

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,

Petitioner,

v.

HEADWATER RESEARCH LLC,

Patent Owner.

Case No. IPR2026-00049

U.S. Patent No. 9,179,359

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 9,179,359**

Table of Contents

I.	Introduction.....	1
II.	Relief Request and Grounds	2
III.	The '359 Patent.....	3
A.	Overview	3
B.	Prosecution History	5
IV.	Level of Ordinary Skill.....	6
V.	Claim Construction.....	6
VI.	Ground 1: Shell and Cole Render Obvious Claims 1 and 26.....	8
A.	Overview of the Prior Art.....	8
1.	Shell.....	8
2.	Cole	17
B.	Motivation to Combine Shell and Cole with a Reasonable Expectation of Success.....	20
C.	Independent Claim 1	24
1.	[1pre]: “A wireless end-user device, comprising:”.....	24
2.	[1a]: “a wireless wide area network (WWAN) modem to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN;”	26
3.	[1b]: “a wireless local area network (WLAN) modem to receive and transmit Internet data between the device and at least one WLAN when configured for and connected to the at least one WLAN; and”	30

4.	[1c]: “one or more processors configured to apply a stored WWAN differential traffic control policy to Internet data service provided using the WWAN modem and the at least one WWAN, and”	34
5.	[1d]: “indicate to a particular application, via an application program interface (API), one or more network access conditions based on the applied WWAN differential traffic control policy,”	40
6.	[1e]: “wherein the one or more network access conditions includes a network access condition that indicates the unavailability to the particular application of an Internet data service that is currently available via the WWAN modem to a different application, but that is indicated as unavailable to the particular application.”	47
D.	Claim 26: “The wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.”	59
VII.	Ground 2: Shell, Cole, and Flack Render Obvious Claim 1	63
A.	Overview of Flack	63
B.	Independent Claim 1	65
1.	[1pre]-[1d]:	65
2.	[1e]: “wherein the one or more network access conditions includes a network access condition that indicates the unavailability to the particular application of an Internet data service that is currently available via the WWAN modem to a different application, but that is indicated as unavailable to the particular application.”	66

VIII.	Ground 3: Shell, Cole, and Flack Render Obvious Claim 26	71
A.	Claim 26: “The wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.”	71
IX.	Mandatory Notices.....	77
A.	Real Parties-in-Interest	77
B.	Related Matters.....	77
C.	Lead and Back-Up Counsel.....	78
D.	Service Information	83
X.	Grounds for Standing.....	83
XI.	Conclusion	83

TABLE OF AUTHORITIES

	Page(s)
Cases	
<i>Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.</i> , 868 F.3d 1013 (Fed. Cir. 2017)	6-7
<i>KSR v. Teleflex</i> , 550 U.S. 398 (2007)	24
Statutes	
35 U.S.C. § 102	2
35 U.S.C. § 103	2

LIST OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 9,179,359 to Raleigh et al. (“the ’359 patent”)
1002	Prosecution History of U.S. Patent No. 9,179,359
1003	Declaration of Dr. Kevin Butler
1004	Curriculum Vitae of Dr. Kevin Butler
1005	U.S. Pat. App. Pub. No. 2004/0224668 (“Shell”)
1006	U.S. Pat. App. Pub. No. 2008/0080458 (“Cole”)
1007	U.S. Pat. App. Pub. No. 2008/0159175 (“Flack”)
1008	International Publication No. WO 03/100581 (“Dive-Reclus”)
1009	David Flanagan, O’Reilly & Associates, Inc., “Java in a Nutshell,” 1996, ISBN: I-56592-183-6
1010	U.S. Pat. App. Pub. No. 2009/0210537 (“Irwin”)
1011	Korhonen, “Host Identity Protocol (HIP) Implementation in the Symbian Environment,” Master of Science Thesis, Tampere University of Technology (12/10/2007) (“Korhonen”)
1012	U.S. Pat. App. Pub. No. 2006/0039354 (“Rao”)
1013	S. Lee & N. Golmie, “Power-Efficient Interface Selection Scheme using Paging of WWAN for WLAN in Heterogeneous Wireless Networks,” 2006 IEEE International Conference on Communications (Istanbul, 2006), 1742–1747 (“Lee”)
1014	U.S. Patent No. 9,215,613
1015	<i>Headwater Research LLC v. AT&T Services Inc.</i> , 2:23-CV-00397-JRG-RSP (Claim Construction Order)
1016	A.-J. Moerdijk and L. Klostermann, “Opening the networks

Exhibit	Description
	with Parlay/OSA: standards and aspects behind the APIs.” IEEE Network, vol. 17, no. 3, pp. 58-64, May-June 2003 (“Moerdijk”)
1017	Richard Stevens et al., “UNIX Network Programming Volume 1, Third Edition: The Sockets Networking API,” 2004, ISBN: 0-13-141155-1
1018	<i>Samsung Electronics Co., Ltd. v. Headwater Research LLC</i> , IPR2024-01407, Paper 9 (P.T.A.B. Mar. 20, 2025) (Institution Decision)
1019	<i>Samsung Electronics Co., Ltd. v. Headwater Research LLC</i> , IPR2024-01407, Paper 17 (P.T.A.B. Mar. 20, 2025) (Director Review Decision)

I. Introduction

Google LLC (“Petitioner”) requests *inter partes* review and cancellation of claim 26 of U.S. Patent No. 9,179,359 (“the ’359 patent”).

Patent Owner disclaimed claims 1-25 and 27-29 of the ’359 patent, rendering claim 26 the sole surviving claim. EX1002, 1. Claim 26 recites a wireless device with at least one processor configured to determine that a “different application is an active application based on a user interaction with the different application.” EX1001, 108:11-14. Determining an application is “active” based on “user interaction” with the application was not novel. Indeed, the asserted prior art discloses this unremarkable activity.

Shell and Cole or Shell, Cole, and Flack render obvious claim 26. For example, Shell and Flack both disclose claim 26’s determination of an active application based on user interaction with the application in the same way as the ’359 patent. Shell discloses network connection requests made in response to a user interacting with an application. Flack explains determining whether a connection request is active based on a user’s interaction with an application. As applied in the combinations, Shell and Flack’s disclosures demonstrate that claim 26 does not recite anything novel or nonobvious. Indeed, the Board previously instituted an IPR of the ’359 patent, finding that Shell and Cole render claim 26

obvious, before the IPR was vacated for discretionary reasons not applicable here.

EX1018; EX1019.

As shown by the grounds below, these references render claim 26 unpatentable.

II. Relief Request and Grounds

The following prior art is presented in this Petition:

Exhibit	Prior Art	Filed	Published	Prior Art Under at Least Pre-AIA 35 U.S.C.
EX1005	Shell (U.S. 2004/0224668 A1)	Feb. 1, 2002	Nov. 11, 2004	§102(a), (b), (e)
EX1006	Cole (U.S. 2008/0080458 A1)	Sept. 29, 2006	Apr. 3, 2008	§102(a), (e)
EX1007	Flack (U.S. 2008/0159175 A1)	Dec. 29, 2006	July 3, 2008	§102(a), (e)

The specific grounds of unpatentability include:

Ground	Claim	Statutory Basis	Prior Art
1	26	§103	Shell and Cole
2	1	§103	Shell, Cole, and Flack
3	26	§103	Shell, Cole, and Flack

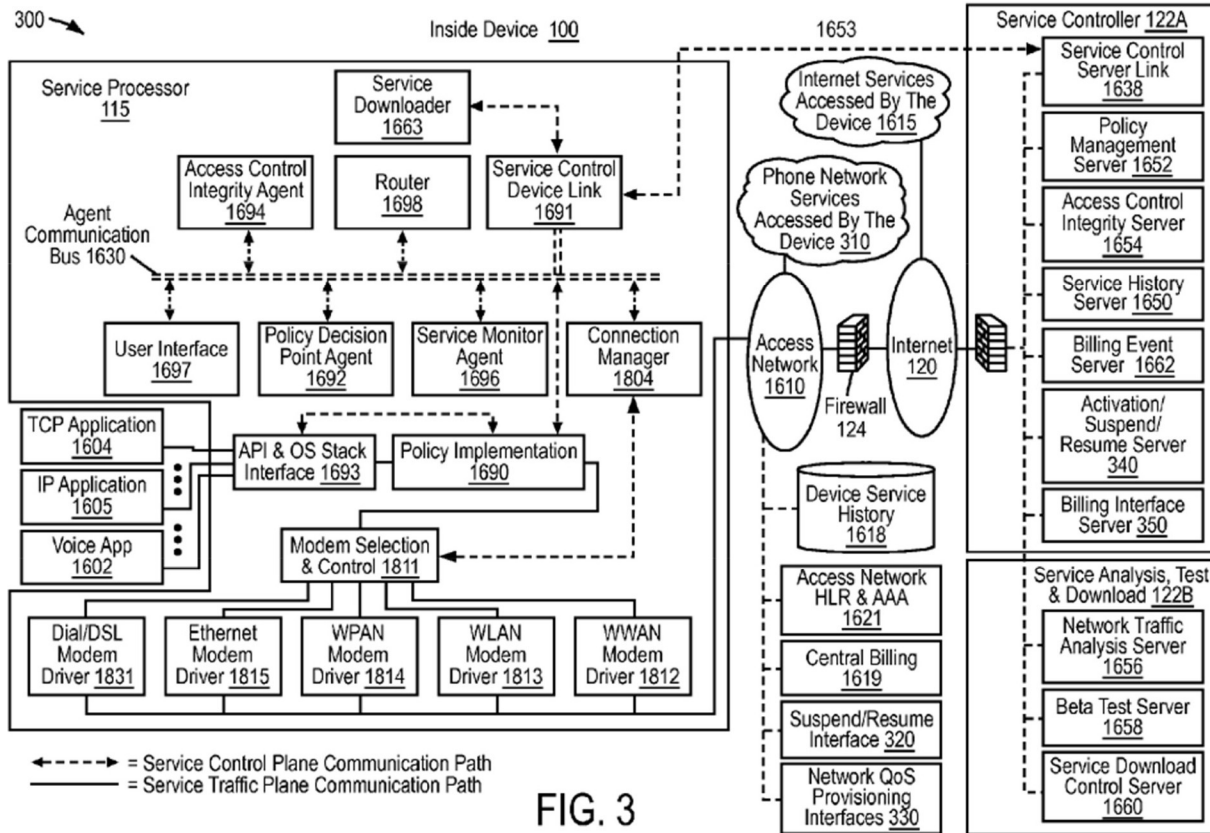
III. The '359 Patent

The '359 patent was filed March 30, 2015, and its earliest asserted priority date is January 28, 2009. EX1001, codes (21), (60).¹

A. Overview

The '359 patent is directed to a “wireless end-user device with differentiated network access status for different device applications.” EX1001, Title. The patent explains that “a wireless end-user device has a wireless wide-area network (WWAN) modem and supports multiple wireless network types,” which can include a wireless local area network (“WLAN modem”), as illustrated below. EX1001, Abstract, FIG. 3. The '359 patent teaches that “in wireless networks, managing the wireless access connection capacity and network access connection resources is important to maintain network performance as network resources/capacity demand increases.” EX1001, 3:44-4:5.

¹ Petitioner does not concede the claims are entitled to this priority date (or Critical Date), but the relied-upon prior art references predate it. Petitioner reserves the right to challenge the priority date and to respond if Patent Owner attempts to antedate any of the prior art.



EX1001, Fig. 3.

To manage capacity, the device can “apply a service usage control policy for the network service usage activities to protect network capacity (e.g., reduce network capacity demand).” EX1001, 11:16-12:2. The device can “assign[] a priority to a network service usage activity and control[] traffic associated with the network services usage activity based on the assigned priority” and can provide a “differentiated and dynamic background services classification ... as a function of network availability state and/or network busy state.” *Id.*, 12:3-10. The service usage control policy can assist “in network access control of network service usage activities (e.g., deferring some or all of the network capacity demand from these

source activities).” *Id.*, 12:11-27. The device can associate “the network service usage activity with a network service usage control policy” based on factors such as “a network busy state, a time of day, ..., an application behavior, ..., a power state of device, a device usage state, a time based criteria, availability of alternative networks, and a set of policy rules.” *Id.*, 85:9-20. When a policy applies to an access request, the device can take various actions including acting to “throttle, delay, priority queue, time window, suspend, quarantine, kill, remove, and other well known traffic control techniques.” *Id.*, 13:10-30.

To enforce the controls, the device includes “a policy implementation driver/agent 1690 (e.g., policy enforcement point driver/agent)” and a policy decision point agent 1692. EX1001, FIG. 3, 57:34-58:17. A person of ordinary skill in the art (“POSITA”) would have recognized policy decision points (PDPs) and policy enforcement points (PEPs) as terms of art describing components that determine whether a policy applies (PDP), and when the policy applies, enforcing the policy decision (PEP). EX1008, ¶20.

B. Prosecution History

Application Serial No. 14/673,625 was filed with 1 original claim, which was replaced in a preliminary amendment, adding the 29 claims that later issued. EX1002, 745-749. The Examiner allowed these claims without issuing any Office Action. *Id.*, 179. Shell and Flack were not before the Examiner. *See* EX1001,

Cover. While Cole was not directly before the Examiner, references related to Cole were listed on an Information Disclosure Statement. *Id.*; EX1002, 213, 229, 278, 286, 587, 605. The Examiner, however, never applied Cole (or references related to Cole) in any substantive rejection. EX1002, 179.

IV. Level of Ordinary Skill

A person of ordinary skill in the art (“POSITA”) at the ’359 patent’s earliest claimed filing date would have had at least a bachelor’s degree in computer science, electrical engineering, or a similar field, and approximately two years of industry or academic experience in a field related to computer software development and/or computer networking. EX1003, ¶53. More related education would compensate for fewer years of work experience (and vice versa). *Id.*

V. Claim Construction

Claims must be construed only to the extent necessary to resolve a controversy. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d

1013, 1017 (Fed. Cir. 2017). Here, no terms need construction to resolve the controversy in this forum.²

Petitioner notes that the '359 patent has been and is involved in district court and PTAB proceedings. *See* Section IX.B. No terms in the '359 patent have been construed. Petitioner notes, however, that the term “differential traffic control policy” is found in at least one other Headwater patent (U.S. Patent No. 9,215,613, EX1014) that claims priority to some of the same applications as the '359 patent. EX1014, cover. The '613 patent was asserted in *Headwater Research LLC v. AT&T Services Inc.*, 2:23-CV-00397-JRG-RSP, and the parties presented constructions related to the term “differential traffic control policy.” EX1015, 21-22. The court construed the term to mean a “rule for controlling network traffic that distinguishes between two or more things.” *Id.* Headwater also argued that a “differential traffic control policy” does not include “preventing background untrusted applications from running” because “no network traffic is generated or requested by those [non-running] applications.” *Samsung Elec. Co. Ltd. v.*

² Petitioner reserves the right to argue limitations in the challenged claim are indefinite, lack written description support, or are invalid for other reasons not presented here.

Headwater Research LLC, IPR2023-01253, Paper 6 at 20 (P.T.A.B. Dec. 22, 2023).

VI. Ground 1: Shell and Cole Render Obvious Claim 26

Claim 26 is the only active claim of the '359 patent. Petitioner also addresses the obviousness of claim 1 because claim 26 depends from claim 1.

A. Overview of the Prior Art

1. Shell

Shell focuses on “[i]ncreasing the level of automation when establishing and managing network connections.” EX1005, Abstract. This is accomplished using “connection manager” modules that “relieve[] users from having to know detailed information about [a] remote resource and [a] destination network.” EX1005, Abstract. Specifically, “[w]hen the connection manager receives a request to access a remote resource,” it uses “extensible decision logic to identify a most appropriate connection method for connecting to a destination network.” EX1005, Abstract. Through its connection manager modules, the Shell system “abstract[s] connection information from a user so as to limit required user interactions when creating network connections.” EX1005, ¶3; EX1003, ¶56.

Shell recognizes that “conditions associated with a network may make establishment of a connection using a particular connection method undesirable at certain times,” e.g., if a network is experiencing “heavy usage” or “high degree of

latency.” EX1005, ¶7. This is because a user may not be “aware of the costs associated with different types of connections” and thus may use a “less desirable connection method to access a resource.” EX1005, ¶7. Beyond this issue, Shell attributes “[o]ther problems associated with connection methods [that] also exist at the programming level.” EX1005, ¶9. Because “[m]ost connection methods utilize different Application Program Interfaces (‘APIs’) when establishing a connection[,]” they may require different APIs or “different parameters,” which “[r]equires programmers to know the nuances of different connection methods” and thereby may be “time consuming” and lead to reduced efficiency. EX1005, ¶9. Shell thus focuses on “automating the establishment and management of various kinds of network connections in a manner that reduces the amount of information a user must know to establish a connection.” EX1005, ¶11. In this way, “[a] user does not need to identify the underlying properties of the network where the resource is located or a connection method for getting to that network.” EX1005, ¶12; EX1003, ¶55.

During operation, “an application included in a computing device may request access to a remote resource located in a destination network.” EX1005, ¶33. “A connection manager” (or “connection planner”) “analyzes a variety of factors to determine the most appropriate connection method to connect to the destination network.” EX1005, ¶33. Examples of factors include “local resources

currently available on the computing device, the priority of the request, or connection parameters (such as latency, cost, and bandwidth) associated with a connection method.” EX1005, ¶33. “When the most appropriate connection method is known, a connection service provider associated with the connection manager may cause a system level [API] to create a connection from a local computing device to the destination network.” EX1005, ¶33; EX1003, ¶56.

Figure 1 shows an example of a “suitable computing environment” for the techniques disclosed throughout Shell. EX1005, ¶37.

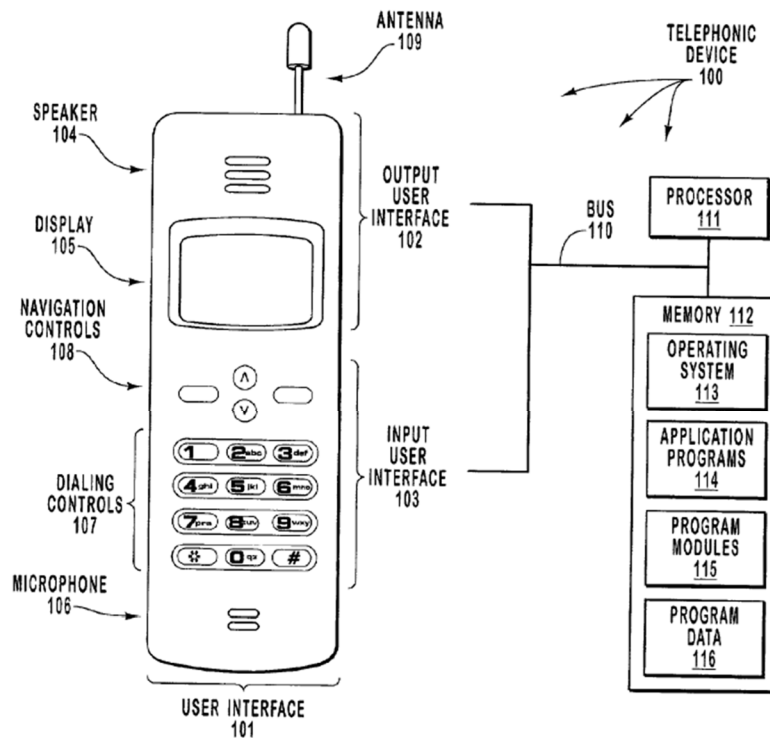


FIG. 1

EX1005, Fig. 1.

The operating environment includes a “telephonic device 100” as a “general-purpose computing device” with standard computing elements for “complex and flexible general-purpose processing capabilities.” EX1005, ¶¶39, 38-44; EX1003, ¶¶57-58.

Figure 2 shows an “arrangement” of “functional components” that “offers a level of abstraction between application modules that may request a connection, and the system level APIs that actually establish the connections.” EX1005, ¶45.

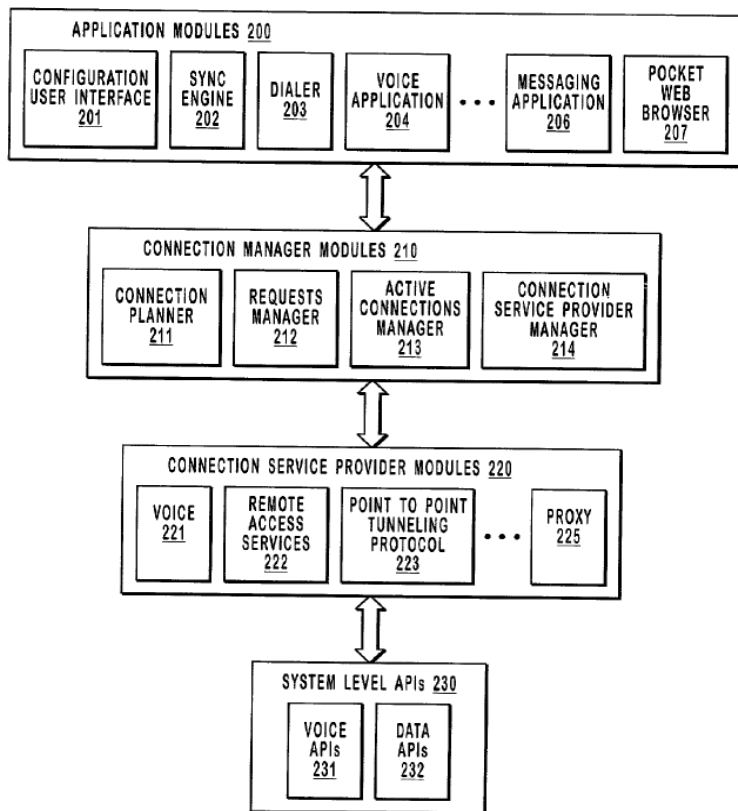


FIG. 2

EX1005, Fig. 2.

Specifically, “[c]onnection manager modules 210 and connection service provider modules 220 may be configured as APIs that receive standard data when being called by another module.” EX1005, ¶46. These modules “include computer-executable instructions that cause, directly or indirectly, the appropriate calls to be sent to system level APIs 230[,]” thereby simplifying the process of establishing connections for application modules 200. EX1005, ¶46. Further, “APIs associated with connection manager modules 210 and connection service provider modules 220 may be integrated into protocol stacks.” EX1005, ¶48. Thus, “APIs associated with connection manager modules 210 and connection service provider modules 220 may be called instead of APIs associated with conventional protocol stacks.” EX1005, ¶48; EX1003, ¶¶59-60.

Further, “connection manager modules 210 may cause resources to be released from and allocated to connection methods.” EX1005, ¶92. “When this occurs, connection manager modules 210 may notify application modules 200 of such changes[,]” e.g., “application modules may be notified that a connection will be established, that a connection is currently active, that connection is being disconnected, that no connection is currently active, that a connection to a resource is blocked for some reason....” EX1005, ¶92. Connection manager modules 210 may also “provide connection specific parameters to application modules 200

including, bandwidth, latency, power consumption, cost, physical location, past connection successes, or other parameters....” EX1005, ¶92; EX1003, ¶61.

Figure 5A shows an example of a technique of network connection request handling based on prioritization of network connection requests. EX1005, ¶57.

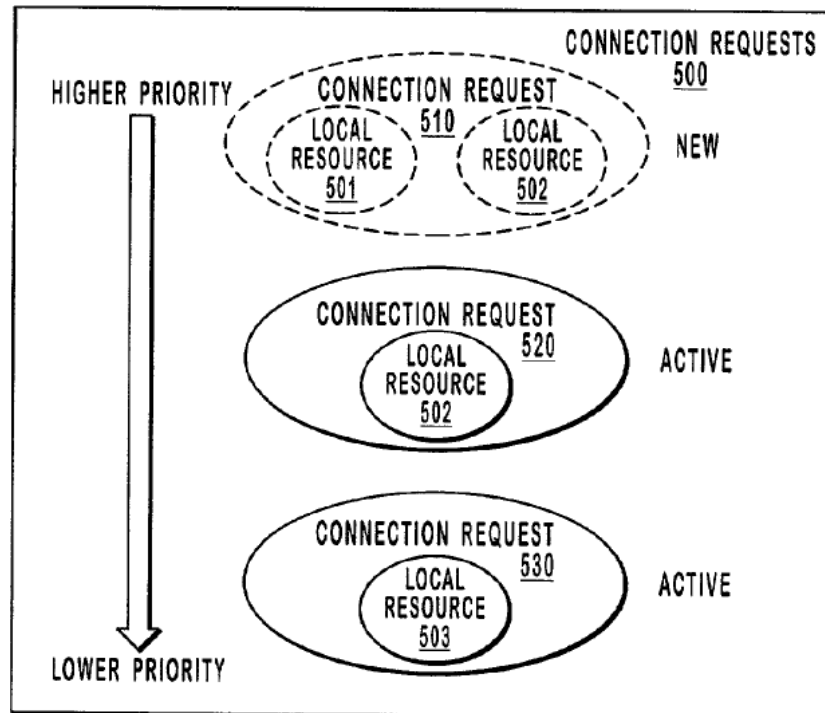


FIG. 5A

EX1005, Fig. 5A.

In the example above, “connection requests 500 includ[e] new connection request 510 and active connection requests 520 and 530.” EX1005, ¶57.

“Connection requests 510, 520, and 530 are prioritized, with connection 510 having the highest priority and connection 530 having the lowest priority.”

EX1005, ¶57. In this example, “requests from sync engine 202 may be assigned a

higher priority than requests from messaging application 206[,]” though “[p]riorities may also be assigned to connection requests based on the current state of applications associated with the connection requests[,]” e.g., “a request from an application that is currently active may be assigned a higher priority than a request from an application running in the background.” EX1005, ¶57. Other examples of prioritization are “based on characteristics associated with the connection requests[,]”: e.g., “voice requests may be assigned a higher priority than data requests.” EX1005, ¶57; EX1003, ¶62.

As depicted in Figure 5B, Shell’s connection manager allocates resources (e.g., provides an application with access to a network) based on the priority associated with a connection request. EX1005, ¶¶60-66, Fig. 5B.

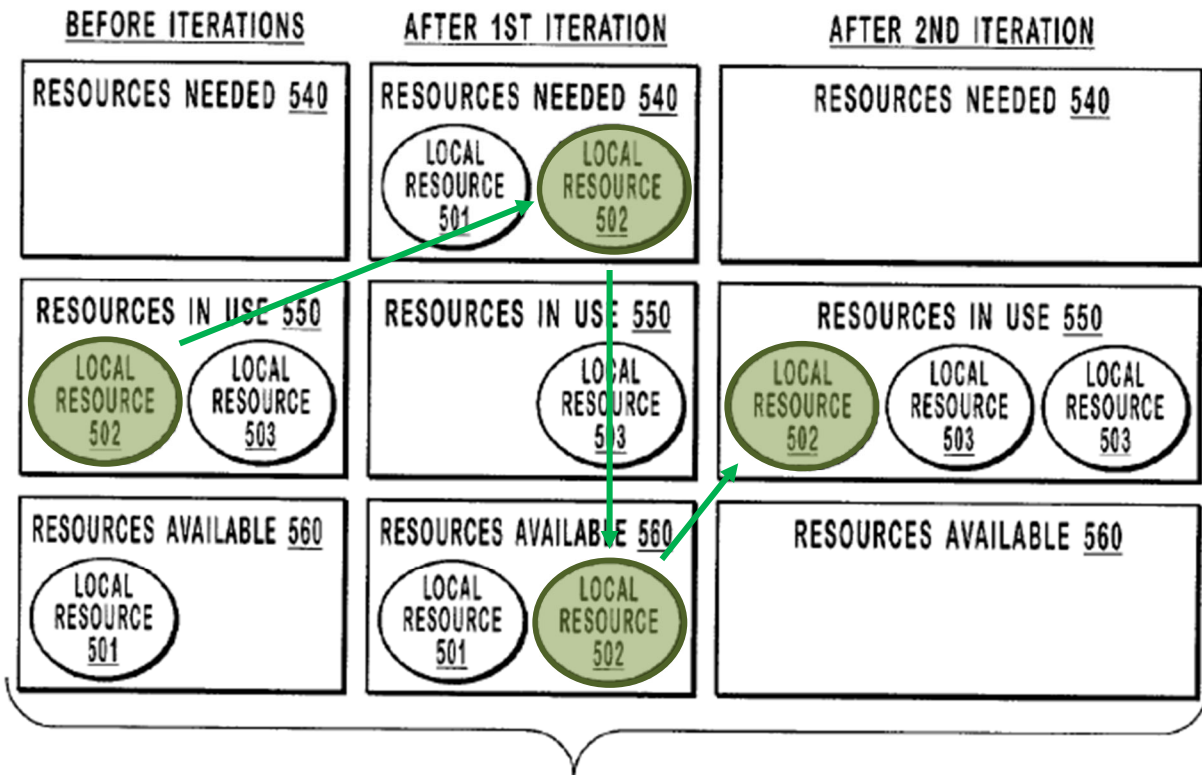


FIG. 5B

EX1005, Fig. 5B (annotated).

For example, at the start of the process (“Before Iterations”), resources **502** and 503 are in use (by connection requests 520 and 530), and resource 501 is available. EX1005, ¶60. When a new connection request 510 is made (first iteration), the connection manager registers that new connection request 510 needs local resources 501 and **502**. EX1005, ¶¶61-66. As the process continues, the connection manager weighs the priority of the new request 510 against the active connections 520 and 530 to determine how to allocate the resources. EX1005, ¶67. Because new request 510 has a higher priority than request 520, the connection

manager disconnects the application associated with request 520 to resource 502. EX1005, ¶67. The connection manager then allocates resources 501 and 502 to new request 510, which provides the application associated with new request 510 a connection. EX1005, ¶67. The above process describes that resource 502 is made unavailable to connection request 520 but is available to new request 510. EX1005, ¶67; EX1003, ¶63.

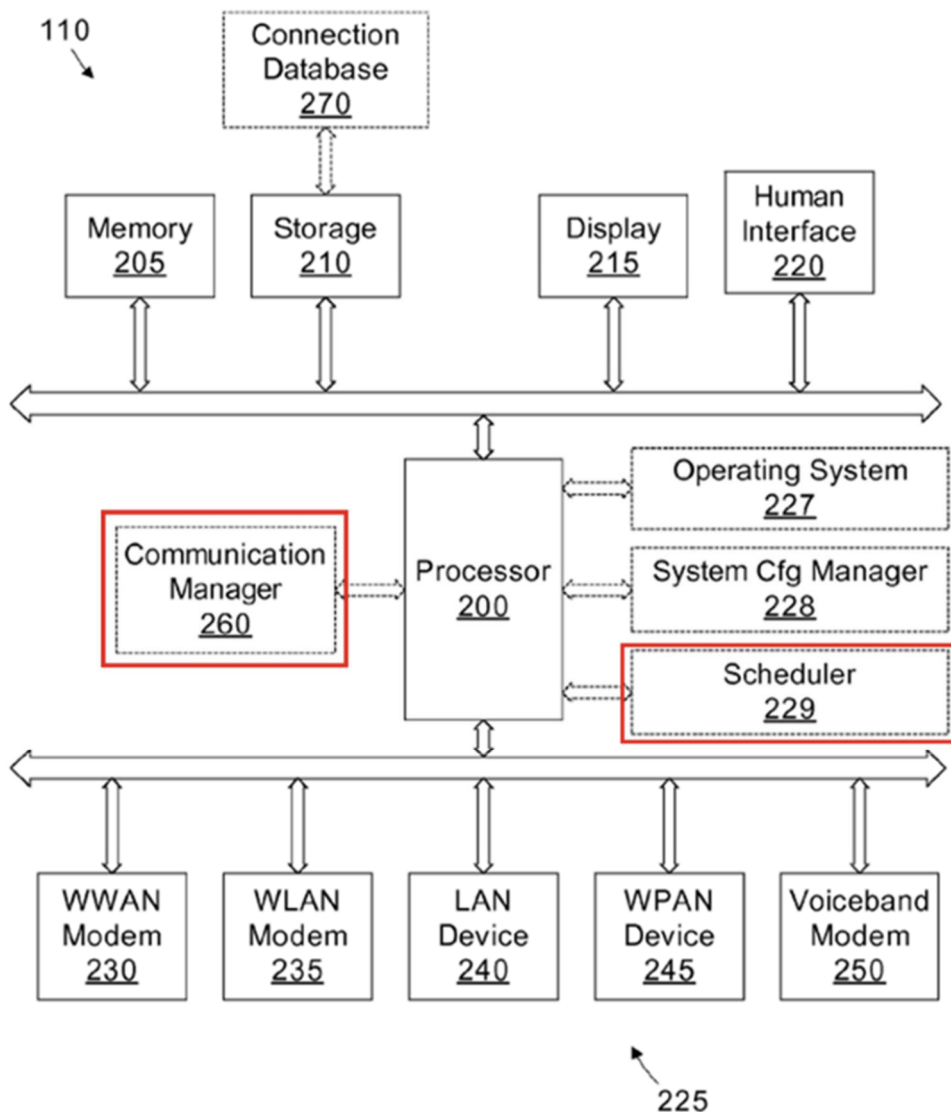
Shell further teaches embodiments in which “an application may notify connection [planner] 211 it wishes to ‘piggy back’ on other connection requests,” i.e., “an application may request to only access resources in a destination network when another application is already accessing resources.” EX1005, ¶70. In such embodiments, “[a]pplications that wish to piggy-back, may call APIs associated with connection manager modules 210 and be given an open handle.” EX1005, ¶70. Open handles are not always granted; for example, they may be blocked if a resource is unavailable. EX1003, ¶¶64, 109, 116-117. The open handle “facilitates notification back to the application that a connection on which the application may want to piggy back has been established.” EX1005, ¶70; EX1003, ¶64.

Shell is analogous art to the '359 patent. For example, Shell is in the same field of endeavor: management of an application's ability to connect to wireless networks. *Compare* EX1005, Abstract, ¶¶57-67, *with* EX1001, Abstract. Shell is also reasonably pertinent to at least one problem addressed by the '359 patent:

optimizing use of data on an application-by-application basis. *Compare* EX1005, ¶¶69-70, 12, *with* EX1001, 4:65-5:20. EX1003, ¶65.

2. Cole

Cole describes “[a] method for configuring a mobile device including a plurality of communication interfaces.” EX1006, Abstract, ¶14. A “connection manager 260” selects “communication interfaces” for connecting to remote networks (e.g., a wireless local area network (WLAN), a wireless wide area network (WWAN), and a wireless personal area network (WPAN)). EX1006, ¶¶15, 3, 31, 35, 36, Figs. 1, 2. The connection manager can also “schedule various events” using a “scheduler” (also annotated) to “effectively use available bandwidth.” EX1006, ¶¶74, 36, 118, 119. This includes taking action when “monitored loads” exceed a “predetermined threshold.” EX1006, ¶¶75, 77; EX1003, ¶66.



EX1006, Fig. 2 (annotated).

The connection manager manages communication interfaces to “determine which connections should be active at any given time.” EX1006, ¶36. This determination includes monitoring various network connectivity metrics, such as battery power. EX1006, ¶¶36, 69, 80; EX1003, ¶67.

To assist in evaluating communication interfaces, the connection manager “maintains data related to each of the communication interfaces,” including “connection name” and “connection technology type.” EX1006, ¶37. This information may be “based on historical data[,] ... user specified parameters, or static parameters,” and may also be “monitored” by the connection manager. EX1006, ¶38; EX1003, ¶68.

When “the monitored load exceeds a predetermined threshold (e.g., 85% of capacity), the connection manager 260 may take one of several actions to manage the load.” EX1006, ¶75. Applications can be “aborted,” “postpone[d],” cancelled and rescheduled, and “block[ed].” EX1006, ¶¶80, 85-90, 125. These actions can also be taken in response to mobile device states, including a “power state.” EX1006, ¶¶90-98. “By monitoring the load on an active connection, the connection manager 260 tries to proactively prevent increased latency from an overloaded connection from impacting the performance of the running applications or the efficiency of the user.” EX1006, ¶75; EX1003, ¶69.

Cole is analogous art to the ’359 patent. For example, Cole is in the same field of endeavor: management of an application’s ability to connect to wireless networks. *Compare* EX1006, ¶¶68, 73, 80, *with* EX1001, Abstract. Cole is also reasonably pertinent to at least one problem addressed by the ’359 patent:

optimizing use of data on an application-by-application basis. *Compare* EX1006, ¶¶73-74, 80, 12, *with* EX1001, 4:65-5:20. EX1003, ¶70.

B. Motivation to Combine Shell and Cole with a Reasonable Expectation of Success

While Shell’s prioritization techniques efficiently utilized network resources in mobile devices, power management is also critical, at least because applications cannot execute without sufficient power. EX1003, ¶76. Network transmissions from a mobile device can cause high power utilization, especially using some types of power-intensive network connections (e.g., WWAN). EX1006, ¶11. Thus, when a high rate of network connection requests are provided by applications, power demand can increase, highlighting the importance of power management. EX1003, ¶76; EX1006, ¶¶11, 78. Indeed, Shell recognizes the need for power management in mobile devices, explaining that factors used to identify most appropriate connection techniques include “power consumption” and “current battery status,” among others. EX1005, ¶16.

By the Critical Date, one well-known technique for mobile device power management was selectively adjusting between available network connections (e.g., WWAN, WLAN). EX1003, ¶77; EX1013, 1. In addition to reducing power consumption, this technique could also increase available bandwidth, reducing network congestion. EX1003, ¶77; EX1006, ¶78. WLANs typically provide more

bandwidth and involve lower cost relative to that involved with use of WWANs (and further involves less power usage). EX1003, ¶77. This would have motivated a POSITA to seek solutions beyond Shell. EX1003, ¶77.

A POSITA would have looked to references directed to controlling network connectivity for mobile devices to address related concerns, e.g., cost, power, network selection. EX1003, ¶78; EX1006, Abstract, ¶¶14-15. Cole offers Internet-access control techniques that address excess power consumption and selection of an appropriate network. EX1006, ¶¶78, 83, 88, 98. In various examples, Cole determines whether to apply an Internet access control (connecting using the “best” connection technique) based on a condition (power state change of the device and high load on the current communication link from device usage). EX1006, ¶¶83, 98. Integration of these techniques into Shell’s device would have therefore allowed power saving when determining network-access connectivity. EX1003, ¶78.

In operation, the Shell-Cole device first evaluates network and power conditions using Cole’s techniques to determine whether to apply an Internet-access control to connect to the network with the “best” connection type, then evaluates further conditions (e.g., whether an application is running in the foreground or background, which application is running, etc.) using Shell’s techniques to determine the priority of the packets traveling over the network

selected by the Shell-Cole device using Cole's techniques. EX1003, ¶79; EX1005, ¶92; EX1006, ¶¶11, 78, 83, 88, 98.

A POSITA would have incorporated functionality of Cole's communication manager (communication manager 260) into Shell's connection manager modules (connection manager modules 210). EX1003, ¶80. The Shell-Cole device would have included a network interface containing WWAN Modem 230 (supporting 2G and 3G cellular technologies) and WLAN Modem 235 (supporting Wi-Fi) from Cole. EX1003, ¶80; EX1005, Fig. 1; EX1005, ¶31. A POSITA would have been motivated to do so because, using such connections, the Shell-Cole device would establish and manage "various kinds of network connections." EX1005, ¶32; EX1003, ¶80. In other words, it would have improved flexibility in establishing connections. EX1003, ¶80. Indeed, to the extent not explicitly disclosed in Shell, the modification to Shell achieves the ability to connect to these wide area networks, allowing the Shell-Cole device to send and receive data packets required for Internet services like web browsing, email, and media content. EX1003, ¶80. A POSITA would have further been motivated to incorporate functionality of Cole's communication manager (communication manager 260) into Shell's connection manager modules (connection manager modules 210) to ensure the mobile device was efficient from a power and battery usage standpoint. While Shell teaches techniques of selecting connection techniques based on "power consumption" and

“current battery status,” EX1005, ¶16, Cole provides more implementation details and optimizations. *See, e.g.*, EX1006, ¶¶87, (“determining if sufficient batter[y] power is present to complete the transaction”), 89-90, (connection manager 260 changing configurations “in direct response to the power state of the mobile device 110”), 92, (considering “power and thermal characteristics”), 91-94 (considering whether “AC power” is available). The Shell-Cole device would have otherwise remained unchanged, including elements found in Shell’s mobile device, such as a processor (e.g., Shell’s CPU 111), non-transitory memory (e.g., Shell’s memory 112), and a display (e.g., Shell’s display 105). EX1005, Fig. 1, ¶¶38-44. These elements are similarly found in Cole. EX1006, Fig. 2, ¶34. The processor of the Shell-Cole device would have been configured to perform the operations discussed in Ground 1 because it “may allow for complex and flexible general-purpose processing capabilities.” EX1006, ¶40; EX1003, ¶81.

A POSITA would have had a reasonable expectation of success in incorporating power management functionality (as taught by Shell and explained in further detail by Cole) into the connection manager modules taught by Shell. EX1003, ¶82. Such a combination would have required only routine programming knowledge well within the skill of a POSITA before the Critical Date. EX1003, ¶82. Indeed, the change would have amounted to nothing more than the use of a known technique to improve similar devices—in each instance a mobile device

equipped with network interfaces—in a similar way and combining prior art elements according to known methods to yield the predictable results described above. EX1003, ¶82; *KSR v. Teleflex*, 550 U.S. 398, 417 (2007); EX1005, ¶¶36, 40; EX1006, Fig. 2, ¶¶34-35. A POSITA would have known that processors commonly run processes such as those run by the software modules described in Shell and Cole. EX1003, ¶82. Therefore, the combination of the techniques of Cole into Shell in the Shell-Cole device would require only routine programming knowledge. EX1003, ¶82.

C. Independent Claim 1

1. [1pre]: “A *wireless end-user device*, comprising:”

To the extent the preamble is limiting, Shell-Cole renders it obvious. EX1003, ¶¶83-84. Shell’s Figure 1 depicts a telephonic device 100 (*wireless end-user device*):

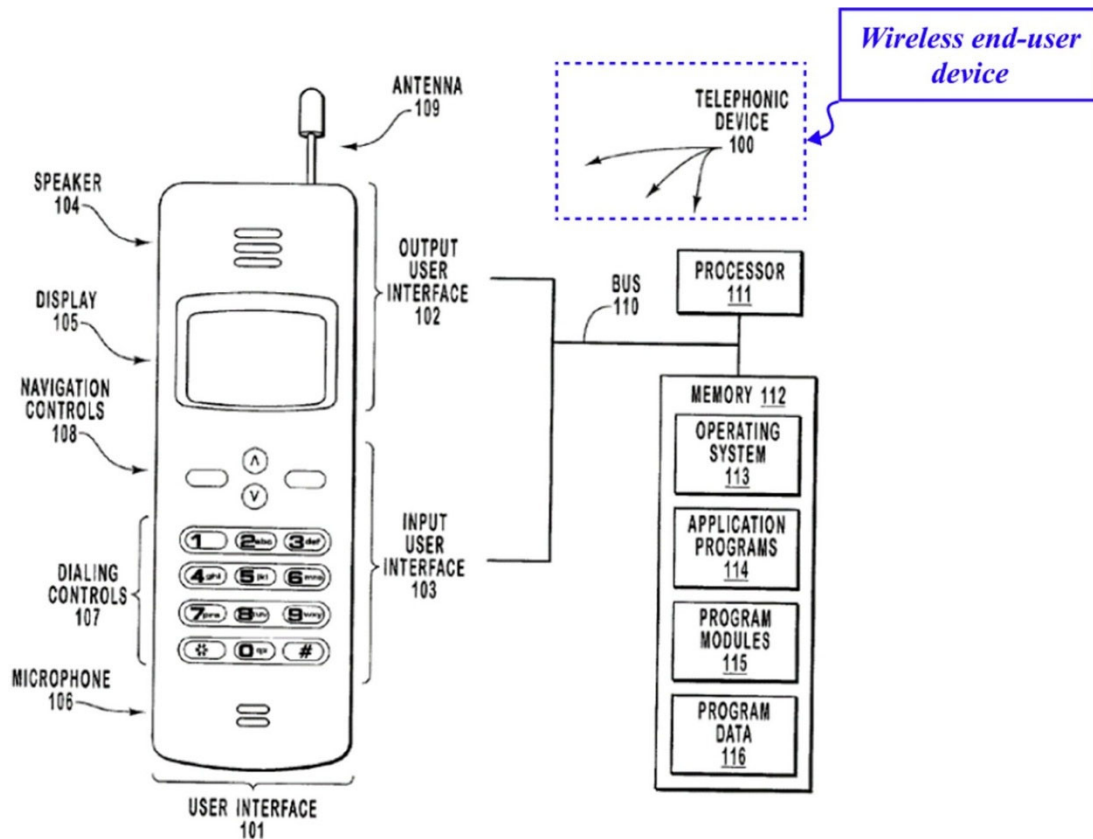


FIG. 1

EX1005, Fig. 1 (annotated).

The telephonic device 100 can be any device running a “suitable operating environment for the principles of the invention,” including “a general-purpose computing device.” EX1005, ¶38. Further, the telephonic device 100 may be implemented in “many types of computer system configurations,” such “personal computers, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, pagers, and the like.” EX1005, ¶36. The

telephonic device 100 also has “an antenna 109 if the telephone device 100 has wireless capabilities.” EX1005, ¶38; EX1003, ¶¶84.

2. [1a]: “a wireless wide area network (WWAN) modem to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN;”

Shell-Cole renders [1a] obvious based on disclosures in Shell and Cole regarding network connectivity. EX1005, ¶¶38, 55, 61; EX1006, ¶¶31, 35, 88, Fig. 2; EX1003, ¶¶85-90.

Shell discloses that telephonic device 100 includes various features that enable the device to establish a wireless network connection. EX1005, ¶¶20, 38. Shell instructs that a “wireless modem” is an example of how devices can establish a wireless connection. EX1005, ¶¶10, 13. Shell provides that “wireless carriers may define logic for accessing resources included in networks under their control[,]” suggesting connectivity of the telephonic device 100 with a cellular network (*wide area network*). EX1005, ¶55. This logic manages local resources, “such as a **modem** or network adapter” in the device, to establish connections using those resources. EX1005, ¶61. Although Shell explicitly discloses the use of modems, it does not expressly disclose a WWAN modem.

Cole discloses a mobile device with different modems for wireless connectivity. In Cole’s Figure 2 (below), a mobile device includes a “WWAN

modem 230” (*wireless wide area network (WWAN) modem*) providing connectivity to a “wireless wide area network (WWAN) 130,” confirming that such a modem configuration was well-known by the Critical Date. EX1006, ¶35; EX1003, ¶86.

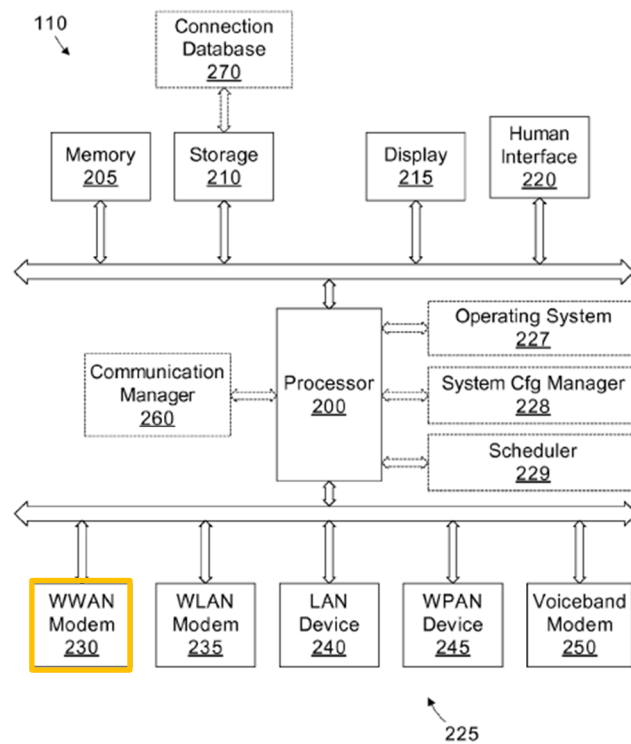


Figure 2

EX1006, Fig. 2 (annotated).

Cole’s “WWAN modem 230” is configured *to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN*. The “WWAN modem 230” can be a “3G channel” so that a mobile device can have network connectivity when in a “public

place.” EX1006, ¶88. Further, in Cole, a mobile device interfaces “with a plurality of communication networks 120, such as a wireless wide area network (WWAN) 130” (e.g., EVDO, HSDPA, 3G, GSM, CDMA, 2G, WiMAX, WiBro). EX1006, ¶31. As explained in Section VI.B, this modem configuration described in Cole represents implementation detail for Shell’s telephonic device 100. EX1003, ¶87; EX1012, ¶¶95, 125. Thus, in Shell-Cole, Shell’s telephonic device incorporates Cole’s WWAN modem 230 and thereby would have enabled data to be wirelessly transmitted over a WWAN network. EX1003, ¶87.

Integration of “WWAN modem 230” from Cole into Shell’s telephonic device 100 enables the Shell-Cole device to utilize data over a WWAN. As described in Cole, WWAN modem 230 is designed to interface with various communication networks such as EVDO, HSDPA, 3G, GSM, CDMA, 2G, WiMAX, and Wi-Bro. EX1006, ¶31. These networks provide the infrastructure necessary for high-speed data transmission and continuous connectivity. EX1003, ¶88. When Shell’s device is configured with this modem, the device gains the ability to connect to these wide area networks, allowing the Shell-Cole device to send and receive data packets required for Internet services like web browsing, email, and media content. EX1003, ¶88. This data communication is facilitated by the modem’s capability to handle the necessary network protocols and manage the

data flow between the device and the WWAN. EX1003, ¶88; EX1005, ¶¶6-8; EX1006, ¶88.

Moreover, Cole's network-selection techniques further enhance the device's ability to maintain robust Internet connectivity. EX1006, ¶6. Specifically, WWAN modem 230 enables the device to automatically select and connect to the most appropriate WWAN based on availability and signal strength. EX1006, ¶¶31, 39, 62. Such dynamic network selection ensures that the device remains connected to a WWAN, thereby providing continuous Internet service. The modem can function as a "3G channel" and is capable of maintaining connectivity even in public places. EX1006, ¶88. In Shell-Cole, this automatic selection and seamless switching between networks allows the device to consistently communicate data for Internet activities, ensuring that users can access online services reliably whenever the device is within the coverage of any supported WWAN. EX1003, ¶89. This capability, derived from Cole's detailed network-interface descriptions, illustrates how the combined disclosures of Shell and Cole render the Shell-Cole device's data communication functionality over a WWAN both feasible and obvious. EX1003, ¶89.

Further, accessing a network resource (as described in Shell) involves transmitting and receiving data over a network, examples of an *Internet service activity* involving data exchange over the Internet. EX1003, ¶90. Examples of such

Internet service activity in Shell include “electronic mail,” “a voice call,” “browsing stock quotes on the Internet,” and messaging via a “messaging application.” EX1005, ¶¶14, 15, 17, 57. Similar examples in Cole include “phone, email, or web” communications. EX1006, ¶¶80, 103-105. As Dr. Butler explains, “the mechanisms and protocols used for accessing general network resources directly support Internet service activities, making them analogous in their operational principles and requirements.” EX1003, ¶90 (citing EX1017).

Therefore, as discussed above, the Shell-Cole combination would have made it obvious for Shell’s telephonic device 100 to include *a wireless wide area network (WWAN) modem to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN.*

3. [1b]: “a wireless local area network (WLAN) modem to receive and transmit Internet data between the device and at least one WLAN when configured for and connected to the at least one WLAN; and”

Shell-Cole renders [1b] obvious based on disclosures in Shell and Cole regarding network connectivity. EX1005, ¶35; EX1006, ¶31, 88, Fig. 2. In Cole’s Figure 2 (below), a mobile device includes a “WLAN modem 235” (*wireless local area network (WLAN) modem*) providing connectivity to a “wireless local area network (WLAN) 140,” confirming that such a modem configuration was well-known by the Critical Date. EX1006, ¶35; EX1003, ¶¶91-94.

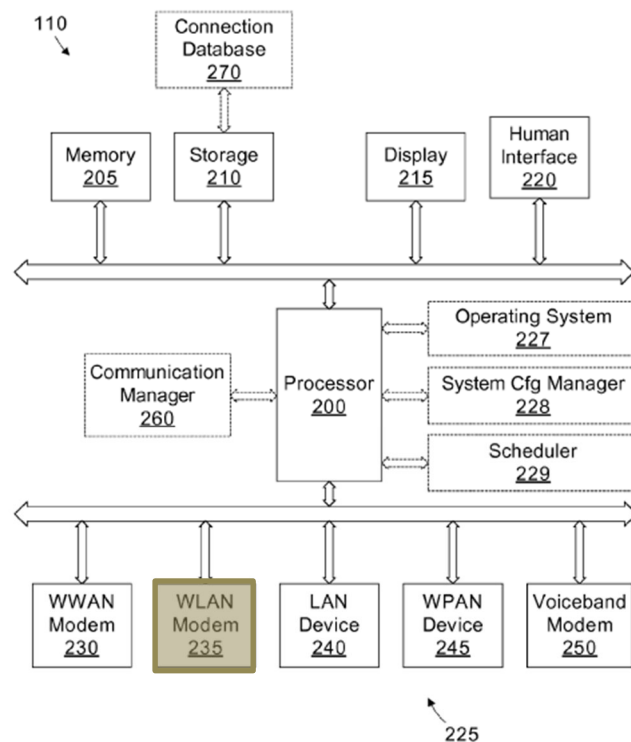


Figure 2

EX1006, Fig. 2 (annotated).

Cole discloses that its “WLAN modem 235” is configured to *receive and transmit Internet data between the device and at least one WLAN when configured for and connected to the at least one WLAN*. The “WLAN modem 235” can be a “Wi-Fi connection” so that a mobile device can have network connectivity “[w]hen the user returns home.” EX1006, ¶88. Further, in Cole, a mobile device interfaces “with a plurality of communication networks 120,” including a “wireless local area network (WLAN) 140” (e.g., IEEE 802.11, Wi-Fi). EX1006, ¶31. As explained in Sections VI.B and VI.C.2, this modem configuration

represents implementation detail for Shell's telephonic device 100, which, as discussed above, discloses wireless connectivity but lacks specific descriptions of the modems that make these connections. EX1003, ¶92; EX1012, ¶¶95, 125. Thus, in Shell-Cole, Shell's telephonic device incorporates Cole's WLAN modem 235 and thereby would have enabled data to be wirelessly transmitted over a WLAN network. EX1003, ¶92; EX1012, ¶¶95, 125.

Integration of "WLAN modem 235" from Cole into Shell's telephonic device 100 enables the Shell-Cole device to utilize data over a WLAN. In Cole, WLAN modem 235 is designed to interface with wireless local area networks, such as IEEE 802.11 and Wi-Fi. EX1006, ¶31. These networks provide the infrastructure necessary for high-speed data transmission and continuous connectivity while reducing network bandwidth costs. EX1003, ¶93. When Shell's device is configured with this modem, the device gains the ability to connect to these local area networks, allowing the Shell-Cole device to send and receive data packets required for Internet services like web browsing, email, and streaming, without requiring access to a wide area network. EX1003, ¶93. This data communication is facilitated by the modem's capability to handle the necessary network protocols and manage the data flow between the device and the WLAN. EX1003, ¶93; EX1005, ¶¶6-8; EX1006, ¶88.

Moreover, as discussed for [1a], Cole's network-selection techniques further enhance the device's ability to maintain robust Internet connectivity by using different modems. EX1006, ¶6. This enables the device to be connected to a WLAN, thereby providing continuous Internet service. The modem can provide connectivity to a "home wireless network" or a "office wireless network" and is capable of maintaining connectivity with lower bandwidth costs. EX1006, ¶38. In Shell-Cole, this automatic selection and seamless switching between networks allows the device to consistently communicate data for Internet activities, ensuring that users can access online services reliably whenever the device is within the coverage of any supported WLAN. EX1006, ¶4; EX1003, ¶94. This capability, derived from Cole's detailed network interface descriptions, illustrates how the combined disclosures of Shell and Cole render the Shell-Cole device's data communication functionality over a WLAN both feasible and obvious. EX1006, ¶4; EX1003, ¶94. Further, as discussed for [1a], accessing a network resource involves the transmission and reception of data over a network, which represents examples of an *Internet service activity* involving data exchange over the Internet. EX1003, ¶94. Examples of such Internet service activity in Shell include "electronic mail," "a voice call," "browsing stock quotes on the Internet," and messaging via a "messaging application." EX1005, ¶¶14, 15, 17, 57. Similar examples in Cole include "phone, email, or web" communications. EX1006, ¶¶80, 103-105.

4. [1c]: “**one or more processors** configured to apply **a stored WWAN differential traffic control policy** to Internet data service provided using the **WWAN modem** and the at least one WWAN, and”

Shell-Cole discloses [1c] based on the processing capabilities of a mobile device taught in Shell. EX1005, ¶40, Fig. 1; EX1006, ¶34. In Shell, telephonic device 100 includes a processor 111 (*one or more processors*) with “complex and flexible general-purpose processing capabilities.” EX1005, ¶40; EX1003, ¶¶95-102. Cole likewise describes a mobile device 110 with a processor 200. EX1006, ¶¶15, 34, 91; *see* Section VI.A.2.

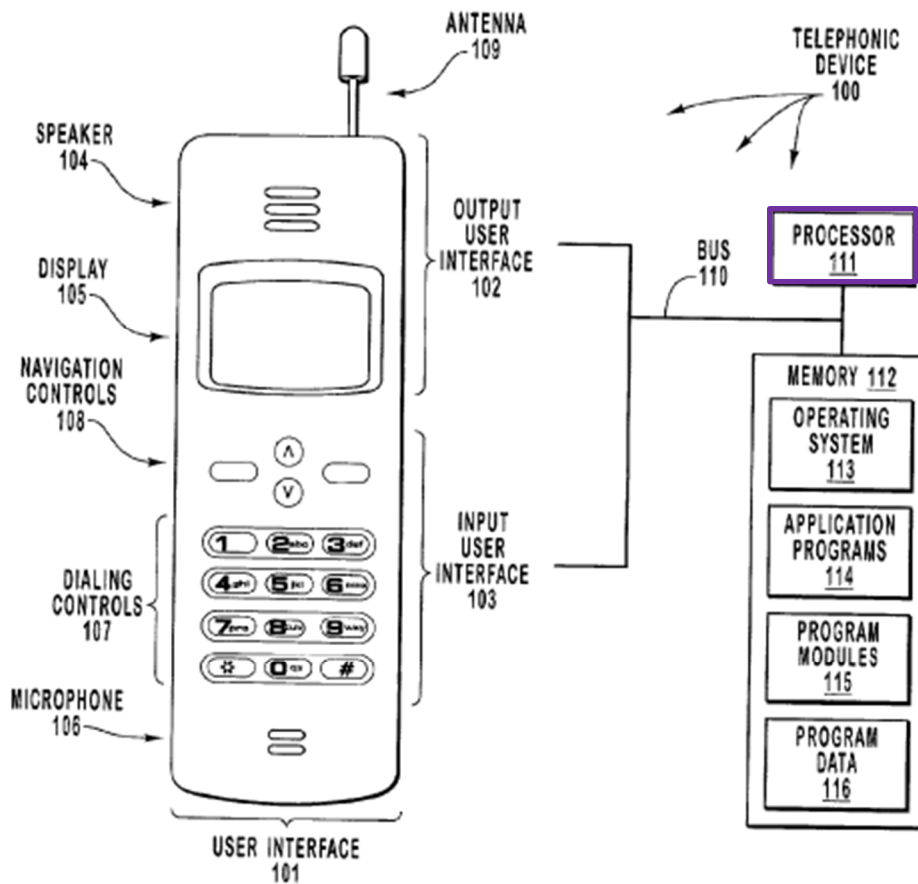


FIG. 1

EX1005, Fig. 1 (annotated).

Shell-Cole also renders obvious application of *a stored [] differential traffic control policy to Internet data service*, including as construed by the district court (see Section V), based on Shell’s disclosures of “connection manager modules” regulating network access by applications on telephonic device 100. EX1005, ¶¶13, 17, 32; EX1006, ¶¶9, 80, 85; EX1003, ¶96. As discussed in Section VI.A.1, in Shell, the “connection manager modules” are configured to regulate network access by applications, e.g., by prioritizing connection requests, determining an

appropriate method for establishing a network connection, granting a connection access request, etc. EX1005, ¶¶13, 17, 32. This is the function of a processor. EX1003, ¶96.

Specifically, a POSITA would have recognized that network connection (*Internet data service*) request handling is performed by Shell's "connection manager modules" and that functionality encompasses examples of applying a *traffic control policy* described in the '359 patent, such as "block/allow settings," "throttle settings," "adaptive throttle settings," and "QoS class settings." EX1001, 85:59-86:24; EX1003, ¶97. Further, as noted in the '359 patent, the "traffic control policy" can be used for "selecting the network from a list of available networks, blocking or reducing access until a connection is made to an alternative network, and/or modifying or replacing a network stack interface of the device to provide for intercept or discontinuance of network socket interface messages to applications or OS functions." EX1001, 85:59-86:24; EX1003, ¶97.

Like these examples, Shell teaches "connection manager modules" (or a "connection planner") receiving a connection request provided by an application that "request[s] access to a remote resource located in a destination network." EX1005, ¶33. The connection manager modules then "analyze[] a variety of factors to determine the most appropriate connection method to connect to the destination network." EX1005, ¶33. Further, "[w]hen the most appropriate

connection method is known, a connection service provider associated with the connection manager may cause a system level Application Programmer Interface ('API') to create a connection from a local computing device to the destination network." EX1005, ¶33. By granting or refusing connection requests, the connection planner is configured to apply decision logic (*stored [] differential traffic control policy*) for network connection (*Internet data service*) request handling (e.g., "block/allow settings"). EX1001, 85:59-86:24; EX1003, ¶98.

Further, in Shell, "connection requests may be prioritized so that higher priority requests cause lower priority requests to release a connection method so it may be used by the higher priority request." EX1005, ¶13. The "[p]riorities may be assigned to connection requests based on applications associated with the connection requests." EX1005, ¶57. This is shown in Figure 5A below:

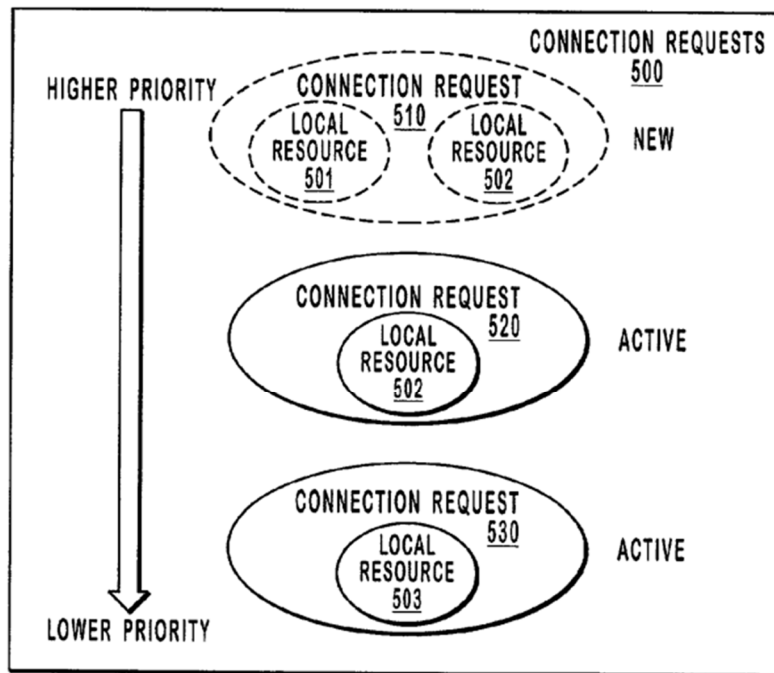


FIG. 5A

EX1005, Fig. 5A.

As shown above, “[c]onnection requests 510, 520, and 530 are prioritized, with connection 510 having the highest priority and connection 530 having the lowest priority.” EX1005, ¶57. A POSITA would have recognized that Shell’s prioritization techniques of network connection requests coincide with the ’359 patent specification’s disclosure of “blocking or reducing access until a connection is made to an alternative network” as an example of applying a “traffic control policy.” EX1001, 85:59-86:24; EX1003, ¶¶99-100. Indeed, Shell’s disclosures demonstrate “rule[s] for controlling network traffic that distinguish[] between two or more things” because the examples control access to a network. EX1003, ¶100.

Further, a POSITA would have found it obvious that decision logic applied by the Shell-Cole device would have provided *a WWAN differential traffic control policy* based on disclosures in Shell and Cole regarding regulating network access in relation to WWAN networks. Shell teaches multiple types of applications, e.g., data application(s), voice application(s), running on telephonic device 100 at the same time and submitting connection requests to an available network, such as a cellular network (e.g., WWAN) provided by a “wireless carrier.” EX1005, ¶¶49, 55. In this example, when a voice application submits a request to connect to the cellular network, the connection manager modules may prioritize the voice application over the data application. EX1005, ¶13. When this occurs, the data application module receives a notification that its “connection” is “being disconnected....” EX1005, ¶92. Thus, decision logic applied in this example relates to access to a cellular network and thereby involves applying a *WWAN differential traffic control policy*. EX1003, ¶101.

Similar to Shell’s decisional logic, Cole expresses that its “connection manager 260” may perform “communication load monitoring and deferred action functions,” which are “useful in many situations.” EX1006, ¶88. For example, if “the user is surfing a web site and selects a file for download[,]” “[t]he file [may be] larger than the user anticipated and is taking a long time to transfer.” EX1006, ¶88. In this example, “the lengthy transfer is conducted using the *WWAN modem*

230 (e.g., a 3G channel), which consumes significant battery power[.]” and thus, “[t]he connection manager 260 identifies the load on the WWAN modem 230 and asks the user to find a better connection.” EX1006, ¶88. As another example, where “a scheduled synchronization starts while the user is traveling[.]” the connection manager 260 may perform traffic monitoring to reduce “consum[ing] battery power.” EX1006, ¶89. The connection manager 260 “notices the load from the synchronization operation and asks the user if it should be deferred (i.e., no better connection is available).” EX1006, ¶89. Cole’s disclosures demonstrate that decisional logic, such the one described in Shell, are applicable to the allocation of resources (such as *WWAN modems*) and connections that supply *Internet data* to devices. EX1003, ¶102.

5. [1d]: “indicate to a particular application, via an application program interface (API), one or more network access conditions based on the applied WWAN differential traffic control policy,”

Shell-Cole renders obvious [1d] because Shell discloses this limitation in at least two ways. EX1003, ¶¶103-117. *First*, in response to an application’s connection request, Shell discloses its connection manager utilizes APIs to notify the application about the allocation of resources. *Second*, when an application makes a piggyback request, Shell discloses that the request and response to the request is handled through APIs associated with the connection manager.

a. Shell discloses its connection manager utilizes APIs to notify the application about the allocation of resources

Shell-Cole renders obvious [1d] because Shell discloses that its “connection manager modules 210 . . . may be configured as **APIs**” or are associated with “**APIs** . . . [that] may be called” (EX1005, ¶¶46, 48) and “may notify application modules 200” [i.e., *indicate to a particular application*] of a “notification associated with the connection to resources” [i.e., *one or more network access conditions*] after the connection manager “cause[s] resources to be released from and allocated to connection methods” [i.e., based on the applied *WWAN differential traffic control policy*] (EX1005, ¶92). *See* EX1005, ¶¶9, 22, 45-48, 70, 90-92; EX1003, ¶¶104-113.

As discussed above, Shell-Cole includes Cole’s *WWAN modem 230*, which would have made it obvious that the decision logic applied by the Shell-Cole device, in some circumstances, involves network connection handling relating to connection requests to access resources associated with a WWAN network (*WWAN differential traffic control policy*). EX1005, ¶¶13, 55, 92; EX1006, ¶¶15, 34, 91; EX1003, ¶105; *see* Sections VI.C.1-VI.C.4.

Specifically, in Shell, “application modules may be notified” of a decision regarding a network connection request, and the “connection manager” may also “provide connection specific parameters” to the application modules. EX1005,

¶92. Thus, a POSITA would have recognized that, in Shell-Cole, processor 111 of telephonic device 100 is configured to provide notification data (e.g., decision, connection-specific parameters) related to a network connection request (*indicate*) to an application module that sent the network connection request (*particular application*). EX1003, ¶106.

The notification data (*one or more network access conditions*) includes the results of the decision logic. EX1005, ¶92. The notification data may relate to a decision regarding the network connection request, e.g., “a connection will be established,” “a connection is currently active,” “connection is being disconnected,” “no connection is currently active,” “a connection to a resource is blocked for some reason,” or “any other notification associated with the connection to resources” (each of which represents *one or more network access conditions*). EX1005, ¶92. As another example, the notification data may include “connection specific parameters,” e.g., “bandwidth, latency, power consumption, cost, physical location, past connection success, or other parameters” (each of which similarly represents *one or more network access conditions*). EX1005, ¶92; EX1003, ¶107.

Further, in Shell-Cole, the connection-manager modules “may be configured as **APIs**” or are associated with “**APIs** . . . [that] may be called. . . .” EX1005, ¶¶46, 48. When the connection manager is configured as an **API**, it “receive[s] standard data when being called by another module,” which simplifies the

establishment of connections. EX1005, ¶46. When the connection manager is associated with an **API**, this allows the connection manager to “be called instead of APIs associated with conventional protocol stacks.” EX1005, ¶48. In either instance, Shell’s teachings are designed so that the “connection manager operates between system level APIs and application programs, providing a level of abstraction.” EX1005, Abstract, ¶¶9, 22, 45-48, 70. Thus, functionality of the connection manager modules of Shell-Cole (e.g., providing notification data to application modules) may be implemented through use of well-known techniques for “providing a level of abstraction” between system-level APIs and applications. EX1005, ¶48; EX1003, ¶108. As one example, Irwin (EX1010), which was published on August 7, 2008, describes a Widget Engine 11 (with a connection manager 16) running on a mobile device and configured to call APIs relating to connection requests. EX1003, ¶108 (citing EX1010, ¶¶13, 23, Fig. 2).

Further, as Dr. Butler explains, “mobile operating systems (e.g., Symbian OS) were known to include connection management services that used **APIs** to allow applications to access network connectivity information.” EX1003, ¶108 (citing EX1011, ¶¶45-46). Indeed, “[a]nother example is EX1008 (Dive-Reclus), which...uses permissions regulating network access by a third-party application using an API call.” EX1003, ¶¶109-113 (citing EX1008, 16:23-30, 17:1-28). Thus, in Shell-Cole, the connection manager modules would have been configured to

provide notification data to application modules through use of well-known **APIs** used to provide network-related information (*via an application program interface (API)*). Indeed, Shell's disclosures would have motivated a POSITA to use an API to communicate Shell's notification data because this form of abstraction would have "reduce[d] the programming complexity associated with creation connections...." EX1005, ¶22. Moreover, a POSITA would have had a reasonable expectation of success in using an **API** to communicate with an application because this was a well-known technique prior to the '359 patent and would have been within a POSITA's general knowledge. EX1003, ¶¶109-113 (discussing EX1016, 1, 3).

b. Shell discloses that a piggyback request and response to the request is handled through APIs associated with the connection manager

Shell-Cole also renders obvious [1d] because Shell discloses that "sync engine 202 may wish to synchronize an electronic mail account [i.e., *a particular application*] only when a connection to the Internet has been established for another application. Applications that wish to piggy back, may call **APIs** associated [i.e., *via an application program interface (API)*] with connection manager modules 210 and be given an open handle," which notifies the application (i.e., *indicates to a particular application*) of an established connection (i.e., *one or more network access conditions*). See EX1005, ¶¶70, 92. A POSITA would

have understood or found it obvious that the notification communicating an established connection is based on the applied *WWAN differential traffic control policy* because its execution involves operation of the connection manager.

EX1003, ¶¶114-117; *see* EX1005, ¶¶9, 22, 45-48, 70, 90-92.

As discussed above (Sections VI.C.4 and VI.C.5.a), Shell-Cole's connection manager applies a decision logic (*WWAN differential traffic control policy*) to determine which applications may utilize the combination's WWAN modem and, using an API, notifies the requesting application about the result of the decision logic (*indicate to a particular application [] one or more network access conditions*). EX1005, ¶¶13, 55, 92; EX1006, ¶¶15, 34, 91; EX1003, ¶114. The use of Cole's *WWAN modem* in the Shell-Cole combination would have made it obvious that the decision logic applied by the Shell-Cole device, in some circumstances, involves network connection handling relating to connection requests to access resources associated with a WWAN network (*WWAN differential traffic control policy*). EX1005, ¶¶13, 55, 92; EX1006, ¶¶15, 34, 91; EX1003, ¶¶114-115.

Shell further suggests that its decision logic applies to Shell's piggyback requests as well. EX1003, ¶116. For example, Shell teaches: "Applications that wish to piggy back, may call *APIs* associated [i.e., *via an application program interface (API)*] with connection manager modules 210 and be given an open

handle,” which notifies the application of an established connection (i.e., *indicates to a particular application // one or more network access conditions*). See EX1005, ¶70. This notification is the exact type or at least mirrors the other notifications that Shell describes more generally. EX1005, ¶¶70, 93 (“application modules may be notified that a connection will be established...”); EX1003, ¶116. Moreover, because Shell’s connection manager (via its connection planner 211) is responsible for processing the piggy back requests, the request is subject to Shell’s decision logic (*WWAN differential traffic control policy*). EX1005, ¶70, EX1003, ¶116. This is how Shell’s piggyback requests are processed because the connection planner “evaluate[s] a variety of data when identifying a most appropriate connection method.” EX1005, ¶56. In other words, the connection planner “execute[s] the [decision] logic” even to the piggyback requests (i.e., applies the *WWAN differential traffic control policy* to them). EX ¶¶58-59, 68-70; EX1003, ¶¶116-117 (explaining that this is consistent with how a POSITA would have understood Shell’s piggyback request); see also EX1005, ¶¶9, 22, 45-48, 90-92; EX1009, 3.

6. [1e]: “wherein the **one or more network access conditions** includes a **network access condition** that indicates the **unavailability** to the **particular application** of an Internet data service that is currently **available** via the **WWAN modem** to a **different application**, but that is indicated as **unavailable** to the **particular application**.”

As discussed in [1d], Shell-Cole renders obvious [1e] because Shell discloses this limitation in at least two ways. EX1003, ¶¶118-129. *First*, in response to an application’s connection request, Shell discloses its connection manager utilizes APIs to notify the application about the allocation of resources, indicating the availability or unavailability of a resource to the application. *Second*, a POSITA would have found it obvious that the responsive notification to Shell’s piggyback request involves an indication of the availability or unavailability of a resource to the application.

a. **Shell discloses allocating a resource to one application but not another based on the priority of the application’s connection request and notifying the application about the availability of the resource**

Shell-Cole renders obvious [1e] because Shell teaches that “telephonic device 100 may request synchronization of an electronic mail account [i.e., a **different application**] and access to a remote web page [i.e., a **particular application**],” using the same resource (i.e., **WWAN modem**) and connection (i.e., **Internet data service**), but teaches that the connection manager may be configured to allocate resources to the synchronization of an electronic mail account based on

priority. EX1005, ¶¶15, 17, 57, 67-68. As with the discussion in [1d] (Section VI.C.5.a), when the connection manager allocates resources, it also notifies the requesting application about the *availability* of the resource and connection. See Section VI.C.5.a. Accordingly, a POSITA would have found [1e] obvious based on Shell's disclosures. EX1003, ¶119.

Shell contemplates allowing some applications to use a connection while preventing other applications from using the same connection. As discussed above, Shell-Cole's connection manager executes decisional logic (i.e., *differential traffic control policy*) to allocate resources to applications based on priorities. See Sections VI.C.4-VI.C.5; EX1003, ¶¶95-117. When resources are allocated or not allocated, the connection manager notifies the application of the decision (i.e., *indicates to a particular application*). See EX1005, ¶92 ("For example, application modules may be notified that a connection will be established, that a connection is currently active, that connection is being disconnected, that no connection is currently active, that a connection to a resource is blocked for some reason, or any other notification associated with the connection to resources."). Shell further discloses that "there may be a plurality of new connection requests at a given time." EX1005, ¶¶15, 68. In other words, "telephonic device 100 may request synchronization of an electronic mail account and access to a remote web page," using the same resource and connection. EX1005, ¶¶15, 68. Or "a voice call [i.e., a

different application] may be given higher [] priority over data calls [i.e., a *particular application*] that share the same modem.” EX1005, ¶13. However, Shell also provides that just because both requests are made does not mean that both will obtain access to the resource. EX1005, ¶¶13, 17, 67-68. For example, the “connection manager may delay the request to browse the Internet until mail synchronization has completed, or take some other action to ensure that the lower priority request does not adversely impact the high priority request.” EX1005, ¶¶17, 67. Shell also discloses that other actions beyond delay may include blocking the application from accessing the connection. EX1005, ¶92 (notifying an application “that a connection to a resource is blocked for some reason”); EX1003, ¶120.

Shell’s decision logic further describes how multiple application connection requests would be handled in Figures 5A and 5B. EX1005, ¶¶57-67, Figs. 4, 5A, 5B. Figure 5A depicts a prioritization order of connection requests 510, 520, and 530 while Figure 5B illustrates the resources (e.g., modems) available to make the connections. EX1005, ¶57.

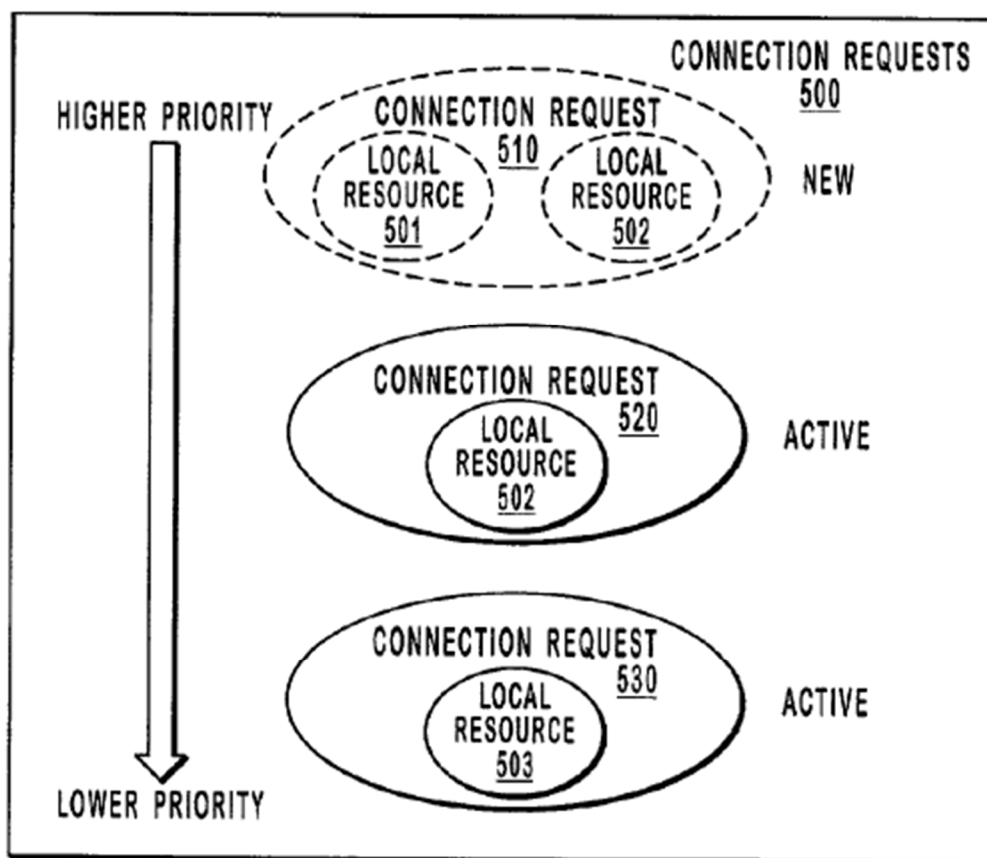


FIG. 5A

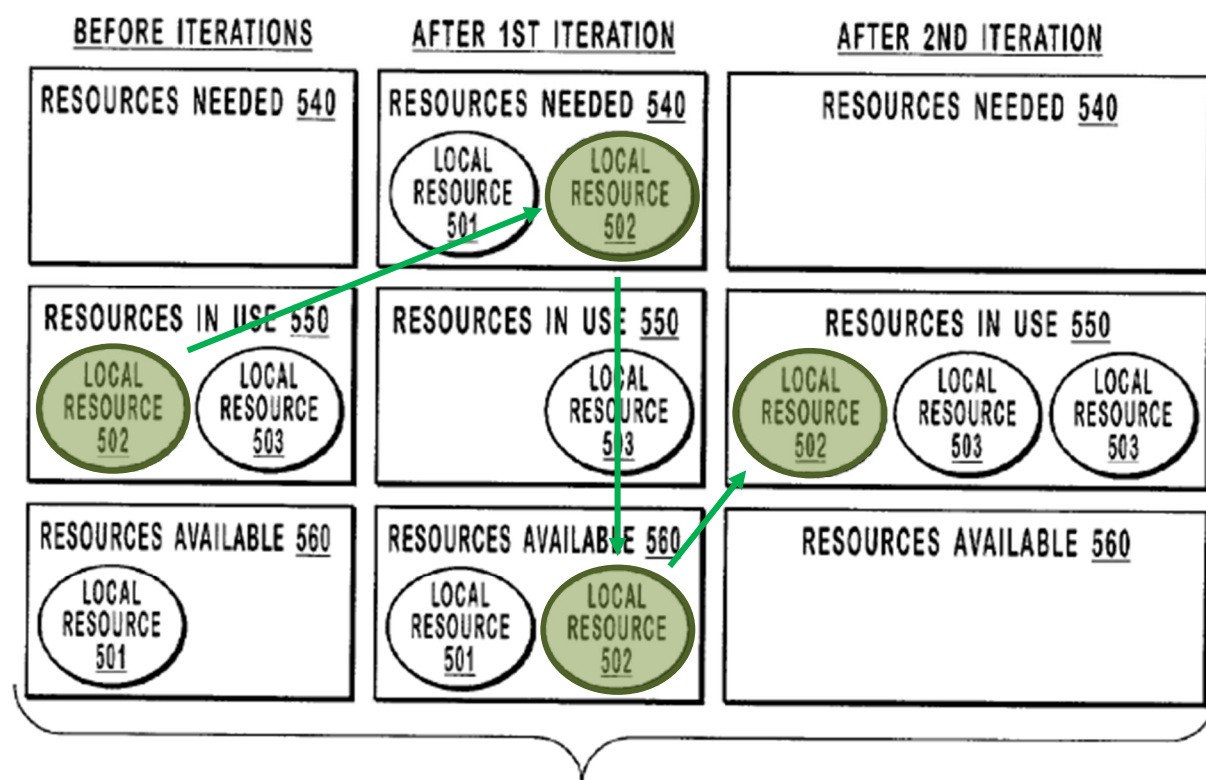


FIG. 5B

EX1005, Figs. 5A, 5B (annotated).

At the start of the process (“Before Iterations”), resources 502 and 503 are in use (by connection requests 520 and 530), and resource 501 is available. EX1005, ¶60. In the first iteration, when a new connection request 510 is made (e.g., when a user requests the synchronization of an electronic mail account [i.e., a *different application*]), the connection manager registers that the new connection request 510 needs local resources 501 and 502. EX1005, ¶¶61-66. As the process continues, the connection manager weighs the priority of the new request 510 against the active connections 520 (e.g., a remote web page [i.e., a *particular*

application]], and 530 to determine how to allocate the resources. EX1005, ¶67.

Because new request 510 has a higher priority than request 520, the connection manager disconnects the application associated with request 520 to resource 502.

EX1005, ¶67. The connection manager then allocates resources 501 and 502 to new request 510, which provides the application associated with new request 510 with a connection. EX1005, ¶67. The above process describes that resource 502 is made *unavailable* to connection request 520 but is *currently available* to new request 510. EX1005, ¶67; EX1003, ¶¶121-122. A POSITA would have understood that this allocation of resources is exactly what is claimed in [1e].

EX1003, ¶122. And indeed, such an allocation of resources is consistent with the '359 patent's description. EX1001, 101:47-102:56.

As discussed above, Shell further discloses that when the connection manager releases or allocates resources, it “may notify application modules 200 of such changes.” EX1005, ¶92. Shell explains that “application modules may be notified that a connection will be established, that a connection is currently active, that a connection is being disconnected, that no connection is currently active, that a connection to a resource is blocked for some reason, or any other notification associated with the connection to resources.” *Id.* In other words, when a resource is made unavailable to one application and available to another application, these applications are each notified of the availability of the connection—this

encompasses a notification to one application that a connection is unavailable to that application while that same connection is currently available to a second application. EX1003, ¶123. For example, the application that sent connection request 520 would receive a notification that resource 502 and the connection are unavailable even though resource 502 is currently available to the application corresponding to request 510. To the extent Patent Owner disputes that Shell teaches availability notifications, a POSITA would have found it obvious to send such notifications based on Shell's notification disclosures. EX1003, ¶123. For example, a POSITA would have understood that a notification that a connection will be established/is currently active is an indication of availability because these connections allow an application with access to the Internet. EX1003, ¶123. A POSITA would have further understood that a notification that a connection to a resource is blocked is an indication of unavailability because the application is prevented from accessing the Internet. EX1003, ¶123.

Based on these disclosures, a POSITA would have understood that when the processor of Shell-Cole's device reallocates resources, it notifies (i.e., *indicates*) the *unavailability* of a connection using resource 502 (i.e., *Internet data service via the WWAN modem*) to the application associated with request 520 (i.e., *particular application*) even though the connection using resource 502 is currently *available* to the application associated with new request 510 (i.e., *different*

application). EX1003, ¶124. Moreover, a POSITA would have found it obvious to use a WWAN over other types of networks. A POSITA would have been motivated to use a WWAN over other applications at least when other networks are unavailable due to a user's location. EX1003, ¶124; EX1007, ¶4. Indeed, using a WWAN would have provided a user with flexibility that other networks may not provide. EX1003, ¶124; EX1007, ¶4. A POSITA would have had a reasonable expectation of success in using a WWAN for applications because it was known that WWANs support the transmission of Internet data. EX1003, ¶124; EX1007, ¶4. Indeed, as discussed in [1c] (Section VI.C.4), Cole even suggests the use of decisional logic to allocate resources (e.g., a WWAN modem) that results in a connection being available or unavailable to an application. EX1006, ¶¶88-89; EX1003, ¶124.

b. A POSITA would have found it obvious that the responsive notification to Shell's piggyback request involves an indication of the availability or unavailability of a resource to the application

Shell and Cole further render obvious [1e] because Shell teaches that “an application [i.e., *particular application*] may request to only access resources in a destination network when another application [i.e., *different application*] is already accessing resources.” EX1005, ¶70. Shell's piggyback requests “may call APIs associated with connection manager modules 210 and be given an open

handle” that notifies the application whether a connection has been established.

EX1005, ¶70. As noted above, Shell instructs that its communication manager notifies applications of, for example, a “notification associated with the connection to resources [i.e., the *availability* or *unavailability* to the applications].” EX1005, ¶92. In other words, Shell describes how resources that establish connections (*i.e., a WWAN modem that provides an Internet data service*) may be available to some applications but not others. EX1005, ¶70, 92. Because Shell discloses allocating a resource to one application while blocking the same resource to another application, a POSITA would have found it both disclosed and obvious to block piggyback requests as well. EX1003, ¶¶125-129.

Shell describes that a piggyback request occurs when one application requests access to resources that another application is already using. EX1005, ¶70. “For example, sync engine 202 may wish to synchronize an electronic mail [i.e., *a particular application*] account only when a connection to the Internet *has been established* [i.e., *currently available*] for another application [i.e., *a different application*].” EX1005, ¶70; EX1003, ¶126 (noting how email synchronization may be a particular or different application based on whether “a user specif[ies] the synchronization by activating a control in a user interface” (quoting EX1005, ¶14)). Shell emphasizes that a piggyback request is an attempt to use the same connection that is already being used, which is contrasted from instances where

“[a] variety of connection methods may be used.” EX1005, ¶70; *see also* EX1005, ¶68. In other words, a piggyback request involves a request for *an Internet data service that is currently available via the WWAN modem to a different application*. EX1003, ¶126.

Shell describes notifying the piggyback requesting application of an established connection (i.e., *indicates to a particular application [] one or more network access conditions*). *See* EX1005, ¶70. Further, as discussed for [1d] (Section VI.C.5.b), as with any other connection request, Shell’s connection planner “execute[s] the [decision] logic” to the piggyback requests (i.e., applies the *WWAN differential traffic control policy* to them), which then triggers the allocation of resources to the applications and notifications of the allocations. EX1005, ¶¶58-59, 68-70; EX1003, ¶127; *see* Section VI.C.5.b. A POSITA would have understood that, from the piggyback-requesting application’s standpoint, the request is similar to any other connection request. EX1003, ¶127. Consequently, a POSITA would have known that the notifications associated with the connections discussed above also would have been applied to the piggy backing requests. EX1003, ¶127. These notifications would include instances when a resource is not allocated and “a connection to a resource is blocked for some reason.” EX1005, ¶92. As discussed above, when a resource is not allocated and/or a connection is blocked, this signifies that the Internet is unavailable to the application. EX1003,

¶127; *see* Section VI.C.6.a. Shell suggests, and a POSITA would have found it obvious, that these same allocation/notification teachings advantageously apply to piggyback requests as well. EX1003, ¶127. For example, because Shell’s decision logic is invoked for piggyback requests (*see* Section VI.C.5.b), a POSITA would have understood that the notification response to piggyback requests may include indications to the requesting application of the *availability/unavailability* of a resource and connection for the same reasons discussed in Section VI.C.6.a. EX1003, ¶127.

A POSITA would have found it obvious and beneficial that Shell-Cole’s connection manager would have applied the same decision logic to piggyback requests that Shell discloses for independent requests based on Shell’s disclosures. EX1003, ¶128. A POSITA would have been motivated to apply the same decision logic to piggyback requests because the purpose of the connection manager is “to simplify the amount of data a user must be familiar with in order to request a connection to a remote resource.” EX1005, ¶¶32, 11-12; EX1003, ¶128. Indeed, applying the same decision logic and notification responses to piggyback requests would have furthered this goal as it would prevent a user from having to engage with the connection request by selecting which network to use. EX1003, ¶128. A POSITA would have had a reasonable expectation of success in implementing the decision logic to piggyback requests because they are already processed by the

connection manager. EX1003, ¶128. Running a piggyback request through the decision logic would merely have involved the same routine programming, as expressly disclosed and encouraged, that a POSITA would have used to process other connection requests from applications. EX1003, ¶128.

In particular, a POSITA would have found it obvious to deny a piggyback request and indicate to the requesting application that the resources and connection are unavailable to the application. EX1003, ¶129. For example, a POSITA would have been motivated to deny a piggyback request because providing the requesting application with access to the resource and connection may risk exceeding the allocated bandwidth of the connection and negatively impacting the performance of the application already using the resource/connection. EX1003, ¶129. Indeed, a POSITA would have found denying piggyback requests to be consistent with Shell's teachings that lower priority requests may be delayed to ensure that the request "does not adversely impact the high priority request." EX1005, ¶17; EX1003, ¶129. A POSITA would have had a reasonable expectation of success in the modification for the same reasons mentioned above, and because Shell expressly sets forth its decision logic to be applied to piggyback requests. EX1003, ¶129.

D. Claim 26: “The wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.”

Shell-Cole renders claim 26 obvious because Shell teaches prioritizing the allocation of resources and connections for active applications (i.e., *a different application*) over other applications based on a user “[a]ctivating a control in a user interface” (i.e., *determine that the different application is active based on user interaction with the different application*). EX1005, ¶¶13-14, 51, 57, 92; EX1003, ¶¶130-136.

In Shell-Cole, a user’s interaction with an application may instigate connection requests, allowing its *processor* to determine that the *different application* is active. EX1005, ¶¶14, 51. For example, to connect to a network, Shell describes that “a user may request access to a resource that is accessed by connecting to a destination network.” EX1005, ¶14. Shell discloses that “[t]his may be as simple as a user specifying the synchronization of an electronic mail account by activating a control in a user interface.” EX1005, ¶¶14, 51. Indeed, it is the user’s interaction with the application that instigates the application to send a connection request, which triggers the decision logic that subsequently establishes a network connection based on priority. *See* Sections VI.C.4-VI.C.6. A user may also place a voice call that then causes the connection manager to execute its

decision logic and determine whether the voice call should be allocated resources over a data call that would share the same modem. EX1005, ¶¶8, 13. A POSITA would have recognized that a user's input can initiate a regular connection request (Section VI.C.6.a) and may also serve as the foundational connection request upon which a secondary application attempts to piggyback (Section VI.C.6.b). EX1003, ¶131.

As discussed above in [1e] (Sections VI.C.6.a and VI.C.6.b), Shell describes that connection-manager modules may assign priorities to network-connection requests "based on the current state of applications associated with the connection requests." EX1005, ¶¶57-92. Shell provides that "a request from an application that is currently active may be assigned a higher priority than a request from an application running in the background." EX1005, ¶¶57-92. And the currently active requests include those initiated by the user. EX1005, ¶57; EX1003, ¶132.

Based on these disclosures, a POSITA would have understood or would have found it obvious that Shell's processor is configured to assess whether a request is coming from an application that the user is interacting with as part of the prioritization of connection requests as discussed in Sections VI.C.6.a and VI.C.6.b. EX1005, ¶¶51, 57, 92; EX1003, ¶133. This is because user input is a way of activating an application, and in certain circumstances like a phone call or web browsing, the processor would only provide these applications with an Internet

connection when they are in use. EX1005, ¶¶51, 57, 92; EX1003, ¶133.

Accordingly, a POSITA would have known and considered obvious that the assessment of whether a request is coming from an in-use application is a determination that the different application is an active application based on a user interaction with the different application. EX1003, ¶133.

As discussed in Sections VI.C.5.a and VI.C.6.a, Shell describes that when a new connection request 510 is made (e.g., when a user requests the synchronization of an electronic mail account or a user places a voice call [i.e., *different application*]), the connection manager registers that the user interacted with the email application, which initiated the request. EX1005, ¶¶13-14, 61-66. The new connection request 510 asks for local resources 501 and 502. EX1005, ¶¶61-66. And as the process continues, the connection manager weighs the priority of the new request 510 against the active connections 520 (e.g., a remote web page or a data request [i.e., *particular application*]) and 530 to determine how to allocate the resources. EX1005, ¶¶13-14, 67. Because new request 510 has a higher priority than request 520, the connection manager disconnects the application associated with request 520 to resource 502—for example, because the email application is currently active while the web page is operating in the background. EX1005, ¶¶13-14, 67. The connection manager then allocates resources 501 and 502 to new request 510, which provides the application associated with new

request 510 a connection. EX1005, ¶67. The above process describes that resource 502 is made unavailable to connection request 520 but is currently available to new request 510. EX1005, ¶67. Shell-Cole therefore describes and renders obvious that a user's input or interaction with an application may be part of the decision logic and taken into account in the determination of a priority assignment. EX1003, ¶134.

As a result, applications that are currently active based on user interaction with the application (e.g., synchronizing email or placing a voice call) are connected to a network while other applications that are running in the background are not provided with access to the network. EX1003, ¶135. As discussed in Sections VI.C.6.a and VI.C.6.b, Shell-Cole's device monitors the allocation of resources and connections and notifies the applications about the status of their requests. As applied to the decision flow described in Figures 5A/5B of Shell, an application that is operating in the background (i.e., *particular application*), would be notified of the reallocation of resources and connection status while the same resources and connection are currently available to the active application. EX1003, ¶¶135-36. Therefore, a POSITA would have found claim 26 obvious. EX1003, ¶¶135-36.

As discussed in Sections VI.C.5.b and VI.C.6.b, the same decision logic and notifications apply to piggyback requests. EX1003, ¶¶125-29. A POSITA would

have understood a user interfacing with an application may instigate a regular connection request (Section VI.C.6.a) and may also instigate the foundational connection request upon which a second application attempts to piggyback (Section VI.C.6.b). *See* EX1005, ¶70 (“an application [i.e., *particular application*] may request to only access resources in a destination network when another application [i.e., *different application*] is already accessing resources.”); EX1003, ¶136. Consequently, a POSITA also would have found claim 26 obvious based on the disclosed techniques of piggybacking in Shell-Cole’s device. EX1003, ¶136.

VII. Ground 2: Shell, Cole, and Flack Render Obvious Claim 1

Petitioner provides Ground 2 as a basis for additional arguments in Ground 3 related to claim 26. As noted above, claim 1 has been disclaimed.

A. Overview of Flack

Flack discloses systems and methods of “selectively establish[ing] network connectivity by monitoring, on a client, communications activity incident to accessing a network.” EX1007, Abstract. Flack’s client-side system may be executed on a wireless phone or other computing device. EX1007, ¶¶38, 62. Once communication activities are detected, the system “identif[ies] an application associated with the communications activity,” accesses a “communications configuration associated with the application,” and “selectively establish[es]

network connectivity based on the communications configuration.” EX1007, ¶¶3, 17, 50, 54. EX1003, ¶71.

Flack identifies and seeks to address inefficiencies in maintaining constant network connectivity. For example, Flack recognizes that “background and ambient processes, such as spyware protection and update tools generate small amounts of traffic.” EX1007, ¶16. Processes like these unnecessarily keep network connections active even when they “do not represent mission critical communication or represent communications that may be exchanged at a later time when communications do not incur the same costs for the user.” EX1007, ¶16. Flack teaches that “[i]t might be better to establish and preserve connectivity for only selective user applications,” providing the example of only providing connectivity for applications that involve “user actions that generate visual feedback useful to the user in perceiving the user application.” EX1007, ¶16; EX1003, ¶72.

Flack discloses a system that uses a configurable profile to determine how and when applications may access a network. This includes applying policies for various applications such as email, instant messaging, web browsing, and media playback. EX1007, ¶¶22, 31, 50, 55, 59-61. The system is configured to distinguish between user-initiated applications and non-user-initiated applications. EX1007, ¶¶8, 50. Considering user activity in its policy determination allows the

system to allow applications to connect to a particular network only when the user is using an application (EX1007, ¶19) or to allow a background connection only if the network connection is already established by a user application (EX1007, ¶24). *See also* EX1007, ¶¶5-6, 42, 48-58, , Figs. 4, 4A; EX1003, ¶73.

Flack is also analogous art to the '359 patent. For example, Flack is in the same field of endeavor: management of an application's ability to connect to wireless networks. *Compare* EX1007, ¶¶16, 17, *with* EX1001, Abstract. Flack is also reasonably pertinent to at least one problem addressed by the '359 patent: optimizing use of data on an application-by-application basis. *Compare* EX1007, ¶17, *with* EX1001, 4:65-5:20. EX1003, ¶74.

B. Independent Claim 1

1. [1pre]-[1d]:

Shell-Cole renders obvious [1pre]-[1d] for the same reasons discussed above in Ground 1. *See supra* Sections VI.B-VI.C.5; EX1003, ¶¶137-138; *see also* EX1003, ¶¶83-117.

2. [1e]:“wherein the **one or more network access conditions** includes **a network access condition** that indicates the **unavailability** to the **particular application** of an Internet data service that is currently **available** via the **WWAN modem** to a **different application**, but that is indicated as **unavailable** to the **particular application**.”

To the extent that Patent Owner disputes the discussion above regarding Shell-Cole’s denial of an application’s piggyback request (Sections VI.C.5.b, VI.C.6.b), a POSITA also would have found [1e] obvious because Flack teaches the “**first program** may be **blocked** from accessing the network based on network connectivity that has **already been established** for the **second program**.” EX1007, ¶5; EX1007, ¶¶139-145.

As discussed above (*supra* Section VII.A), Flack discloses a system that allows applications access to a network based on a communications configuration. EX1007, ¶20. Flack instructs that its system includes a processor that executes its process. EX1007, ¶¶39, 63. As part of this process, the client “identifies an application associated with the communications activity (420). Identifying an application may include identifying the source of a communications activity Some of these applications may represent user applications that a user has launched in attempting to retrieve information from the Internet (e.g., a web browser, an email application, or an Instant Messaging application).” EX1007, ¶50. Flack discloses that “[i]t might be better to establish and preserve connectivity for

only selective user applications,” rather than applications/programs that operate in the background. EX1007, ¶16. Flack suggests that there are two ways to respond to a connection request from a first program to use the same “network connectivity that has already been established for [a] second program.” EX1007, ¶¶5-6, 42, 57-58, Figs. 4, 4A. These include “enabl[ing] access to the network” and “block[ing] access[.]” to the network. EX1007, ¶¶5, 58.

a. Motivation to Combine Shell, Cole, and Flack with a Reasonable Expectation of Success

Shell, Cole, and Flack relate to wireless devices with similar functions and uses. EX1003, ¶141. For example, Shell, Cole, and Flack all detail wireless devices that are configured to connect to different networks. EX1005, ¶13, EX1006, ¶35; EX1007, ¶44. Shell, Cole, and Flack describe methods that involve the use of policies to determine whether an application should be connected to a network. EX1005, ¶57; EX1006, ¶36; EX1007, ¶5.

Given Shell, Cole, and Flack’s similarities, a POSITA would have been motivated with a reasonable expectation of success to consider their combined teachings to optimize network usage, battery and power consumption, and cost. EX1003, ¶142. Aiming to “establish[.] and maintain[.] connections in a manner that promotes cost efficiency and efficient utilization of available connection methods,” Shell describes a connection manager that enables a user to merely “request the

resource he or she wishes to connect to and the connection manager devices which connection method to utilize.” EX1005, ¶93; *see also* EX1005, ¶¶1, 12-13. Shell suggests that this may be simple when “a user specify[ies] the synchronization of an electronic mail account by activating a control in a user interface.” EX1005, ¶14. But this may be more complicated “when a user attempts a voice call on a telephone line that is currently being used for a data connection.” EX1005, ¶14. Shell provides other examples of when multiple applications request access to the same resource, such as piggybacking, when “sync engine 202 may wish to synchronize an electronic mail account only when a connection to the Internet has been established for another application.” EX1005, ¶70. Although Shell suggests that these requests can be made and blocked, Shell does not expressly discuss every technique for efficiently handling piggyback requests. EX1005, ¶70. A POSITA would thus have looked to other references to better understand the outcomes of a piggyback request and to efficiently handle them under different network and operational circumstances. EX1003, ¶142.

A POSITA would have looked to and appreciated Flack for these details because Flack discusses balancing network access for multiple applications. EX1003, ¶143. As noted above, Flack describes that “[i]t might be better to establish and preserve connectivity for only selective user applications,” rather than applications/programs that operate in the background. EX1007, ¶16. Flack

suggests two ways in which a processor may respond to a connection request when a second program is already running: “enabl[ing] access to the network” and “block[ing] access[.]” to the network. EX1007, ¶¶5-6, 42, 57-58, Figs. 4, 4A.

A POSITA would have been motivated to implement Flack’s specific **allowing/blocking** decisions into the processing of Shell-Cole’s piggyback request because doing so would further Shell’s goal of promoting “cost efficiency and efficient utilization of available connection methods.” EX1005, ¶93; EX1003, ¶144. For example, blocking access to a background application’s update/synchronization as Flack suggests would “ensure that the lower priority request does not adversely impact the high priority request” as Shell provides. EX1005, ¶17. This would have been an efficient utilization of available connection methods and resources because it would prevent the overloading of a connection method. EX1003, ¶144; EX1006, ¶¶9, 74. Furthermore, it would have reduced the risk of latency in active applications. EX1003, ¶144; EX1006, ¶¶9, 74. Moreover, as described in Ground 1 [1d] and [1e], Shell-Cole provides notification data (i.e., indicates a network access condition) to applications as part of its process. *See* Sections VI.C.5-VI.C.6. A POSITA would have included Flack’s specific **allowing/blocking** decisions in Shell-Cole’s notification data because this information informs the requesting application whether it has been allocated

resources, allowing the application to know whether it can execute a task that prompted the request. EX1003, ¶144.

A POSITA would have had a reasonable expectation of success in implementing Flack’s blocking step into Shell-Cole because Shell already is configured to block certain connection requests. EX1003, ¶145; EX1005, ¶92. Shell-Cole already includes the physical components to process application data network requests (*see* Sections VI.C.2-VI.C.4); therefore, a POSITA would have understood that implementing Flack’s teachings into Shell-Cole would merely have required routine programming that was well within a skilled artisan’s toolbox prior to the earliest claimed priority date. EX1003, ¶145. Moreover, as in the Shell-Cole combination, Flack discloses that this system may be implemented on a cell phone communicating on a WWAN, such as a “wireless circuit-switched network.” EX1007, ¶15. A POSITA would have thus understood that Flack is applicable to WWANs/WWAN modems because it suggests compatibility with cellular networks. EX1003, ¶145. Combining the references would have been nothing more than applying a known technique—i.e., incorporating Flack’s denial of connection requests—to a similar mobile device communications system—i.e., Shell-Cole’s system. EX1003, ¶145. The implementation would have yielded predictable results: denying Internet access to an application that is not actively being used and notifying the application of that denial. EX1003, ¶145. And the

result would have been advantageous and obvious enhancements in network bandwidth usage, power consumption, and application activity. EX1003, ¶145.

VIII. Ground 3: Shell, Cole, and Flack Render Obvious Claim 26

- A. **Claim 26: “The wireless end-user device of claim 1, wherein the **one or more processors** are further configured to determine that the **different application** is **an active application based on a user interaction** with the **different application**.”**

Shell-Cole (Ground 1) further in view of Flack and Shell, Cole, and Flack (Ground 2) also render obvious claim 26 because Flack discloses that when its “client determines that communications activity [within an application] **represents user actions**,” then the client may establish a network connection. EX1007, ¶¶19, 8, 16-17, 50; EX1003, ¶¶146-152.

As discussed above, Shell teaches that “a request from an application [i.e., **different application**] that is currently active may be assigned a higher priority than a request from an application [i.e., **particular application**] running in the background.” EX1005, ¶¶57, 92. Shell notes that requests from the application modules may come from a user “[a]ctivating a control in a user interface.” EX1005, 14, 51. In other words, Shell describes that the request may be made based on a user’s interaction with the device, allowing its **processor** to determine that the **different application** is active. EX1005, ¶¶14, 51; EX1003, ¶147.

Disclosing a process that is implemented on the same types of mobile devices as Shell, Flack teaches allowing applications access to a network based on a communications configuration. EX1007, ¶20. Flack instructs that its client includes a processor that executes its process. EX1007, ¶¶39, 63. As part of this process, the client “identifies an application associated with the communications activity (420). Identifying an application may include identifying the source of a communications activity Some of these applications may represent user applications that a user has launched in attempting to retrieve information from the Internet (e.g., a web browser, an email application, or an Instant Messaging application).” EX1005, ¶50. “If the client determines that communications activity represents user actions, the client may establish network connectivity by, for example, establishing a dialup connection, so that the client may support the user activity within the user application.” EX1007, ¶¶19, 8, 16-17, 50. But, as noted above in Section VII.B.2, Flack describes how certain applications may be allowed or blocked access to a network if another application is already using the network connection. EX1007, ¶¶5, 50; EX1003, ¶148. Flack thus further evidences that a POSITA would have understood a user interfacing with an application may instigate a regular connection request (Section VI.C.6.a) and may also instigate the foundational connection request upon which a secondary application attempts to piggyback (Sections VI.C.6.b, VII.B.2). *See* EX1007, ¶5 (“The first program may

be blocked from accessing the network based on network connectivity that has already been established for the second program.”); EX1003, ¶148.

A POSITA would have found it obvious to implement Flack’s determination of application status based on **user activity** as part of Shell-Cole’s traffic policy related to regular connection requests (Ground 1, Sections VI.C.5.a, VI.C.6.a, VI.D) and piggyback requests (Ground 1, Sections VI.C.5.b, VI.C.6.b, VI.D). EX1003, ¶149. For example, a POSITA would have implemented Flack’s “determin[ation] that communications activity represents user actions” as part of Shell-Cole’s decision logic that makes priority determinations based on, in-part, “a request from an application that is currently active ... [and] a request from an application running in the background.” EX1005, ¶57. This combination would provide resources and network connections to applications that are currently active based on user interaction with the application (e.g., synchronizing email or placing a voice call) while preventing resource and network access to other applications that are running in the background. EX1003, ¶149. As discussed in Sections VI.C.6.a and VI.C.6.b, Shell-Cole’s device monitors the allocation of resources and connections, and notifies the applications about the status of their requests; the addition of Flack would not modify Shell-Cole’s notification steps. EX1003, ¶149. Indeed, as applied to the decision flow described in Figures 5A/5B of Shell, an application that is operating in the background (i.e., *particular application*), would

be notified of a change in availability of resources and connection status as a consequence of the reallocation of resources and connection status, while the same resources and connection are currently available to the active application. EX1003, ¶149. Therefore, a POSITA would have found claim 26 obvious based on the incorporation of Flack's determination of application status based on **user activity** into Shell-Cole's decision logic that processes user-initiated connection requests. EX1003, ¶149.

A POSITA would have further found it obvious to implement Flack's determination of application status based on user activity as part of Shell-Cole-Flack's traffic policy as discussed in Ground 2 (Section VII.B.2). EX1003, ¶150. For example, as described above (Sections VI.C.5.b, VI.C.6.b), the same decision logic and notifications apply to piggyback requests as well as other requests. EX1003, ¶150, *see also* EX1003, ¶¶126-129. Consequently, a POSITA would also have found claim 26 obvious based on the disclosed technique of piggybacking. EX1003, ¶150.

A POSITA would have been motivated to implement both changes for the same reasons. EX1003, ¶151. For example, a POSITA would have been motivated to do so because users would have wanted the application that they are currently using to be connected to the Internet (when the application requires a data connection to operate). EX1003, ¶151. Indeed, without an Internet connection,

some applications, such as a web-based application, would have been unusable; consequently, a POSITA would have wanted to prioritize providing Internet data to active applications over inactive or background applications. EX1003, ¶151. A POSITA would have had a reasonable expectation of success in this combination because it would have involved minor programming changes to Shell's prioritization preferences. EX1003, ¶151. Indeed, Shell already contemplates that "[p]riorities may also be assigned to connection requests based on the current state of applications associated with the connection requests." EX1005, ¶57. And, therefore, prioritizing based on the active state of an application would have merely been one of these possible states. EX1003, ¶151. A POSITA would have also found that this modification would have been consistent and compatible with Shell's teachings that a "variety of connection methods may be used" to satisfy different connections request. EX1005, ¶¶69, 17. This is because a WWAN-based connection might be the best option for certain applications at a specific time when the user is actively using the application while a WLAN-based connection might be more appropriate for other applications at the same time. EX1006, ¶¶4, 64. As an example, using a WWAN-based connection to download a document may not be an appropriate connection because it could incur extra charges; however, a WLAN may not incur the same charges. EX1003, ¶151; EX1005, ¶6; EX1006, ¶¶88.

A POSITA would also have appreciated that Flack's mechanism for determining user-driven communications activity and initiating network connectivity complements Shell-Cole's prioritization of active applications.

EX1003, ¶152. A POSITA would have recognized the benefit of integrating these teachings to enhance resource allocation and connection management, ensuring that user-engaged applications receive timely access to network resources.

EX1003, ¶152. The integration of Flack into Shell-Cole would have amounted to nothing more than the use of a known technique (Flack's determination of application status based on user activity) to improve similar devices (Shell-Cole's telephonic device) in a similar way (using connection policies), and combining these prior art teachings according to the known, expressly disclosed methods to yield the predictable results described above. EX1003, ¶152. In this Ground 3, incorporating Flack's teachings into Shell-Cole provides a *processor* that is configured to determine whether a user's interaction with an application (i.e., *a different application*) instigates connection requests, meaning that its *processor* determines that the application is *an active application* based on *user interaction*. Sections VI.C.5-VI.C.6, VI.D; VII.B.2; EX1007, ¶¶8, 16-19, 39, 50, 63. Therefore, claim 26 would have been obvious over Shell-Cole-Flack. EX1003, ¶152.

IX. Mandatory Notices

A. Real Parties-in-Interest

Google LLC³ and Apple Inc. are the real parties-in-interest.

B. Related Matters

The '359 patent is or has been involved in the following proceedings:

Name	Number	Forum	Filed
<i>Headwater Research LLC v. Apple Inc.</i>	7-25-cv-00379	W.D. Tex.	8/27/25
<i>Headwater Research LLC v. Google LLC</i>	7-25-cv-00372	W.D. Tex.	8/27/25
<i>Headwater Research LLC v. Dish Network Corp. et al</i>	2-25-cv-00905	E.D. Tex.	8/27/25
<i>Headwater Research LLC v. Charter Communications, Inc. et al</i>	2-25-cv-00904	E.D. Tex.	8/27/25
<i>Headwater Research LLC v. Comcast Cable Communications, LLC d/b/a Xfinity et al</i>	2-25-cv-00909	E.D. Tex.	8/28/25

³ Google LLC is a subsidiary of XXVI Holdings Inc., which is a subsidiary of Alphabet Inc. XXVI Holdings Inc. and Alphabet Inc. are not real parties-in-interest to this proceeding.

Name	Number	Forum	Filed
<i>Headwater Research LLC v. Samsung Electronics Co., Ltd. et al</i>	2-23-cv-00641	E.D. Tex.	12/29/23
<i>Samsung Electronics Co., Ltd. v. Headwater Research LLC</i>	IPR2024-01407	PTAB	9/6/24

C. Lead and Back-Up Counsel

Lead Counsel	Back-Up Counsel
Erika H. Arner, Reg. No. 57,540 erika.arner@finnegan.com Finnegan, Henderson, Farabow, Garrett, & Dunner LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2754 Fax: 202-408-4400	Joshua L. Goldberg, Reg. No. 59,369 joshua.goldberg@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413 Phone: 202-408-6092 Fax: 202-408-4400 Daniel C. Tucker, Reg. No. 62,781 daniel.tucker@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2793 Fax: 202-408-4400 Benjamin A. Saidman, Reg. No. 69,325 benjamin.saidman@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 271 17th Street NW, Suite 1400 Atlanta, GA 30363-6209 Phone: 404-653-6510 Fax: 404-653-6444 Daniel F. Klodowski, Reg. No. 72,693

Lead Counsel	Back-Up Counsel
	daniel.klodowski@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413 Phone: 202-408-4216 Fax: 202-408-4400 Elliot Cook, Reg No. 61,769 elliot.cook@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2738 Fax: 202-408-4400 Michael V. Young, Reg. No. 61,180 michael.young@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2788 Fax: 202-408-4400 Alexander M. Boyer, Reg. No. 66,599 alexander.boyer@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2759 Fax: 202-408-4400 Pier D. DeRoo, Reg. No. 69,340 pier.deroo@finnegan.com Finnegan, Henderson, Farabow,

Lead Counsel	Back-Up Counsel
	Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413 Phone: 202-408-4418 Fax: 202-408-4400 Christopher C. Johns, Reg. No. 68,664 Christopher.Johns@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413 Phone: 202-408-4155 Fax: 202-408-4400 Kassandra M. Officer, Reg. No. 74,083 Kassandra.Officer@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413 Phone: 202-408-4270 Fax: 202-408-4400 Bradford C. Schulz, Reg. No. 75,006 bradford.schulz@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2739 Fax: 202-408-4400 William C. Neer, Reg. No. 78,874 Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413

Lead Counsel	Back-Up Counsel
	Phone: 202-408-4054 Fax: 202-408-4400 Christopher B. Anderson, Reg. No. 77,898 christopher.anderson@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2765 Fax: 202-408-4400 Kai Rajan, Reg. No. 70,110 kai.rajan@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413 Phone: 202-408-4363 Fax: 202-408-4400 Umber Aggarwal, Reg. No. 76,098 umber.aggarwal@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2423 Fax: 202-408-4400 Abhay A. Watwe, Reg. No. 59,461 abhay.watwe@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2716

Lead Counsel	Back-Up Counsel
	Fax: 202-408-4400 Kathleen C. Galleher, Reg. No. 80,837 kathleen.galleher@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Phone: 571-203-2448 Fax: 202-408-4400 Connor M. McGregor, Reg. No. 79,634 connor.mcgregor@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413 Phone: 202-408-4363 Fax: 202-408-4400 Heather J. Ren, Reg. No. 82,089 heather.ren@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue NW Washington, DC 20001-4413 Phone: 202-408-4024 Fax: 202-408-4400 Yi Yang, <i>pro hac vice</i> to be filed yi.yang@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 2 Seaport Lane, 6th Floor Boston, MA 02210 Phone: 617-646-1642 Fax: 202-408-4400

D. Service Information

Petitioner consents to being served electronically at the e-mail addresses provided above and at HeadwaterIPRTeam@finnegan.com.

X. Grounds for Standing

Petitioner certifies that the '359 patent is available for IPR and that Petitioner is not barred or estopped from requesting IPR challenging the patent claims on the grounds in this Petition.

XI. Conclusion

Petitioner requests IPR and cancellation of claim 26. The Office may charge any required fees to Deposit Account No. 06-0916.

Date: October 31, 2025

Respectfully submitted,

By: /Erika H. Arner/
Erika H. Arner, Lead Counsel
Reg. No. 57,540

CERTIFICATION OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24, Petitioner certifies that the word count of this Petition for *Inter Partes* Review (exclusive of any table of contents, table of authorities, mandatory notices under § 42.8, certificates of service and word count, and list of exhibits or claim listing) as measured by Microsoft Word is 13,996 words.

Dated: October 31, 2025

By: /Erika H. Arner/
Erika H. Arner, Lead Counsel
Reg. No. 57,540

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(a), the undersigned certifies that on October 31, 2025, a copy of the foregoing Petition for *Inter Partes* Review, the associated power of attorney, and Exhibits 1001-1019 were served by FedEx Priority Overnight on the correspondence address of record indicated in the Patent Office's Patent Center website for U.S. Patent No. 9,179,359:

Headwater Research LLC
c/o Farjami & Farjami LLP
26522 La Alameda Ave., Suite 360
Mission Viejo, CA 92691

Date: October 31, 2025

By: /Lisa C. Hines/
Lisa C. Hines
Case Manager
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, LLP