Trials@uspto.gov
Tel.: 571-272-7822

Paper No. 10
Entered: December 18, 2015

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SERVICENOW, INC.,
Petitioner,

v.

BMC SOFTWARE, INC.,
Patent Owner.

Case IPR2015-01329
Patent 6,895,586 B1

Before JUSTIN T. ARBES, BRIAN P. MURPHY, and JOHN A. HUDALLA, *Administrative Patent Judges.*

MURPHY, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
37 C.F.R. § 42.108

I. INTRODUCTION

ServiceNow, Inc. ("Petitioner") filed a Petition requesting *inter partes* review of claims 1, 4, and 7 of U.S. Patent No. 6,895,586 B1 ("the '586 patent"). Paper 1 ("Pet."). BMC Software, Inc. ("Patent Owner") filed a Preliminary Response to the Petition. Paper 9 ("Prelim. Resp."). We have statutory authority under 35 U.S.C. § 314(a), which provides that an *inter partes* review may not be instituted "unless . . . there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition."

Petitioner challenges claims 1, 4, and 7 of the '586 patent as unpatentable under 35 U.S.C. § 103. Pet. 3–4. Based on the information presented in the Petition and Preliminary Response, we are not persuaded there is a reasonable likelihood Petitioner would prevail with respect to at least one of the claims challenged in the Petition. Therefore, we decline to institute *inter partes* review.

A. Related Proceedings

The parties identify the following as a related proceeding regarding the '586 patent: *BMC Software, Inc. v. ServiceNow, Inc.*, Case No. 14-CV-00903 JRG (E.D. Tex. Sept. 23, 2014). Pet. 1; Paper 5, 1.

B. The '586 Patent

The '586 patent, titled "Enterprise Management System and Method which Includes a Common Enterprise-Wide Namespace and Prototype-Based Hierarchical Inheritance," issued from an application filed August 30, 2000. Ex. 1001. The '586 patent is directed to a method for managing one or more networked computer systems (an "enterprise"). Ex. 1001, 1:21–22. The method employs a "hierarchical namespace" for efficient referencing and retrieval of

information to manage the enterprise. *Id.* at Abstract, 1:50–52. The hierarchical namespace is a logical arrangement of "objects" in a hierarchical, uniquely addressable system of object names. *Id.* at 1:54–60, 2:12–13. The objects typically relate to the "monitoring and analysis activities of the enterprise," and therefore relate to the hardware and software components in the enterprise. *Id.* at 9:22–26. The Internet is one example of an enterprise namespace in which many individual computer systems are linked together through the use of a hierarchical name structure and addressing system, where "every server name is different from every other server name." *Id.* at 1:61–67.

Figure 6 illustrates the organization of information in a hierarchical namespace, and is reproduced below.

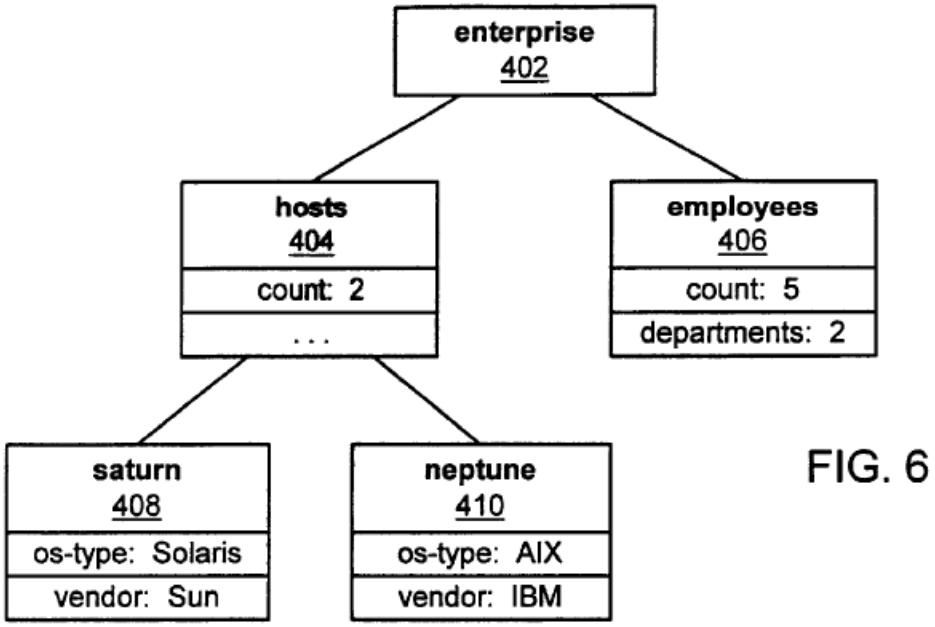


FIG. 6

In Figure 6, above, root object 402 at the top of the hierarchy is named "enterprise," and it has two child objects 404, 406 named "hosts" and "employees," respectively. *Id.* at 13:52–56. "Hosts" 404 has two child objects 408, 410 named "saturn" and "neptune," respectively. *Id.* at 13:56–57. The

vendor for each host is a so-called "object attribute" that is uniquely addressed by name, for example, by the path "hosts/saturn vendor." *Id.* at 14:5–7.

The method of the '586 patent manages the enterprise by sharing objects with one or more applications and/or computer systems in the enterprise. *Id.* at 9:28–32. In particular, the method for sharing objects uses a technique styled "dynamic inheritance." *Id.* at 14:41–15:2. Dynamic inheritance includes the ability of one object "to derive attributes, values, and/or children information from another object, where the attributes, values, and/or children may change over time." *Id.* at 14:57–60. Dynamic inheritance ensures that changes to object data, which reflect changes in the components of the enterprise, are shared among the appropriate components in the hierarchical namespace. *Id.* at 14:52–15:2, Fig. 7.

Figure 7 illustrates the technique of dynamic inheritance, and is reproduced below.

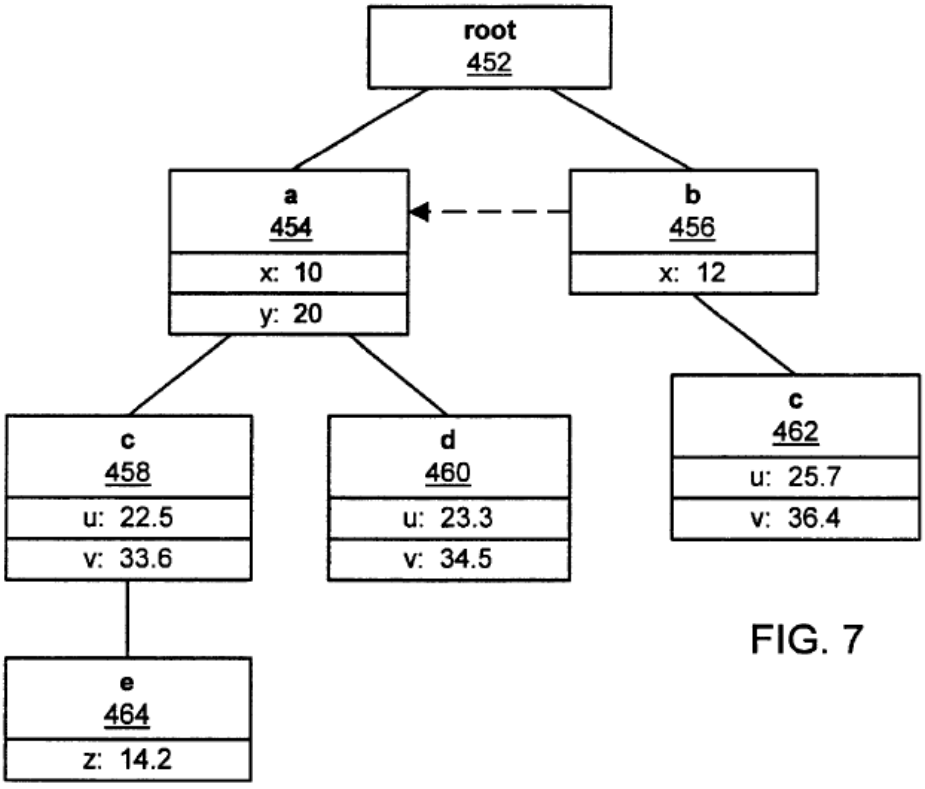


FIG. 7

In Figure 7, above, root object 452 at the top of the hierarchy has child objects 454, 456 named "a" and "b," respectively, which are connected by a dynamic inheritance link represented by the dashed arrow pointing from "b" to "a." *Id.* at 14:51–54. In the example of Figure 7, object "b" will dynamically inherit certain attributes, values, and/or children from object "a." *Id.* at 14:55–15:2. Because object "b," as compared to object "a," already has an attribute named "x" of its own and a child object named "c" of its own, object "b" will inherit only attribute "y" and child object "d" from object "a." *Id.* The '586 patent styles object "a" as a "prototype" object, "from which attributes, values, and/or children are dynamically inherited by another object," and object "b" as an "instance" object "which dynamically inherits attributes, values, and/or children from another object in the namespace." *Id.* at 14:44–49.

C. The '586 Patent Claims

Claim 1 of the '586 patent is illustrative and reproduced below:

1. A method for managing an enterprise, wherein the enterprise comprises one or more networked computer systems, the method comprising:

providing a hierarchical namespace;

adding a plurality of objects to the namespace, wherein the objects relate to software and hardware of the one or more computer systems;

sharing the plurality of objects with a plurality of the one or more computer system, wherein at least one of the objects is a prototype and at least one of the objects is an instance, wherein the instance dynamically inherits traits from the prototype; and wherein the values of the

traits inherited from the prototype change dynamically.

D. Asserted Ground of Unpatentability

Petitioner asserts that claims 1, 4, and 7 of the '586 patent are unpatentable as obvious over Glasser[1] and Davis.[2] Pet. 3–4. Petitioner relies on the Declaration of Dr. David Klausner in support of its arguments. Ex. 1002. Patent Owner opposes. Prelim. Resp. 21–42.

II. ANALYSIS

A. Claim Construction

In an *inter partes* review, we construe claim terms of an unexpired patent according to their broadest reasonable interpretation in light of the patent specification. 37 C.F.R. § 42.100(b); *In re Cuozzo Speed Techs., LLC*, 793 F.3d 1268, 1279–81 (Fed. Cir. 2015). Under the broadest reasonable interpretation standard, we assign claim terms their ordinary and customary meaning, as understood by one of ordinary skill in the art, in the context of the entire patent disclosure. *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). Any special definition for a claim term must be set forth in the specification with reasonable clarity, deliberateness, and precision. *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994).

The '586 patent provides the following definition of the term "object": "As used herein, an object is a self-contained entity that contains data and/or

[1] U.S. Patent No. 5,956,715 issued September 21, 1999 to Glasser et al., on an application filed September 23, 1996, which is a continuation of an earlier application filed December 13, 1994. Ex. 1003 ("Glasser"). Petitioner asserts that Glasser qualifies as prior art under 35 U.S.C. §§ 102(a), (e). Pet. 21.

[2] Davis, selected excerpts from Fred Davis, THE WINDOWS BIBLE, Peachpit Press (1996). Ex. 1004 ("Davis"). Petitioner asserts that Davis qualifies as prior art under 35 U.S.C. § 102(b). Pet. 22.

procedures to manipulate the data. Objects may be stored in a volatile memory and/or a nonvolatile memory." Ex. 1001, 9:19–22. The '586 patent uses lexicography to define the term "object" with reasonable clarity, deliberateness, and precision. Petitioner and Patent Owner both agree to the broadest reasonable construction of "object" as it is defined in the '586 patent. Pet. 14, 16; Prelim. Resp. 8, 10. Therefore, we construe the term "object" as "a self-contained entity that contains data and/or procedures to manipulate the data, which may be stored in a volatile memory and/or a nonvolatile memory."[3]

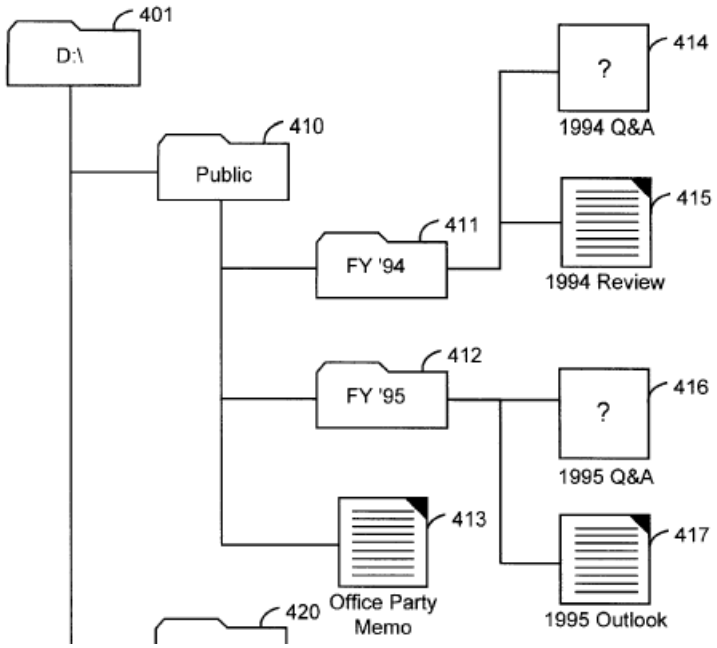
B. Asserted Obviousness of Claims 1, 4, and 7 over Glasser and Davis

Petitioner argues that in Glasser's Windows 95-based file storage system, the "plurality of objects" added to the namespace "takes the form of the plurality of folders and files as shown in hierarchy **400** of Figure 4." Pet. 34.

1. Glasser

A portion of Glasser Figure 4 is reproduced below.

[3] We note that Webster's dictionary defines "self-contained" as "complete in itself." http://www.merriam-webster.com/dictionary/self-contained (Ex. 3001).



Glasser Figure 4, above, shows an example of a hierarchical namespace with a parent folder named "Public" (410) having child folders named "FY '94" (411) and "FY '95" (412), and a file named "Office Party Memo" (413). Ex. 1003, 6:59–64, Fig. 4. Glasser states: "Files can store data, programs, and other computer information, and can be manipulated as objects . . . . Folders . . . are used to collect related files together." Ex. 1003, 4:28–35; Ex. 1002 ¶ 75. Glasser's folders and files are stored on hard disk 121, a nonvolatile memory. Ex. 1003, 3:33–36, 4:50–54, 6:58–59, 7:11–12, Fig. 2A; Ex. 1002 ¶¶ 36, 37. Davis discloses details about Windows 95, including how files and folders relate to hardware and software. Ex. 1004, 68 (84), 137 (310), 215 (498).

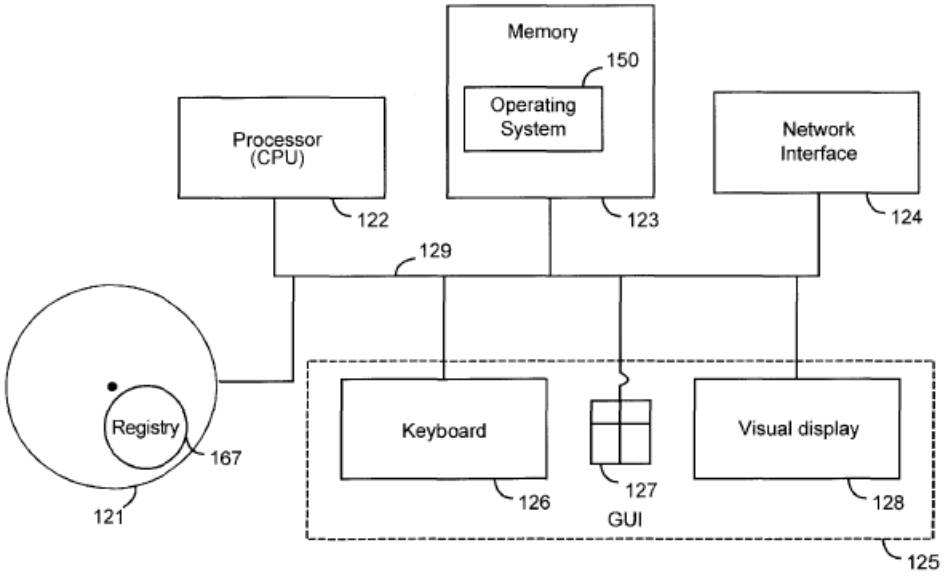
In view of the above facts, Petitioner asserts that Glasser's folders and files qualify as "objects" defined in the '586 patent. Pet. 34–36.

2. Analysis

Glasser limits access to folders and files by using access control lists ("ACLs") to specify user privileges. Ex. 1003, 1:56–58, 7:7–9. Petitioner asserts that "each folder object in Glasser can include an access control list (ACL)," and

"[a] folder object in Glasser can inherit values from the access control list (ACL) of its parent folder." Pet. 43. Petitioner asserts, therefore, that Glasser's system of folders, files, and ACLs satisfies the limitations of claim 1 that "at least one of the objects is a prototype and at least one of the objects is an instance," "the instance dynamically inherits traits from the prototype," and "the values of the traits inherited from the prototype change dynamically." Pet. 43–46 (citing Ex. 1003, 7:5–10, 7:13–25, 7:28–32 (Fig. 4), 9:58–10:29 (Fig. 9); Ex. 1002 ¶¶ 84–88). As Patent Owner points out, however, for security reasons, the ACLs associated with each folder are stored in a separate registry 167 on hard disk 121, not as part of or with the folders and files themselves. Prelim. Resp. 30–31 (citing Ex. 1003, 4:60–64); Ex. 1003, 7:11–12, Fig. 2A/2B.

Registry 167 is depicted in Glasser Figure 2A, reproduced below.

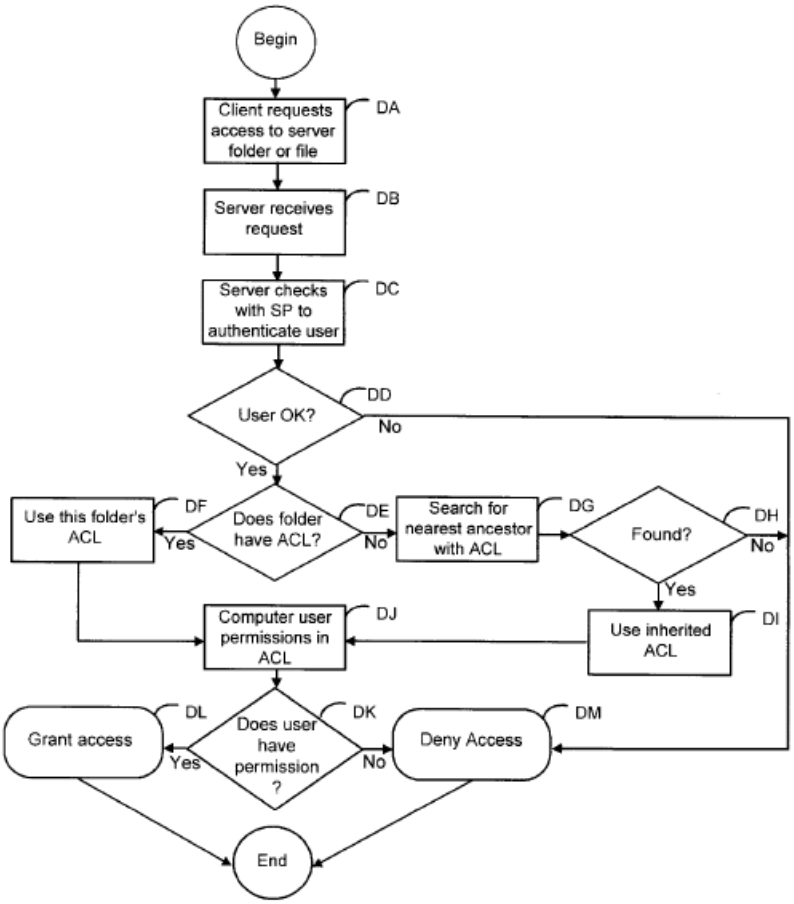


Registry 167, in Figure 2A, above, is a "configuration database" used by the operating system to maintain various kinds of system information. Ex. 1003, 4:58–59. Figure 2A depicts storage registry 167 as a separate location on hard disk 121. File security component 166 of the operating system (Figure 2B) uses registry 167

"to store access permissions (access control lists) for resources of peer server 120, such as file folders." *Id.* at 4:60–63. Petitioner and Dr. Klausner do not address the import of the ACLs being stored in registry 167. Pet. 43–46; Ex. 1002 ¶¶ 84–88.

Patent Owner argues that, because ACLs are the only inherited "traits" disclosed in Glasser, ACLs must be self-contained within folders or files to satisfy the construction of "object" as a "self-contained entity." Prelim. Resp. 30–31. Patent Owner argues that the claimed dynamic inheritance of self-contained objects is not disclosed in Glasser, because the ACLs are stored in registry 167 under control of file security component 166, separate and apart from the folders and files stored on hard disk 121. *Id.* at 30 (citing Ex. 1003, 4:50–54, 4:60–64); Ex. 1003, 7:11–12, Figs. 2A, 2B. Patent Owner argues, therefore, that Glasser's files and folders "are not self-contained with their associated ACL" and cannot qualify as "objects" of "dynamic inheritance," because the ACL attribute values are inherited from the ACL registry, not from a file or folder itself. *Id.* at 30–31. Patent Owner further argues that Glasser does not disclose the recited "sharing the plurality of objects with a plurality of the one or more computer system[s]," because the ACLs are not shared along with the folders and files identified by Petitioner as the "objects" in the hierarchical namespace of Glasser's system. *Id.* at 36–38.

Patent Owner supports its argument with reference to Glasser Figure 9. Prelim. Resp. 36–38. Glasser Figure 9 is reproduced below.



Glasser Figure 9, above, is a flowchart depicting the steps for a system user to access a folder or file "in a resource hierarchy having associated ACLs." Ex. 1003, 9:58–60.[4]

With reference to steps DA–DD, a client user who requests access to a folder or file must be authenticated before being granted access to the requested folder or file. *Id.* at 9:61–10:3. User authentication first involves security components 165 and 168 (Fig. 2B) checking with security provider 190 to determine whether the user is a valid user of the system. *Id.* at 9:65–10:2. If the user is a valid system user (step DD), then the system determines whether the user

[4] The ACLs "associated" with folders and files are those stored in registry 167, managed by file security component 166. Ex. 1003, 7:11–12, Figs. 2A, 2B.

is authorized to access the requested folder by searching for the ACL associated with the folder (steps DE, DF). *Id.* at 10:4–6. If the requested folder does not have its own associated ACL, then peer server 120 searches the hierarchy of folders to find the nearest ancestor folder having an ACL (step DG). *Id.* at 10:6–8. "If an ancestor is found (step DH), its ACL is inherited (step DI) . . . . Peer server **120** performs steps DE through DI using file security component **166**." *Id.* at 10:8–13.

Glasser Figure 2B, the relevant portion of which is reproduced below, depicts the use of file security component 166.

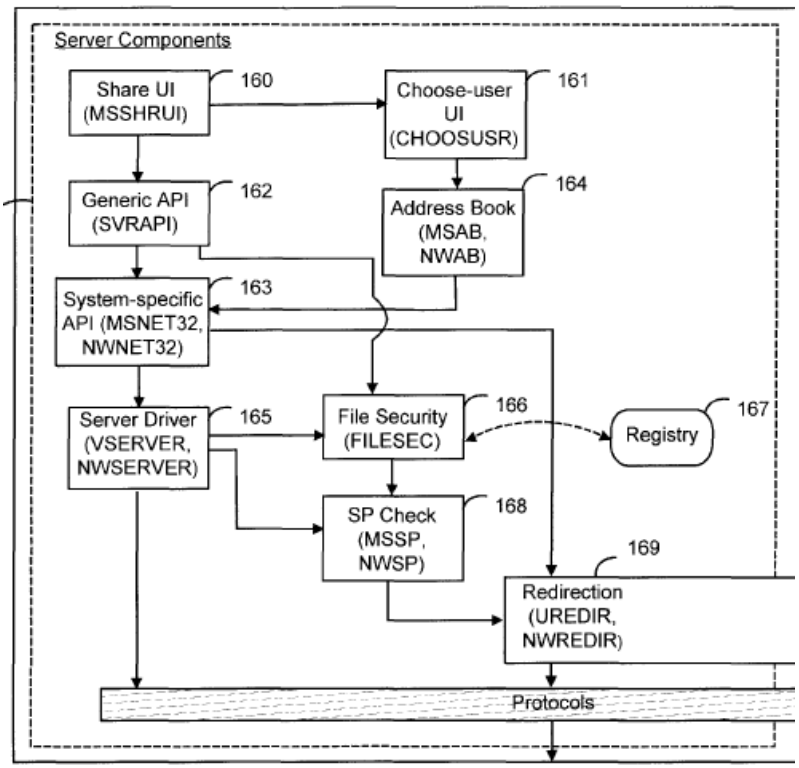


FIG. 2B

With reference to Figure 2B, above, file security component 166 "checks file folder access permissions" stored in registry 167. *Id.* at 5:54–55, 5:66–6:1. Therefore, security component 166 checks registry 167, rather than the ancestor folder itself stored on the hard disk, to locate any ACL associated with the ancestor

folder that may be "inherited" for purposes of granting or denying user access to the requested folder. *Id.*; *see id.* at 4:58–65 ("[F]ile security component **166** . . . uses registry **167** to store access permissions (access control lists) for resources of peer server **120**, such as file folders . . . . Registry **167** can be stored as shown on hard disk **121**.") Only after a user is authenticated will access to the folder itself be provided. *Id.* at 10:25–27. Although Petitioner and Dr. Klausner address other aspects of Glasser Figure 9, they do not address the disclosed protocol for user authentication and inheritance of ACLs from registry 167 using security component 166. Pet. 41, 43, 50–51.

We agree with Patent Owner. Petitioner's evidence does not establish that the ACLs in Glasser's system are stored in a folder or file within a hierarchical namespace. To the contrary, the evidence shows that Glasser stores the ACLs in registry 167, *separate* from the hierarchically organized arrangement of folders and files that Petitioner identifies as the asserted "objects" defined in the '586 patent. *See* Pet. 34 (arguing that the Windows 95 "folders and files [in Glasser and Davis] qualify as 'objects'"), 35 ("Glasser and Davis confirm that folders and files are self-contained entities that contain data, and therefore, qualify as 'objects' that were added to the namespace."). Petitioner also acknowledges that "[a] folder object in Glasser can inherit values *from the access control list* (ACL) of its parent folder" (Pet. 43 (emphasis added)), which is not the same thing as inheriting ACL values from the parent folder. Therefore, Petitioner has not provided sufficient evidence to establish that i) the ACLs are contained in Glasser's hierarchy of folders and files, which are the recited "objects" identified by Petitioner, and ii) the ACLs are "shar[ed] . . . with a plurality of the one or more computer system[s]," because the ACLs are not dynamically inherited by a child folder (the recited "instance" object) from a parent folder, i.e., "from the prototype" object, as recited

in claim 1 of the ’586 patent.

For the reasons given above, we are not persuaded by Petitioner that Glasser or Davis discloses the self-contained “objects” defined in the ’586 patent or “sharing the plurality of objects with a plurality of the one or more computer system[s]” by “dynamic[] inherit[ance],” as recited in claim 1. Claims 4 and 7 depend from claim 1. Therefore, for the same reasons, we are not persuaded by Petitioner’s obviousness argument regarding claims 4 and 7.

III. CONCLUSION

Petitioner has failed to demonstrate a reasonable likelihood of prevailing with respect to at least one of the claims challenged in this Petition, based on the ground asserted and information presented therein.

IV. ORDER

For the reasons given, it is

ORDERED that the Petition is denied.

FOR PETITIONER:

Heidi Keefe
hkeefe@cooley.com

Phillip Morton
pmorton@cooley.com

Andrew Mace
amace@cooley.com

Mark Weinstein
mweinstein@cooley.com

FOR PATENT OWNER:

Robert Cote
rcote@mckoolsmith.com

Pierre Hubert
phubert@mckoolsmith.com

Robert Auchter
rauchter@mckoolsmith.com

Kevin Schubert
kschubert@mckoolsmith.com