106a may ***load the class files into a virtual machine*** located in device 106a and proceed to execute the Java program.

’240, 5:46-60 (emphasis added). This passage makes clear that when ’720 dependent claims 8 and 17 add the further limitation “wherein the object-oriented program code is written in the Java programming language,” the claims embrace class files, which are programs written in Java. Google’s expected argument that Claims 8 and 17 demonstrate that Claims 1 and 10 are limited to compiling source code has no foundation in the ’720 specification. Even if that were not the case, Google’s argument would still fail, because the doctrine of claim differentiation means that Claim 1 is broader than Claim 8, and likewise Claim 10 is broader than Claim 17. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1315 (Fed. Cir. 2005) (*en banc*) (presence of a dependent claim that adds a particular limitation gives rise to a presumption that the limitation in question is not present in the independent claim). Dependent claims do not limit their independent claims.

B. Google’s proposed construction contradicts the claim language, excludes the preferred embodiments, and is based on a misreading of the patent specification

For the reasons explained above, Google’s proposed construction directly contradicts the ’720 specification and claim language. The ’720 patent does not disclose any embodiment in which the act of obtaining involves loading a class definition by compiling source code. But it does disclose embodiments in which the source definitions are maintained as already-compiled class files, and these particular embodiments are specifically claimed in claims 5 and 14. Google’s proposal is wrong because it ***excludes*** class files from the scope of the claims.

To support Google’s argument, Google and its expert rely on a misreading of a sentence fragment in the ’720 specification, which is the only place the word “compiling” appears in the specification. They misread both the claim element to which the sentence corresponds as well as the meaning of the sentence. Recall that Claim 10 of the ’720 patent has an “interpreting and instantiating” step that follows the obtaining step:

> obtaining a representation of at least one class from a source definition provided as object-oriented program code;

> interpreting and instantiating the representation as a class definition in a memory space of the master runtime system process;

’720, 10:54-59. The ’720 patent describes an embodiment of these steps in this way, in connection with Figure 10:

> Otherwise, the master JVM process 33 ***attempts to load the bytes for the class from the source*** associated with the applicable bootstrap class loader 39 and system application class loader 40 (block 157). If successful (block 158), an ***instance of the class is created by compiling the source and the class instance is installed in the system class dictionary*** (block 160). If the bytes for the class cannot be loaded from the source (block 158), the master JVM process 33 throws a class not found exception (block 159).

’720, 9:48-56 (emphasis added). The attempt to “load the bytes for the class from the source” corresponds to the “obtaining” step. The description that an “instance of the class is created by compiling the source” corresponds to the “interpreting and instantiating the representation as a class definition” step of Claim 10. It is wrong to use the reference to compiling to interpret the meaning of the “obtaining” step. And Google misreads “compiling” by interpreting it to mean “compiling source code” and nothing else. That is far too narrow, because the ’720 specification identifies the “source” as containing “bytes,” not source code.

Google’s argument fails because bytecode can be compiled, just like source code. As persons of ordinary skill in the art knew well at the time of the ’720 invention, Java bytecode (object-oriented program code in binary form) can be and was often compiled to native code, typically by a Java virtual machine that had a “Just-In-Time” compiler. The background section of the ’205 patent, which was first filed in 1997 and published in 2002, well before the ’720 application was filed, discusses the known concept of compiling Java virtual machine instructions (as opposed to source code):

> A known method for increasing the execution speed of Java interpreted programs of virtual machine instructions involves utilizing a just-in-tine (JIT) compiler. The JIT compiler compiles an entire Java function just before it is called. However, native code generated by a JIT compiler does not always run faster than code executed by an interpreter. For example, if the interpreter is not spending the majority of its time decoding the Java virtual machine instructions, then compiling the instructions with a JIT compiler may not increase the execution speed. In fact, execution may even be slower utilizing the JIT compiler if the overhead of

> compiling the instructions is more than the overhead of simply interpreting the instructions.

’205, 2:1-13.

When the ’720 patent specification discloses loading the bytes for the class from the source and then creating an instance of the class by compiling the source, the straightforward reading is that, in this embodiment, the class loader loads a class file (which contains a class representation) containing Java bytecode and then, as part of instantiating the representation as a class definition in the memory space of the Java virtual machine, it compiles the bytecode to native code or another useful form. Nothing in the ’720 specification limits the obtaining step to compiling source code.

C. Google now proposes a construction that contradicts the position it argued to the PTO in its reexamination request

Google’s position here is not only contrary to the patent specification, but also contrary to the position it took in front of the PTO when arguing for the invalidity of the patent. In its *inter partes* reexamination request, Google argued that the Java class files disclosed in a prior art patent to Webb correspond to the “source definition provided as object oriented program code.” Google stated that “Webb implements a class pre-loader: ‘[a]t run-time objects are created as instantiations of these class files, and indeed the class files themselves are effectively loaded as objects.’ *See* Webb at 1:22-38.” Google’s Request for *Inter Partes* Reexamination at 22 & Exhibit 17 (Peters Decl. Ex. D). Google argued to the PTO that a Java class file—which is already-compiled object code, not source code—is the “source definition provided as object oriented program code” from which a class preloader obtains a representation of a class when the class is to be loaded. Now it argues that cannot be true as a matter of law, though Google has not informed the PTO of its new position.

The Court should reject Google’s inconsistent and unsupported argument. The phrase “obtain a representation of at least one class from a source definition provided as object-oriented program code” needs no special construction.

II. RUNTIME ('205 PATENT)

Claim	Term or Phrase	Oracle Proposed Construction	Google Proposed Construction
'205 patent, Claim 1	Runtime	No construction necessary. The ordinary meaning is "during execution of the virtual machine." (per 2/22/2011 Joint Claim Construction Statement)	during execution of the virtual machine instructions (per 2/22/2011 Joint Claim Construction Statement)

The '205 patent generally relates to execution speed optimization techniques. The term "runtime" appears twice in independent and asserted Claim 1, once in the preamble and once in the body. The disputed term does not appear *in haec verba* in the specification of the '205 patent. In the Joint Claim Construction Statement (ECF No. 91), the parties did not identify any part of the '205 prosecution history or extrinsic evidence that informs the construction of the term.

Google's proposed construction appears close to the plain meaning, in that the parties agree that "runtime" as recited in Claim 1 of the '205 patent means during execution of the virtual machine. Indeed this is the ordinary meaning of "runtime" as recited in Claim 1 of the '205 patent. But Google's proposed construction narrows "runtime" to execution of the *particular* virtual machine *instructions* being optimized for increased execution speed. Google's purpose is to support a non-infringement argument that when Android's dexopt program loads application classes into the runtime environment of a running Dalvik virtual machine, that is not "runtime."

This is not required by the plain language of Claim 1 nor supported by the specification of the '205 patent. Because Google's proposed construction is unnecessarily narrow, it is incorrect and should be rejected.

A. No construction is necessary

As used in Claim 1 of the '205 patent, "runtime" has its ordinary meaning—during execution of the virtual machine—and no construction is necessary. *See, e.g.*, *Phillips*, 415 F.3d at 1314 ("To begin with, the context in which a term is used in the asserted claim can be highly instructive."). Claim 1 of the '205 patent reads:

> 1. In a computer system, a method for increasing the execution speed of virtual machine instructions at ***runtime***, the method comprising:

> receiving a first virtual machine instruction;
>
> generating, at ***runtime***, a new virtual machine instruction that represents or references one or more native instructions that can be executed instead of said first virtual machine instruction; and
>
> executing said new virtual machine instruction instead of said first virtual machine instruction.

The parties do not dispute that a virtual machine is the computer system being referred to in the preamble of Claim 1. According to the text of Claim 1, "runtime" allows for the claimed method in a virtual machine to *receive* a first virtual machine instruction, *generate a new virtual machine instruction that can be executed instead of the first virtual machine instruction*, and *execute* the new virtual machine instruction instead. This implies that a virtual machine must be up-and-running in order to perform these steps, but the claim requires nothing more.

B. Google's proposed construction is not supported by the plain language of Claim 1

There is nothing suggested by the plain and ordinary meaning of Claim 1 that requires Google's narrowing construction. Below is Google's proposed construction substituted in for the disputed term:

> 1. In a computer system, a method for increasing the execution speed of virtual machine instructions ***[during execution of the virtual machine instructions]***, the method comprising:
>
> receiving a first virtual machine instruction;
>
> generating, ***[during execution of the virtual machine instructions]***, a new virtual machine instruction that represents or references one or more native instructions that can be executed instead of said first virtual machine instruction; and
>
> executing said new virtual machine instruction instead of said first virtual machine instruction.

First, the only available antecedent basis for Google's proposed construction is contained in the preamble's recitation of "a method for increasing the execution speed of virtual machine instructions." This means that Google's proposed construction necessitates that the execution speed optimization claimed by the '205 patent occur only when the virtual machine is actually executing the virtual machine instructions that are being optimized. But Claim 1 is not so narrowly written. In particular, nothing in the "generating, at runtime" step requires execution of any virtual machine instructions; all that is required is the generation of a new virtual machine instruction "that can be" executed instead of the first virtual machine instruction. The only part of Claim 1 that actually mentions executing a virtual machine instruction is the last step, which recites "executing said new virtual machine instruction." But the "executing" step is separate and independent of the "generating, at runtime" step. The claim does not require the execution of the "virtual machine instructions" mentioned in the preamble; actually, the preamble only states the intended purpose of the invention and is not limiting in any way, because nothing in the body of the claim refers to it or needs it to give "life, meaning and vitality" to the claim. *Catalina Mktg. Int'l, Inc. v. Coolsavings.com, Inc.*, 289 F.3d 801, 808 (Fed. Cir. 2002) (preamble not limiting "where a patentee defines a structurally complete invention in the claim body and uses the preamble only to state a purpose or intended use for the invention") (citations omitted). Indeed, the claim does not require execution of *any* received virtual machine instruction, only the execution of the generated new virtual machine instruction *instead of* the received virtual machine instruction. A virtual machine may never execute any of "the virtual machine instructions" of Google's proposed construction, and yet may satisfy every step of Claim 1.

Google's particularized construction unduly narrows the scope of Claim 1 in direct contravention of the plain language of the claim. Google's proposed construction is incorrect for this reason alone and should be rejected.

C. Google's proposed narrowing construction is not supported by the specification

The question that is dispositive of the parties' dispute is as follows: Does the '205 patent's claimed invention cover the situation where the virtual machine is executing but is not necessarily executing virtual machine instructions? It does. The specification of the '205 patent discloses:

> FIG. 13 shows a bytecode table that may be utilized to store information regarding different Java bytecodes. A bytecode table 1051 includes information regarding each of the bytecodes of the virtual machine instructions. ***In a <u>preferred</u> embodiment, the bytecode table is generated once when the Java virtual machine is initialized.***

’205, 13:3-8 & Fig. 13 (emphases added). To be clear, the ’205 patent uses the term “bytecode” interchangeably with “virtual machine instruction.” *See, e.g.*, 1:43-45 (“Java programs are compiled into class files which include virtual machine instructions (e.g., bytecodes) for the Java virtual machine.”); 4:22-24 (“Bytecode pointer (BCP)--A pointer that points to the current Java virtual machine instruction (e.g., bytecode) that is being executed.”); 5:10-15 (“Typically, computer programs written in the Java programming language are compiled into bytecodes or Java virtual machine instructions which are then executed by a Java virtual machine. The bytecodes are stored in class files which are input into the Java virtual machine for interpretation.”); 5:21-25 (“The bytecodes are virtual machine instructions as they will be executed by a software emulated computer.”); 7:40-42 (“In the Java virtual machine, the virtual machine instructions are bytecodes meaning that each virtual machine instruction is composed of one or more bytes.”).

During initialization, the Java virtual machine is known to perform various tasks that do not involve executing any virtual machine instructions. Examples of such tasks include class loading, class verification, and bytecode optimizations, such as the one disclosed as a preferred embodiment in the ’205 patent’s specification (*see* ’205, 13:3-26 & Fig. 13). *See also, e.g.*, ’205, 5:28-30 (“The Java class file is input into a Java virtual machine 107. The Java virtual machine is an interpreter that ***decodes and executes*** the bytecodes in the Java class file.”) (emphasis added); 5:35-41 (separately explaining what it means for an interpreter or virtual machine to execute a bytecode program).

It was well-known in the art that there are many tasks a virtual machine may perform before executing any virtual machine instructions, and the other patents-in-suit provide many examples. *See, e.g.*, ’702, 3:7-11 (“A ‘class loader’ within the virtual machine is responsible for loading the bytecode class files as needed, and either an interpreter executes the bytecodes directly, or a ‘just-in-time’ (JIT) compiler transforms the bytecodes into machine code, so that they can be executed by the processor.”); *id.* at 36:9-11 (“Instead, Sun’s Java Virtual Machine implementation verifies that each class file it considers untrustworthy satisfies the necessary constraints at linking time (§2.16.3).”); *id.* at 44:4-14 (“Java classes and interfaces are dynamically loaded (§2.16.2), linked (§2.16.3), and initialized (§2.16.4). Loading is the process of finding the binary form of a class or interface type with a particular name and constructing, from that binary form, a Class object to represent the class or interface. Linking is the process of taking a binary form of a class or interface type and combining it into the runtime state of the Java Virtual Machine so that it can be executed. Initialization of a class consists of executing its static initializers and the initializers for static fields declared in the class.").

A virtual machine thus may be up-and-running and performing numerous tasks that do not involve executing virtual machine instructions. One of those tasks is the preferred embodiment disclosed in the '205 patent, which is further detailed below:

> The bytecode table may include a pointer to snippet code 1061 for each bytecode to which native machine instructions will be generated. Thus, as shown, a template table 1063 may be utilized to store templates for the native machine instructions for each bytecode. The template table allows for fast generation of snippets as the native machine instructions for the bytecodes may be easily determined upon reference to template table 1063. ***Additionally, the templates of native machine instructions may also be used to interpret the bytecodes.*** Another column in bytecode table 1051 may indicate a snippet code size 1065 of the template in the template table.

'205, 13:15-26 (emphasis added). In other words, the '205 patent's Figure 13 preferred embodiment describes the technique of generating a new virtual machine instruction that represents or references one or more native instructions that can be executed instead of the original virtual machine instruction while the virtual machine is up-and-running, before executing any virtual machine instructions.

Google's proposed construction conflicts with this preferred embodiment because it necessitates that the optimization claimed by the '205 patent occur only when the virtual machine is actually executing the virtual machine instructions being optimized. Because Google's proposed construction does not encompass the '205 patent's Figure 13 preferred embodiment, Google's proposal is not correct. *See, e.g.*, *Vitronics*, 90 F.3d at 1583-84 ("Therefore, in order to be consistent with the specification and preferred embodiment described therein, claim 1 must be construed such that the term 'solder reflow temperature' means the peak reflow temperature, rather than the liquidus temperature. Indeed, if 'solder reflow temperature' were defined to mean liquidus temperature, a preferred (and indeed only) embodiment in the specification would not fall within the scope of the patent claim. Such an interpretation is rarely, if ever, correct and would require highly persuasive evidentiary support, which is wholly absent in this case.") (citation omitted); *Modine Mfg. Co. v. U.S. Int'l Trade Comm'n*, 75 F.3d 1545, 1550 (Fed. Cir. 1996) ("Indeed, a claim interpretation that would exclude the inventor's device is rarely the correct interpretation; such an interpretation requires highly persuasive evidentiary support, whereas in this case it received none, whether from the specification, the prosecution history, or the prior art."); *Hoechst Celanese Corp. v. BP Chems. Ltd.*, 78 F.3d 1575, 1581 (Fed. Cir. 1996) ("We share the district court's view that it is unlikely that an inventor would define the invention in a way that excluded the preferred embodiment, or that persons of skill in this field would read the specification in such a way.").

III. COMPUTER-READABLE MEDIUM ('476 PATENT)

Claim	Term or Phrase	Oracle Proposed Construction	Google Proposed Construction
'476 patent, Claim 14	computer-readable medium	a storage device for use by a computer (per 2/22/2011 Joint Claim Construction Statement, and fully briefed by the parties)	any medium that participates in providing instructions to a processor for execution, including but not limited to, optical or magnetic disks, dynamic memory, coaxial cables, copper wire, fiber optics, acoustic or light waves, radio-waves and infra-red data communications

In the claim construction process conducted early this year, the Court was asked to construe "computer-readable medium," "computer-usable medium," and "computer readable storage medium," which appeared in six of the then-asserted patents. The parties' arguments for each of the phrases in each of the patents were fully briefed. In May 2011, the Court declined to construe any of the phrases, having construed five other terms and concluding that, because "[c]onstruing these terms properly would require individualized attention to the intrinsic evidence and prosecution history of each of the six patents from which they hail," the "computer-readable medium" phrases were "equivalent to at least three and as many as six additional terms," which was "too much." (ECF No. 137 at 25.) Google again asks the Court to construe "computer-readable medium" in Claim 14 of the '476 patent, but this time not the phrases in the other patents.

Claim 14 of the '476 patent claims a "**computer-readable medium** bearing instructions for providing security . . . ." The '476 patent specification discloses many embodiments of computer-readable media, including non-volatile and volatile media such as optical or magnetic disks, storage devices, main memory, floppy disks, hard disks, magnetic tape, CD-ROMs, RAM, and other physical media. '476, 5:17-25. The '476 patent also discloses that a "computer-readable medium" ***may*** be a transmission medium:

> The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 104 for execution. Such a medium ***may*** take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 110. Volatile media includes dynamic memory, such as main memory 106. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 102. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

'476, 5:4-16 (emphasis added).

The first sentence is definitional: "The term 'computer-readable medium' as used herein refers to any medium that participates in providing instructions to processor 104 for execution." But the second sentence is not definitional. It uses the word "may," which stands in contrast to the "as used herein refers to" language used in the first sentence. "May" is permissive, used to express possibility, but not requirement. The remaining sentences in the paragraphs are not definitions of "computer-readable medium" either, and the language that follows the paragraph does not alter the analysis because it, too, is permissive and non-definitional. '476, 5:17-6:21.

Even if the Court were to regard the language as definitional, it is not controlling. Notwithstanding an express definition in a specification, a court may construe the term more narrowly. *See, e.g.*, *Trading Techs. Int'l, Inc. v. eSpeed, Inc.*, 595 F.3d 1340, 1353-55 (Fed. Cir. 2010) (holding that, despite an express definition in the specification, the district court was correct to make "two important changes to this express definition in construing the word"); *Ecolab, Inc. v. FMC Corp.*, 569 F.3d 1335, 1344-45 (Fed. Cir. 2009) (holding that, although the patent specification provided an express definition of "sanitize" (it "denote[s] a bacterial population reduction to a level that is safe for human handling and consumption"), the term should be construed to "mean that the treated meat has become safe for human handling and post-cooking consumption."). The Court should do so here, and treat "transmission media" as a disclosed but unclaimed embodiment.

Claims directed to a "computer-readable medium" are often referred to as "*Beauregard* claims," after *In re Beauregard*, 53 F.3d 1583 (Fed. Cir. 1995). This type of claim gained popularity after *Beauregard*, in which a patent applicant challenged a Board of Patent Appeals' rejection of software-related claims. Before the Federal Circuit could rule, the United States Patent and Trademark Office changed its position, stating that "computer programs embodied in a tangible medium, such as floppy diskettes, are patentable subject matter under 35 U.S.C. § 101 and must be examined under 35 U.S.C. §§ 102 and 103." *Id*. at 1584. Based on this changed position, the PTO moved to dismiss the appeal, and the Federal Circuit vacated the Board's decision and remanded. *Id*.

Following *Beauregard,* a software invention claimed as a program embodied in a tangible medium has been consistently recognized as statutory subject matter under Section 101. *See* MANUAL OF PATENT EXAMINING PROCEDURE § 2106.01 at 2100-18 (8th ed. 6th rev. 2007) (explaining that "a claimed computer-readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is thus statutory.") (Peters Decl. Ex. E). *Beauregard* claims allow inventors to claim as patentable subject matter: computer programs embodied on a floppy disk, hard disk, CD, DVD, computer memory, and similar storage media.

More recently, the Federal Circuit addressed whether a "signal" was patentable subject matter. *In re Nuijten*, 500 F.3d 1346, 1353 (Fed. Cir. 2007). *Nuijten* concerned an appeal from a decision of the Board of Patent Appeals and Interferences rejecting a claim in an application and so did not concern an issued patent. The PTO had found that a claim to a "storage medium having stored thereon a signal with embedded supplemental data" was a "manufacture" and patentable, but that a claim to "a signal" was not. *Id.* at 1351-53. The court agreed, holding that the "claims on appeal cover transitory electrical and electromagnetic signals propagating through some medium, such as wires, air, or a vacuum. Those types of signals are not encompassed by any of the four enumerated statutory categories: process, machine, manufacture, or composition of matter." *Id*. at 1352.

We expect Google to argue that Claim 14 of the '476 patent is invalid under Section 101 for being drawn to ineligible subject matter based on *Nuijten*. But Claim 14 is not a "signal" claim, and there is no question that a storage medium such as a computer memory is an article of manufacture or a composition of matter, and therefore is patentable subject matter. *See In re Lowry*, 32 F.3d 1579, 1582, 1584-85 (Fed. Cir. 1994). *Nuijten* did not overturn that holding (nor could it, as it is not an *en banc* decision). What the *Nuijten* court did not address is how a district court should construe a "computer-readable medium" claim in an issued patent, when the specification describes both storage and transmission media. As explained below, the correct approach is to construe the phrase to mean only storage media, and preserve its validity.

When construing a patent claim term, it is important to give the claim term "the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention, *i.e.*, as of the effective filing date of the patent application." *Phillips*, 415 F.3d at 1313. In December 1997, when the '476 specification was written, and in 2000 and 2001 when the claims were issued by the PTO after examination, no one thought the claims were not statutory subject matter under Section 101. To the contrary: 1994's *Lowry* and 1995's *Beauregard* ruled the day, and the PTO's position was that "a claimed computer-readable medium encoded with a computer program defines structural and functional interrelationships between the computer program and the medium which permit the computer program's functionality to be realized, and is thus statutory." EXAMINATION GUIDELINES FOR COMPUTER-RELATED INVENTIONS 9 (1996)

(Peters Decl. Ex. F).[1] It was not until more than ten years later that the *Nuijten* court held that a "signal per se" did not belong to any of the four enumerated statutory categories: process, machine, manufacture, or composition of matter. Yet the Federal Circuit did not hold that a "computer-readable medium" is not statutory matter, nor would it be expected to.

"A storage device for use by a computer" is the right construction. Given the choice to construe an issued patent claim to cover statutory or nonstatutory subject matter, it would be unreasonable choose to construe the claim to cover nonstatutory subject matter and risk invalidating the claim. Such a construction would not comport with the intent of the inventor or the PTO's issuance of the patent. Issued patents are presumed valid, and the principle that "claims should be so construed, if possible, as to sustain their validity" applies here. *Rhine v. Casio, Inc.*, 183 F.3d 1342, 1345 (Fed Cir. 1999) (citation omitted); *see also Astra Aktiebolag v. Andrx Pharms., Inc.*, 222 F. Supp. 2d 423, 458 (S.D.N.Y. 2002) ("Whenever a claim is susceptible to one construction that would render it valid and another construction that would render it invalid, the claim will be construed to sustain its validity."). The disclosure of transmission media in the '476 specification does not bar application of the principle, as a court may construe a claim term more narrowly than even an express definition in the specification. *See, e.g., Trading Techs.*, 595 F.3d at 1353-55; *Ecolab*, 569 F.3d at 1344-45.

The issue that "computer-readable medium" presents in the '476 patent is larger than just the one patent: what shall courts do with the thousands of patents issued in the past decade that have *Beauregard* claims and disclose both storage media and transmission media as embodiments? In 2001, the Patent Office instructed its examiners that "a signal claim directed to a practical application of electromagnetic energy is statutory regardless of its transitory nature." MANUAL OF PATENT EXAMINING PROCEDURE § 2106 (8th ed. 2001) (Peters Decl. Ex. G). Many patent attorneys drafted their applications in accordance with this instruction. To this day, the Patent Office instructs that a software invention claimed as a program embodied in a tangible medium is statutory subject matter under Section 101. MANUAL OF PATENT EXAMINING

[1] Available at http://www.uspto.gov/web/offices/pac/dapp/pdf/ciig.pdf.

PROCEDURE § 2106.01 at 2100-18 (8th ed. 6th rev. 2007) (explaining that "a claimed computer-readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is thus statutory.") (Peters Decl. Ex. E).

As the Court considers how to construe this PTO-examined and -issued *Beauregard* claim, entitled to a presumption of validity, it should not "risk destroying the legitimate expectations of inventors in their property" or "unfairly discount the expectations of a patentee who had no notice at the time of patent prosecution." *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 535 U.S. 722, 739 (2002) ("To change so substantially the rules of the game now could very well subvert the various balances the PTO sought to strike when issuing the numerous patents which have not yet expired and which would be affected by our decision."). The '476 patent specification discloses patentable embodiments of computer-readable media—each one "a storage device for use by a computer"—and that is how the phrase should be construed.

One final point: the phrase "computer-readable medium" (and similar phrases like "computer-readable storage medium" and "computer usable medium") are in asserted claims of four other asserted patents. Because Google seeks a construction only for the '476 patent, and because construction of a patent claim depends on that patent's unique intrinsic evidence, the Court should instruct the jury that the "computer-readable medium" phrases in the '104, '520, '720, and '702 patents have their ordinary meaning rather than any special meaning for Claim 14 of the '476 patent that the Court determines is the correct construction for that claim.

//

//

//

//

//

//

//

CONCLUSION

For the foregoing reasons, Oracle respectfully requests that the Court adopt its proposed claim construction for "computer-readable medium" in Claim 14 of the '476 patent, and decline to construe the remaining terms.

Dated: December 9, 2011

MICHAEL A. JACOBS
MARC DAVID PETERS
DANIEL P. MUINO
MORRISON & FOERSTER LLP

By: /s/ Michael A. Jacobs

Attorneys for Plaintiff
ORACLE AMERICA, INC.