UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS
EASTERN DIVISION

RED BEND LTD., and RED BEND SOFTWARE INC., Plaintiffs, v. GOOGLE INC., Defendant.	Civil Action No. 09-cv-11813-DPW

PLAINTIFFS RED BEND LTD. AND RED BEND SOFTWARE INC.'S
OPENING CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Page(s)

CASES

Red Bend hereby submits this memorandum of law in support of its proposed constructions of the following disputed terms: "compact difference result," "invariant references," "data table," "executable program," and "modified (old/new) (program/data table)" (hereinafter the "Disputed Terms").

I. PRELIMINARY STATEMENT

Red Bend proposes constructions of the Disputed Terms that are faithful to the language of the claims in which those terms appear, and are consistent with the usage of those terms in the specification and file history. In contrast, Google proposes constructions that: (1) improperly seek to narrow the full scope of the claim language by adding extraneous limitations from the preferred embodiments or other (unasserted) claims, (2) are inconsistent with the Examiner's understanding of the meaning of the claims as evidenced by the prosecution history, (3) would cause the claims to be so narrow as to fail to cover even the embodiments described in the specification, and (4) would introduce ambiguities into, rather than clarifying the scope of, the claims. Accordingly, Red Bend respectfully requests that this Court adopt Red Bend's proposed constructions.

II. BACKGROUND OF THE INVENTION

Red Bend's U.S. Patent No. 6,546,552 (the "'552 Patent") issued on April 8, 2003, and claims priority to an earlier application filed on August 19, 1998. (Exh. 1,[1] '552 Patent). The '552 Patent relates to updating programs or data tables on an end-user's computing device, such as when software is upgraded and/or security fixes are developed by the software vendor. (*See id*. at 1:22-25 ("an old program is installed at a remote client site and is subject to be upgraded to a new program, where the latter includes some modifications as compared to the

[1] Citations to "Exh. __" refer to the Exhibits attached to the Declaration of Jennifer C. Tempesta submitted herewith.

old program")). At the time of the filing of the '552 Patent, techniques existed for generating updates using a "difference result" or "diff" between the old and new versions of the program, and transmitting only that "diff" to the end-user, rather than re-sending the new computer program in its entirety. Although this was an improvement over techniques that transmitted a complete copy of the new program every time a small update occurred, the inventor of the '552 Patent realized that traditional "difference result" or "diff" approaches were still inefficient. As the inventor noted in his patent, using prior art "diff" techniques "normally results in a relatively large amount of data, even if the modifications that were introduced to the old program (in order to generate the new program) are very few." (*Id*. at 1:56-59). This is so because executable programs are made up of entries containing computer instructions some of which include target addresses or "references"[2] that (for various reasons) identify or "point" to other entries in the program. The inventor realized that the large size of diff files created using prior art techniques "stems from the alterations of reference in reference entries as a result of other newly inserted entries (and/or entries that were deleted)." (*Id.* at 3:31-35).

For instance, a program might include several instructions to invoke a routine stored at a particular location (*i.e.*, at a particular address) in the program, specified by a reference. A simplified example of such a program is shown below, where several entries in the program instruct the processor, via a "CALL" instruction, to invoke machine code stored at a particular location in the code (here, code beginning at address 7 (*i.e.*, INSTRUCTION E)).[3]

---

[2] The parties have agreed that the term "reference" in the claims should be construed as follows: "A part of the data appearing in an entry in the data table which is used to refer to some other entry from the same data table. A reference can be either an address or a number used to compute an address." (Exh. 1, '552 Patent at 2:42-45).

[3] Although the details of this example program are unimportant, it is intended that the CALL instruction would cause the processor to begin execution with the code at the address designated by the numeric reference (in this case "7") following the "CALL" command, and after the

Address	Entries/Machine Code
0	INSTRUCTION A
1	CALL **7**
2	INSTRUCTION B
3	CALL **7**
4	INSTRUCTION C
5	CALL **7**
6	INSTRUCTION D
7	INSTRUCTION E

The problem with traditional diff utilities is that if a single entry were inserted (or deleted) before an entry that was the target of multiple references, those multiple references would also be changed to "point" to the new location of their target entry, causing the diff utility to fail to detect many corresponding entries between the two programs as matching – thereby causing the diff to include the content of those "non-matching" entries in the diff result. For example, if a single insertion were made after the entry at address 6 ("INSTRUCTION D") the references in the entries at addresses 1, 3, and 5 would change from 7 to 8, because the instruction that those multiple calls are intended to invoke (*i.e.*, INSTRUCTION E) has moved down to address 8. The diff utility, which compares the entries (not the addresses) between the old and new programs, would then potentially detect as matches only the entries in new addresses 0, 2, 4, 6 and 8 (corresponding to old entries at addresses 0, 2, 4, 6 and 7). It would reflect those matches as instructions to copy the entry or entries at the addresses in the old programs containing the corresponding entry or entries, as shown:

OLD	
Address	**Entries/Machine Code**
0	INSTRUCTION A
1	CALL **7**

NEW	
Address	**Entries/Machine Code**
0	INSTRUCTION A
1	CALL ***8***

Diff (old address)
Copy (0)
CALL ***8***

---

processor completes execution of the command at that address 7 ("INSTRUCTION E"), it would resume execution with the next instruction in sequence after the most-recently executed "CALL" command. For example, one possible execution sequence for this program would be to execute the instructions at addresses: 0, 1, 7, 2, 3, 7, 4, 5, 7, 6, 7.

OLD	
Address	**Entries/Machine Code**
2	INSTRUCTION B
3	CALL **7**
4	INSTRUCTION C
5	CALL **7**
6	INSTRUCTION D
7	INSTRUCTION E

NEW	
Address	**Entries/Machine Code**
2	INSTRUCTION B
3	CALL ***8***
4	INSTRUCTION C
5	CALL ***8***
6	INSTRUCTION D
7	*INSTRUCTION F*
8	INSTRUCTION E

Diff (old address)
Copy (2)
CALL ***8***
Copy (4)
CALL ***8***
Copy (6)
INSTRUCTION F
Copy (7)

The inventor discovered that the size of the "diff" generated by such prior art techniques could be greatly reduced by performing pre-processing on the old and new programs or data tables, to generate a "modified" old program and a "modified" new program, where the modification occurs such that references in corresponding entries in the old and new programs, which change due to insert/delete modifications, are reflected as "invariant." The diff is then performed on the modified old and new programs or data tables, thereby generating a compact "diff" result. As summarized by the inventor:

> the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs . . . will be reflected as invariant entries in the modified old and new programs. The net effect is that the invariant reference entries . . . will not appear in the difference result, thereby reducing its size as compared to a conventional difference result . . .

(*Id*. at 3:36-46). Accordingly, applying the techniques of one embodiment of the '552 Patent to the example above might result in the following:

MODIFIED OLD	
Address	**Entries/Machine Code**
0	INSTRUCTION A
1	CALL **0**
2	INSTRUCTION B
3	CALL **0**
4	INSTRUCTION C
5	CALL **0**
6	INSTRUCTION D
7	INSTRUCTION E

MODIFIED NEW	
Address	**Entries/Machine Code**
0	INSTRUCTION A
1	CALL **0**
2	INSTRUCTION B
3	CALL **0**
4	INSTRUCTION C
5	CALL **0**
6	INSTRUCTION D
7	*INSTRUCTION F*
8	INSTRUCTION E

Diff (old address)
Copy (0-6)
INSTRUCTION F
Copy (7)

Because the difference utility can detect many more matching entries in this case, the size of the diff would be greatly reduced as compared to the prior art diff, as substantially less information needs to be conveyed in the diff:

Prior Art Difference Result	**Compact Difference Result**
Copy (0)	Copy (0-6)
CALL ***8***	*INSTRUCTION F*
Copy (2)	Copy (7)
CALL ***8***	
Copy (4)	
CALL ***8***	
Copy (6)	
INSTRUCTION F	
Copy (7)	

The ‘552 Patent contains 68 claims directed to different aspects of the invention. Red Bend asserts Claims 8, 9, 10, 11, 12, 21, 22, 23, 24, 25, 28, 29, 30, 31, 32, 33, 34, 42, 43, 44, 45, 46, 55, 56, 57, 58, 59, 60, 62, 63, 64, 65, 66 and 67 of the ‘552 Patent in this action (the “Asserted Claims”). In general, the Asserted Claims involve methods and systems for generating or applying compact difference results between old and new executable programs or between old and new data tables. Exemplary Claim 42 relates to a method of generating a compact difference result involving an old and new data table:

> A method for generating a compact difference result between an old data table and a new data table; each data table including

reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:

(a) generating a modified old data table utilizing at least said old data table;

(b) generating a modified new data table utilizing at least said new data table, said modified old data table and modified new data table have at least the following characteristics:

(i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;

(c) generating said compact difference result utilizing at least said modified new data table and modified old data table.

(*Id*. at 19:63-20:14). The Asserted Claims can be classified as follows (independent claims appear in bold):

	Executable Program Inputs	**Data Table Inputs**
Method	**8**, 9, 10, 11 (Generator)	**42**, 43, 44, 45 (Generator)
	12, 29, 30, 31 (Client Side)	**46**, 63, 64, 65 (Client Side)
System	**21**, 22, 23, 24 (Generator)	**55**, 56, 57, 58 (Generator)
	25, 32, 33, 34 (Client Side)	**59**, 60, 66, 67 (Client Side)
Device	28 [depends from 8-11]	62 [depends from 42-45]

III. PROCEDURAL BACKGROUND

Red Bend previously moved for a preliminary injunction, asserting that Google infringed Claims 8-10, 21-23, 42-44, and 55-57 of the '552 Patent. The parties fully briefed that issue and presented argument during the hearing on April 14th. Since that time, Red Bend has asserted additional claims, and the parties have exchanged their lists of terms to be construed and their respective proposed constructions. In a series of written and telephonic exchanges, the parties met and conferred in an attempt to narrow their disputes regarding the claim constructions. In this process, the parties have agreed on proposed constructions of the following

terms: "difference result," "old executable program," "said old program," "old data table," "reference," and "reference entries." Additionally, the parties have made modifications to their proposed constructions of some claim terms that are at issue in connection with Red Bend's motion for a preliminary injunction. A chart setting forth the parties' agreed upon constructions, and the competing proposals for the construction of the Disputed Terms is attached hereto as Appendix A.

IV. APPLICABLE LAW OF CLAIM CONSTRUCTION

Claim construction is the judicial act of "pronouncing the meaning of claim language as a matter of law . . . ." *Cybor Corp. v. FAS Techs., Inc.*, 138 F.3d 1448, 1454 (Fed. Cir. 1998) (*en banc*). Claim terms are generally given their ordinary and customary meanings, determined by evaluating the meaning a term would have to a person of ordinary skill in the art at the time of the application for the patent. *Phillips v. AWH Corp*., 415 F.3d 1303, 1312-13 (Fed. Cir. 2005) (*en banc*); *CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002) (noting a "heavy presumption" that a claim term should be given its ordinary meaning).

In performing the claim construction analysis, the specification may be used to interpret the claims; but, care must be taken to "avoid the danger of reading limitations from the specification into the claims . . . ." *Phillips*, 415 F.3d at 1323. This follows from the "'bedrock principle' of patent law that 'the *claims* of a patent define the invention to which the patentee is entitled the right to exclude.'" *Id.* at 1312 (emphasis added) (citation omitted). Accordingly, the Federal Circuit has noted that "although the specification often describes very specific embodiments of the invention, we have repeatedly warned against confining the claims to those embodiments." *Id.* at 1323. Instead, the focus must remain on the claim language, because "claims may embrace different subject matter than is illustrated in the specific embodiments in the specification." *Id.* (citation omitted). In other words: "claims, not specification

embodiments, define the scope of patent protection. The patentee is entitled to the full scope of his claims, and we will not limit him to his preferred embodiment or import a limitation from the specification into the claims." *Kara Tech. Inc. v. Stamps.com Inc.*, 582 F.3d 1341, 1347 (Fed. Cir. 2009).

In addition to the specification, the Court may refer to the prosecution history. However, because the prosecution history "represents an ongoing negotiation between the PTO and the applicant, rather than the final product of that negotiation, it often lacks the clarity of the specification and thus is less useful for claim construction purposes." *Phillips*, 415 F.3d at 1317. Further, although the applicant may narrow his claims by making admissions during prosecution, to be effective as such, those statements surrendering claim scope must be "clear and unmistakable" -- otherwise, the ordinary meaning of the terms controls. *Omega Eng'g, Inc v. Raytek Corp.*, 334 F.3d 1314, 1325-26 (Fed. Cir. 2003). *See also Northern Telecom Ltd. v. Samsung Elecs. Co.*, 215 F. 3d 1281, 1294-95 (Fed. Cir. 2000) (refusing to narrow claims based on ambiguous statement in the prosecution history, based on the "heavy presumption in favor of the ordinary meaning of claim language").

Finally, the Court may consider extrinsic evidence, such as the opinions of experts, however this evidence is "in general [ ] less reliable." *Phillips*, 415 F.3d at 1318. As the Federal Circuit noted in its *en banc* opinion in *Phillips*: "undue reliance on extrinsic evidence poses the risk that it will be used to change the meaning of claims in derogation of the 'indisputable public records consisting of the claims, the specification and the prosecution history,' thereby undermining the public notice function of patents." *Id.* at 1319 (citation omitted). Further, "expert reports and testimony is generated at the time of and for the purpose of litigation and thus can suffer from bias that is not present in intrinsic evidence." *Id.* at 1318.

V. ARGUMENT

A. COMPACT DIFFERENCE RESULT

Red Bend Proposal	**Google Proposal**
A difference result of a smaller size as compared to a conventional difference result (obtained by using techniques in existence prior to the invention of the patent-in-suit) *in which the need to reflect changes to references due to delete/insert modifications is reduced or eliminated.*	A difference result in which references that have changed due to delete/insert modifications do not appear.

The inventor drafted his claims in an attempt to capture the broadest contours of his invention, which (in the case of the Asserted Claims) involve generation of modified representations of the old and new programs, such that substantially each reference in corresponding entries between the old and new programs, which are changed due to insert/delete modifications, are reflected as invariant, and then generating (or applying) a "compact difference result" that was created based on the modified old and new representations. (*See* Exh. 1, '552 Patent, Claim 42). Starting with the claims themselves, there is no requirement that this difference result necessarily exclude all "references that have changed due to delete/insert modifications." Moreover, as made clear in the specification of his patent, the words "compact difference result" were used to indicate merely that the inventor's results were smaller (*i.e.*, more "compact") than prior art difference results. (*See id.* at 3:30-46 ("thereby reducing [the difference result's] size as compared to a conventional difference result obtained by using hitherto known techniques"); 14:5-14 ("the resulting **compact** [difference results] as compared to conventional, larger in size difference results, bring about the desired efficient version control") (emphasis added). Those words were not intended to denote any particular limitation regarding what was contained (or not contained) in the difference result. If that is what the inventor intended, he could have said so explicitly, such as by stating "generating said compact

difference result such that said invariant references do not appear." The fact that he did not do so suggests that the invention should not be so-limited. *Kara Tech.*, 582 F.3d at 1347.

Nevertheless, in the interests of narrowing the parties' dispute, Red Bend has agreed, only for the purposes of this litigation, that "compact difference result" can include this further limitation (italicized in the chart above): "in which the need to reflect changes to references due to delete/insert modifications is reduced or eliminated." This limitation is suggested in some places in the specification (*see* Exh. 1, '552 Patent at 3:36-46; 10:3-41), and in the prosecution history of the patent. In particular, in one communication to the Patent Office, the inventor distinguished his invention over prior art techniques, by pointing out that the "need [to include all changed references] is *reduced or eliminated* . . . ." (Exh. 2, RedBend150) (emphasis added).

However, any further narrowing of the claim, as Google proposes, would be wholly inappropriate. First, if Google's construction were adopted, Google would likely argue to the jury that the construction requires *all* references that have changed due to delete/insert operations to be excluded from the difference results. This is contradicted by the claim language itself, which makes clear that only "substantially each" reference in corresponding changed entries need be handled by the claimed techniques. Second, the specification never requires that *all* changed references be excluded from the difference result. Indeed, such a result would likely be impossible, because it would first require that the all references in the program be properly processed, requiring that the executable program be correctly parsed into its constituent entries of instructions and data, which is "very difficult." (Docket No. 60, Edwards Reply Decl. ¶ 39). Third, Google's construction is completely inconsistent with the prosecution history, which makes clear that, at least with regard to certain claims, the invention makes it possible to

generate a compact difference result such that the need to reflect changed references is "*reduced or eliminated.*" (Exh. 2, RedBend150) (emphasis added). Under Google's construction, the references must be *eliminated,* and not merely *reduced* in number, in the difference result. This is directly counter to the teachings of the '552 Patent's prosecution history that the number of changed references merely be reduced. Thus, Google's construction is improperly narrow and not supported by the intrinsic evidence.

B. INVARIANT REFERENCES

Red Bend Proposal	**Google Proposal**
Values made the same.	Values made the same in the modified old and new programs (or data tables) for corresponding reference entries so that the reference addresses are excluded from the difference result.

Both Red Bend and Google agree that the plain meaning of "invariant" is "the same;" accordingly, both parties' proposals for this term include the language: "values made the same."[4] Google, however, asks this Court to further narrow the term, with some language that is already in the claim text (and is therefore surplusage) and which improperly imports a limitation from the specification relating to the generation of the "difference result." Google also asks the Court to include additional language that confuses and blurs the important distinction between a "reference" and an "address." Google's erroneous construction should be rejected and the Court should adopt the plain meaning of invariant references to a person of skill in the art: "Values made the same."

First, Red Bend's construction is consistent with the plain meaning of "invariant" and is consistent with the description of the embodiments in the '552 Patent. The word

[4] Red Bend has slightly modified the proposed construction of this term that it sought during the preliminary injunction phase. The new language more closely tracks the unobjectionable portion of Google's construction. (*See* Edwards 7/15/10 Decl. at n.2).

"invariant" is understood to mean "unvarying, invariable, constant." (Docket No. 60 at 55, Edwards Reply Decl. Exh. A). In the context of the '552 Patent, reflecting as "invariant references in the corresponding entries in said modified old and modified new" programs/data tables means that the data in the modified old and new programs/data tables that represents corresponding references in the original programs/data tables are made to take the same value. As shown above in the example programs (*see* § II), the references in the modified programs are changed to 0, making corresponding references (*e.g.*, the references in old entry at address number 1 and new entry at address number 1) the same value (0). The specification includes a similar description. (Exh. 1, '552 Patent 10:57-60).

Second, Google's proposal that the construction include the language "in the modified old and new programs (or data tables) for corresponding reference entries" is an imprecise paraphrase of language that is already in the claims, and therefore should not be further introduced into the construction of "invariant reference." In particular, the claims already say that the relevant references[5] "are reflected as *invariant references* in the corresponding entries in said modified old and modified new [data tables/programs]." (*See, e.g., id.* at 20:6-11). If the Court were to adopt Google's construction as an instruction to the jury in this case, the jury would likely be hopelessly confused when it attempted to substitute the term "invariant reference" in the claims with Google's construction. The claim language, if so construed, would require the jury to find that the relevant references: "are reflected as values made the same in the modified old and new programs (or data tables) for corresponding reference entries so that the

[5] As used herein, the term "relevant references" refers to "substantially each reference in an entry in said old program/data table that is different than a corresponding entry in said new program/data table due to delete/insert modifications that form part of the transition between said old program/data table and said new program/data table" -- which is recited in each of the Asserted Claims.

reference addresses are excluded from the difference result in the corresponding entries in said modified old and modified new [data tables/programs]." The claim language becomes impossible to parse under Google's construction. For this reason alone, it should be rejected.

More troubling is Google's extraneous addition of the language "so that the reference addresses are excluded from the difference result." That language does not define what an "invariant reference" *is*, but what the *effect* of an invariant reference might be, in accordance with one way of using the inventive techniques. This is legally improper. *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F. 3d 898, 909 (Fed. Cir. 2004) ("[a]bsent a clear disclaimer of particular subject matter, the fact that the inventor may have anticipated that the invention would be used in a particular way does not mean that the scope of the invention is limited to that context") (citation omitted). The full scope of the language of the Asserted Claims include no limitation regarding "excluding" references. Accordingly, such a limitation is legally improper. *Kara Tech.*, 582 F.3d at 1348 ("*The patentee is entitled to the full scope of his claims*, and we will not limit him to his preferred embodiment or import a limitation from the specification into the claims.") (emphasis added).

Moreover, Google's extraneous limitation, which requires that something be "excluded from the difference result" is repetitive of Google's proposed (improper) construction of "compact difference result," which is addressed separately above. (*See* § V.A.). Invariant references are simply values made the same (*i.e.*, "invariant"). It is improper to import any further limitation from the inventor's description in the specification of how he imagined invariant references would be used in different embodiments of his invention. *Liebel-Flarsheim*, 358 F.3d at 909; *Ventana Med. Sys., Inc. v. Biogenex Labs., Inc.*, 473 F.3d 1173, 1181 (Fed. Cir. 2006) ("[E]ach claim does not necessarily cover every feature disclosed in the specification.

When the claim addresses only some of the features . . . it is improper to limit the claim to other, unclaimed features.").

Finally, the use in Google's construction of the phrase "reference addresses" is ambiguous and likely to confuse the jury. The '552 Patent makes a clear and important distinction between *references* and *addresses*. An *address* is a unique physical location within a file or memory where a particular entry is found. (Exh. 1, '552 Patent at 2:37-40). In contrast, a *reference* is data within the contents of a file or memory that refers to some particular *address*. (*Id.* at 2:42-45).

In an old program, there can be many *references* that refer to a single unique *address*. (*See* Edwards 7/15/10 Decl. ¶ 13). In the example shown above (*see* § II), in the old program there are 3 references (in entries 1, 3, and 5) to a single unique target address (address 7). In the new program, the target address has moved, so the old references have changed to point to the updated unique target address (address 8) The inventor's objective in developing the techniques of the '552 patent was to "reduce or eliminate" the appearance in the difference result of *changes to references* so as to reduce the size of the difference result. (Exh. 1, '552 Patent 10:10-15; Exh. 2, RedBend150). In other words, the techniques in the '552 Patent attempt to reduce the effect of those *changed references*, so that instead of including in the difference result *every appearance of a reference* to a particular unique address that has changed, those changed references would appear fewer times (or not at all). (*See* Exh. 2, RedBend150). Because any particular address may be referenced many times in a particular program, it is far more important to the size of the difference result that the appearance of the instances of changed *references* be reduced, not that any particular *address* be excluded. (*See* Edwards 7/15/10 Decl. ¶ 14). Accordingly, Google's construction is inconsistent with the term's plain meaning, the

specification and file history; improperly limits the patentee to a preferred embodiment; and would create confusion and blurs the clear distinction in the patent between *references* and *addresses*. (Exh. 1, '552 Patent at 2:37-45). Google's construction, therefore, should be rejected.

C. DATA TABLE

Red Bend Proposal	**Google Proposal**
A table of entries, where an entry is an addressable unit within the data table. Each entry may have a different size. An executable program is one example of a data table.	A table of entries, each of which may have a different size. An executable program is one example of a data table. It cannot be source or other symbolic code.

The term "data table" is defined by the inventor in the "Glossary" of the '552 Patent. The relevant sections of the Glossary read as follows:

> *Data Table*--a table of *entries*, each may have a different size;
> *Entry*--a *data table* includes *entries*, each of which is an addressable unit that contains data;
>
> ***
>
> As an example, a *data table* can be an executable program either as a loaded program in machine-memory or as an executable-file.

(*See* Exh. 2, RedBend26 (emphasis in original application as filed); Exh. 1, '552 Patent at 2:33-36, 61-63). Red Bend's construction follows directly from these sections of the glossary, and should therefore be adopted. *Vitronics Corp. v. Conceptronic, Inc.*, 90 F. 3d 1576, 1582 (Fed. Cir. 1996) ("the specification acts as a dictionary when it expressly defines terms used in the claims"); *Phillips*, 415 F.3d at 1316 ("the specification may reveal a special definition given to a claim term by the patentee that differs from the meaning it would otherwise possess. In such cases, the inventor's lexicography governs.").

Google seeks to include in the Court's construction this limitation: "It cannot be source or other symbolic code." Such a limitation is inappropriate for several reasons. First, the limitation is not supported by the specification, which *explicitly defines* this term. Nowhere does

the specification say, or even suggest, that a data table cannot be source or other symbolic code. Moreover, this construction would prevent the claims from reading on the embodiments of data tables described in the specification. (Docket No. 60, Edwards Reply Decl. ¶ 7). This is legally improper. *Vitronics*, 90 F. 3d at 1583 ("Such an interpretation is rarely, if ever, correct").

To the extent Google attempts to justify its unduly-narrow construction based on the prosecution history, (*e.g.*, Exh. 2, RedBend151) that attempt should be rejected. The relevant prosecution history section relates only to claim 1 of the '552 Patent, which is not asserted in this action and which is, nonetheless, at best directed to operating on executable programs, not data tables. Accordingly, that section of the prosecution history has no relevance to the proper construction of data table, and is not a clear an unmistakable disavowal of claim scope. (*See also* Docket No. 60, Edwards Reply Decl. ¶ 8-9). Google's proposed construction should therefore be rejected.

D. EXECUTABLE PROGRAM

Red Bend Proposal	**Google Proposal**
A program comprising machine language instructions and corresponding bytes of data used by the program that are ready to be run on a computer.	A program comprising machine language instructions and corresponding bytes of data used by the program that are ready to be run on a computer, excluding source or other symbolic code.

The parties agree on the proper definition of "executable program" except that Google seeks to improperly further limit the definition by including this language: "excluding source or other symbolic code." This additional limitation is not supported by the claim language, or the other intrinsic evidence. All that is required is that the program be a "loaded program in machine memory or…an executable file" -- that is, an *executable* program is a program that is *able* to be *executed* (Exh. 1, '552 Patent 2:61-63).

It is well known in the art that executable files contain some symbolic code. (*See* Docket No. 60, Edwards Reply Decl. ¶¶ 27, 29-33). Indeed, the patent specification itself describes executable files used as inputs to the techniques of the Asserted Claims that contain a "relocation table," which is symbolic. (Exh. 3, Walker Dep. 124:22-125:21; 128:1-128:7; Docket No. 60, Edwards Reply Decl. ¶ 27). Google's construction would therefore preclude the claims from reading on one of the preferred embodiments, a result that is legally improper. *Vitronics*, 90 F. 3d at 1583 ("Such an interpretation is rarely, if ever, correct").

Google has in the past relied on the prosecution history to support its proposed requirement that an executable program be one "excluding source or other symbolic code." But, the prosecution history, which "represents an ongoing negotiation between the PTO and the applicant… often lacks the clarity of the specification and thus is less useful for claim construction purposes." *Phillips*, 415 F.3d at 1317. Additionally, it is legally improper to use the prosecution history to import extraneous claim limitations, as Google proposes. *Bayer AG. v. Biovail Corp.*, 279 F.3d 1340, 1348 (Fed. Cir. 2002) ("[E]xtraneous limitations cannot be read into the claims from the . . . prosecution history.").

Google's argument is also unsupported by the prosecution history excerpt upon which it previously relied. It is true that during prosecution, the applicant noted, with respect to (unasserted) claim 1, that: "In extracting diff between 2 versions of executable files *as defined in amended Claim 1*, there is no source involved, and neither statements, nor any textual or other symbolic representation of the program even exist." (Exh. 2, RedBend151) (emphasis added). This statement, however, does not support Google's proposed construction.

First, the relevant statement was made about claim 1, which is not asserted in this case. When distinguishing the Asserted Claims (*e.g.*, claims 8 and 21), the inventor relied on

other arguments. (Exh. 2, RedBend152-54). Accordingly, the statement regarding claim 1 is irrelevant, and is certainly not a clear disavowal of claim scope of the Asserted Claims. *See Omega Eng'g.*, 334 F.3d at 1330.

Second, the statement in the prosecution history does not go as far as Google's construction suggests -- it states merely that no "textual or symbolic *representation of the program* even exist." (Exh. 2, RedBend151) (emphasis added). Thus, at best for Google, the prosecution history would support a definition that excluded "source or other symbolic representation of *the program.*" In other words, that statement suggests that the claims do not cover instances where the input files represent the *entire* program in source or symbolically. (*See* Edwards 7/15/10 Decl. ¶ 8). Google's construction, however, is far narrower and would preclude a finding of literal infringement if *any* source or symbolic code were in the input file, even where that code was for purposes other than representing the program (or was purposely added simply to avoid literally practicing the claims of the '552 Patent). Such a construction is simply unsupported by the snippet of prosecution history cited by Google.[6]

Relatedly, the statement must be considered in the context is which it was made -- to distinguish the Okuzumi prior art reference. In that context, the statement is nothing more than an explanation of benefits of the techniques provided in the disclosure of the '552 Patent

[6] Contrary to Google's likely suggestion, Red Bend's proposed claim construction would not cause these claims to cover techniques that used only source code or other purely symbolic files as inputs to the algorithm, because such files are not "executable" and would likely not include "references" under the parties' agreed proposed construction of that term. In particular, that construction requires that references in the input executable programs be either an address (which is a number) or a number used to compute an address. In either case, the "reference" must be a number. Source code and symbolic code typically include symbolic references, which are not "resolved" into physical addresses until after they are complied into object code and linked . (*See, e.g.*, Exh. 3, Walker Dep. at 209:14-25). Under Red Bend's proper construction, therefore, the Asserted "Executable" Claims would not cover products that operated on traditional source or symbolic representations of a program.

over Okuzumi. In particular, the comment explains that the techniques of the '552 Patent can work, even where there is no source code or symbolic information. This was a major advantage over Okuzumi, which only works with pure source code files.

In view of the foregoing, Red Bend respectfully submits that the Court reject Google's unwarranted and unsupported addition of the phrase "excluding source or other symbolic code" to the construction of "executable program" and instead adopt Red Bend's proposed construction.

E. MODIFIED (OLD/NEW) (DATA TABLE/PROGRAM)

Claim Term	Claim No.	Red Bend Proposal	Google Proposal
modified old data table	42, 46, 55, 59	An interim result, such as tables or data structures, related to the old data table.	A version of the actual program or data table in its original executable form, with certain portions replaced.
modified old program	8, 12, 21, 25	An interim result, such as tables or data structures, related to the old executable program.	
modified new data table	42, 46, 55, 59	An interim result, such as tables or data structures, related to the new data table.	
modified new program	8, 12, 21, 25	An interim result, such as tables or data structures, related to the new executable program.	

1. Red Bend's Construction Is Supported By the Intrinsic Evidence

These four related terms are treated together herein for simplicity. Red Bend's proposed construction of these terms follows directly from the specification and is supported by the prosecution history.[7]

[7] Red Bend's current proposed constructions of "modified old/new program" and "modified old/new data table" differ from those it proposed at the preliminary injunction stage. This is because only the generator-side claims were at issue then, and those claims call for the "modified old program" to be generated using at least the old program and the "modified new program" to be generated using at least the new program. Hence, Red Bend proposed a construction that the

The specification does not explicitly define these terms, but does explain their meaning by example. In particular, the '552 Patent depicts one possible embodiment of the claimed techniques in Figure 2, where the techniques are applied to "exemplary old and new programs" that are shown together with "various *interim results*." (Exh. 1, '552 Patent at 9:16-17, 9:30) (emphasis added). The specification further explains that the *interim results* in Figure 2 include the modified old and new programs. In particular, the specification, in describing Figure 2, states that "the desired invariant references are accomplished by generating *modified old and new programs* . . . as follows: a) Create P"$_1$ *table* from P_1 . . . and P"$_2$ *table* from P_2 . . . ." (*Id.* at 10:47-60). Numerous other interim results are described as being generated in this process, including "intermediary *data structure*, AL_2." (*Id.* at 13:62-64). Therefore, although the specification does not specifically state which particular ones of the intermediate results shown in Figure 2 constitute the modified old and modified new programs, the specification does clearly describe those modified old and new programs as *interim results* that are *tables* or *data structures*. (*See also* Edwards 7/15/10 Decl. ¶ 10).

The prosecution history further supports Red Bend's construction insofar as it shows that the patent examiner considered a "modified old program" to include a "data structure" related to the old program. In particular, in the Examiner's statement of the reasons for allowance of the '552 Patent, the Examiner cites to the Miller prior art reference as one that shows "scan[ning] the old program and creat[ing] a modified old program, col. 3 lines 1-10 and

---

"modified [old/new] program" be "generated using the [old/new] program." But, for the client-side claims, a reversed process takes place where, for instance, the "modified new program" is *not* generated using the new program, but instead is "reconstituted" using the modified old program and the difference result. The "new program" is then reconstituted using, in part, the modified new program. So, Red Bend's earlier proposals would not work as well in the context of the client-side claims. Accordingly, Red Bend's current proposed constructions are more accurate and insure that like terms are construed consistently throughout all the claims. (*See* Edwards 7/15/10 Decl. ¶ 12).

col. 6 lines 34-44." (*See* Exh. 2, RedBend166). The Examiner also explains in the Supplemental Notice of Allowance that Miller's Text String Index "suffices as the 'modified old program' (again see col. 6 lines 34-66)." (*Id.* at RedBend176). Notably, the portions of the Miller reference cited by the Examiner as disclosing a modified old program relate to a "data structure" derived from the old program. (*See* Exh. 4, Miller at 6:39). Accordingly, the Examiner understood that a modified old program could be any data structure related to the old program -- consistent with Red Bend's construction.

Further, the specification's use of the prime (') and double prime (") notation to distinguish the old program (P_1) from the modified old program (*e.g.*, P'_1 or P''_1) suggests that the inventor intended the modified old program be related to the new program in some fashion. Indeed, this type of prime notation is commonly used in the art to denote things that are derived from or related to each other. (*See* Edwards 7/15/10 Decl. ¶ 10, Exh. B). The manner in which the modified structures are "related" to the old and new programs/data table is apparent in the context of the claim language. For example, in the generation claims (*e.g.*, 8, 21), the "modified new program" is clearly described as being generated "utilizing at least said new program." Accordingly, in the context of those claims, the modified new program and the new program are related in that way. Similarly, in the client-side claims (*e.g.*, 12, 25) the "modified new program" is clearly described as being used to reconstitute the "new program." Accordingly, in that context, the modified new program and the new program are related in that way.

2. Google's Construction Is At Odds With the Claim Language and Other Intrinsic Evidence

a. Google's Construction Is At Odds With the Claim Language

Google's proposed construction of these terms is unsupported by the claims, the specification or by the file history. As to the claims, Google's construction would require that a

modified data table be "in its original executable form." This construction clearly makes no sense when applied to the data table claims (*e.g.*, Claims 42, 46, 55, and 59) because a data table need not be executable at all. (*See* Exh. 1, '552 Patent at 15:9-13 (describing an example of a street map data table, noting "the data table is by no means bound to the representation of a computer program"); *id.* at 3:5-7 (describing an example data table as a group of inter-linked data records)). Google's construction should be rejected for this reason alone.

Additionally, even as to the executable program claims (*e.g.*, Claims 8, 12, 21, 25), Google's construction is flawed because those claims do not require that the "modified old program" or "modified new program" be "in executable form." Instead, the claim clearly states that only the old and new programs that are input to the modification process need to be "executable." When it came time to claiming the modified structures, the applicant did not use the word "executable." Accordingly, there is a strong inference that a "modified old/new program" is not inherently executable. The Federal Circuit's *en banc* ruling in *Phillips* is precisely on point. There, the court noted that "the context in which a term is used in the asserted claim can be highly instructive . . . the claim in this case refers to 'steel baffles,' which strongly implies that the term 'baffles' does not inherently mean objects made of steel." 415 F.3d at 1314. Similarly, the presence of "executable program" in the preamble here "strongly implies" that the "modified old program" and "modified new program" are not executable (or in executable form) as Google proposes. This is further confirmed by the inventor's amendment dated May 8, 2002. On that date, the inventor added the word "executable" to the preamble of the Asserted executable program claims (*e.g.*, Claims 8, 12, 21, 25). (*See* Exh. 2, RedBend160-63). Notably, when the applicant did so, he did not similarly amend the "modified old program" or "modified new program" language, further showing that those terms should not be construed

to be "executable." (*Id.*) For the same reason, the fact that the inventor made no amendment at all to the data table claims at that time confirms that the terms "modified old data table" and "modified new data table" should similarly not be required to be "executable" or in "executable form."

Finally, Google's attempt to inject as a claim limitation the requirement that the modified forms have "certain portions replaced" is inappropriate in light of the language of the unasserted and Asserted claims. None of the Asserted Claims use the word "replacing," as Google's construction would require. In contrast, all of the unasserted independent claims *do* include "replacing" as a required part of the generation of modified old and new programs or data tables. (*See, e.g.*, Claims 1, 5, 14, 18, 35, 39, 48, 52). The fact that the Asserted Claims do not use the word "replacing" is strong evidence that those claims are not limited to techniques involving "replacing" -- contrary to Google's proposed construction. The Federal Circuit faced similar facts in *Kara Tech.*, 582 F.3d 1341. There, the District Court's construction of the asserted claims included a requirement that a "key" be used in the claimed process, despite the fact that none of the asserted claims (but all of the unasserted claims) explicitly required a "key." The court vacated the judgment and remanded, noting: "when the inventor wanted to restrict the claims to require the use of a key, he did so explicitly. None of the claims at issue on appeal recite the term 'key.' By contrast, all of the other independent claims require [a key]." *Id.* at 1347. The same principle applies here. When the inventor wanted to limit his claims to require "replacing" he did so explicitly. But none of those claims are asserted here. Accordingly, Google's proposed construction of these terms should be rejected.

b. Google's Construction Is At Odds With the Specification

In addition to being inconsistent with the clear claim language, Google's construction would prevent the claims from covering any of the embodiments of the invention

described in the specification. "Such an interpretation is rarely, if ever, correct . . . ." *Vitronics*, 90 F.3d at 1583. In particular, Google's construction requires that the modified old and new data tables be in their "original executable form." However, in the exemplary embodiments described in the patent, the generation of the modified old and new programs (*e.g.*, P'_1, P'_2, P''_1 and P''_2) involves "adding label marks" and/or introducing symbolic entries. (Exh. 1, '552 Patent at 10:51-57; Docket No. 60, Edwards Reply Decl. at ¶¶ 7, 10 & Fig. 1). These modified representations of the old and new programs would not be executable, and (at least because they involve the addition of label marks to the data) are not in their "original executable form." (*See* Docket No. 60, Edwards Reply Decl. at ¶¶ 7, 10 & Fig. 1).

c. Google's Construction Is At Odds With the File History

As discussed above, the Examiner, in stating his reasons for allowance, commented that Miller's Text String Index was a "modified old program" within the meaning of the claims. Google's construction is inconsistent with the Examiner's statement because the Text String Index of Miller in no way is a version of the actual [old] program in its original executable form, with certain portions replaced. (*See* Edwards 7/15/10 Decl. ¶ 11). If the Examiner had believed that the claims required the modified old program be "A version of the actual program or data table in its original executable form, with certain portions replaced," as Google proposes, he would have concluded that Miller did not disclose a modified old program. Because he reached the contrary result, it is clear that the Examiner was applying a construction consistent with Red Bend's proposal.

d. Google's Construction Is Legally Inappropriate

Finally, because the purpose of claim construction is to determine the "disputed meanings and technical scope" of patent claims, the proposed construction must resolve the ambiguity so that the parties are not permitted to argue the question to the jury. *O2 Micro Int'l.*

v. Beyond Innovations Tech., 521 F. 3d 1351, 1360-63 (Fed. Cir. 2008) ("When the parties raise an actual dispute regarding the proper scope of these claims, the court, not the jury, must resolve that dispute.") Google's construction fails this test, because its proposal that the modified program or data table be "in its original executable form" is effectively meaningless. (*See* Docket No. 60, Edwards Reply Decl. ¶¶ 15-16 (noting Google's construction is "so vague that just about anything can be argued to be (or not to be) in executable form, rendering the construction effectively meaningless")). Although Google might prefer that this Court adopt a construction that will permit it to argue the meaning and scope of the claims to a jury during a later phase of the case, such a strategy is not legally permitted. *O2 Micro Int'l.*, 521 F.3d at 1361.

VI. CONCLUSION

Because Red Bend's constructions are faithful to the claim language, specification, and prosecution history, and because Google's constructions (1) improperly seek to narrow the full scope of the claim language by adding extraneous limitations from the preferred embodiments or other (unasserted claims), (2) are inconsistent with the Examiner's understanding of the meaning of the claims as evidenced by the prosecution history, (3) would cause the claims to be so narrow as to fail to cover the embodiments described in the specification, and (4) would introduce ambiguities into, rather than clarifying the scope of the claims, this Court should adopt Red Bend's proposed constructions, and reject Google's proposed constructions. Red Bend therefore respectfully requests that this Court enter an Order adopting Red Bend's proposed constructions of the disputed terms.

Dated: July 15, 2010

Respectfully submitted,

By: */s/ Jennifer C. Tempesta*

Daniel Cloherty (BBO# 565772)
Dwyer & Collora, LLP
600 Atlantic Avenue - 12th Floor
Boston, MA 02210-2211
Telephone: (617) 371-1000
Facsimile: (617) 371-1037

Robert C. Scheinfeld (admitted PHV)
Eliot D. Williams (admitted PHV)
Jennifer C. Tempesta (admitted PHV)

Baker Botts L.L.P.
30 Rockefeller Plaza
44th Floor
New York, New York 10012-4498
Telephone: (212) 408-2500
Facsimile: (212) 408-2501

Attorneys for Plaintiffs Red Bend Ltd. and Red Bend Software Inc.

CERTIFICATE OF SERVICE

I hereby certify that this document filed through the ECF system will be sent electronically to the registered participants as identified on the Notice of Electronic Filing (NEF) and paper copies will be sent to those indicated as non-registered participants on July 15, 2010.

By: */s/ Jennifer C. Tempesta*

Jennifer C. Tempesta